CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

Branch-and-Price for the Truck and Trailer Routing Problem with Time Windows

Sophie N. Parragh
Jean-François Cordeau

October 2015

CIRRELT-2015-54

Bureaux de Montréal :
Université de Montréal
Pavillon André-Aisenstadt
C.P. 6128, succursale Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

Branch-and-Price for the Truck and Trailer Routing Problem with Time Windows

Sophie N. Parragh[1,*], Jean-François Cordeau[2]

[1] Department of Business Administration, University of Vienna, Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria

[2] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Logistics and Operations Management, HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montréal, Canada H3T 2A7

Abstract. Motivated by a situation faced by infrastructure service providers operating in urban areas with accessibility restrictions, we study the truck and trailer routing problem with time windows (TTRPTW). In this problem the vehicle fleet consists of trucks and trailers which may be decoupled. A set of customers has to be served and some of the customers can only be accessed by the truck without the trailer. This gives rise to the planning of truck-and-trailer routes containing truck-only subroutes, in addition to truck-only routes and truck-and-trailer routes without subroutes. We propose a branch-and-price algorithm for the TTRPTW, using problem specific enhancements in the pricing scheme and alternative lower bound computations. We also tailor an adaptive large neighborhood search algorithm to the TTRPTW in order to obtain good initial columns. When compared to existing metaheuristic algorithms we obtain highly competitive results. Instances with up to 100 customers are solved to optimality with the proposed branch-and-price algorithm.

Keywords: truck and trailer, vehicle routing, column generation, large neighborhood search, branch-and-price.

Acknowledgements. We wish to thank Vincent Yu and Ulrich Derigs for having provided us with their data sets. The first author was supported by the Austrian Science Fund (FWF): T514-N13. This support is gratefully acknowledged.

---

* Corresponding author: sophie.parragh@univie.ac.at

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2015

1 Introduction

In Austria and many other countries, several infrastructure service provi-ders operate in cities where some areas have accessibility restrictions. These restrictions usually refer to pedestrian zones that cannot be accessed by car and they have to be taken into account in the daily routing and scheduling of field employees providing maintenance and installation services at customer locations. As a result, service providers must often design vehicle routes comprising subroutes that are performed by technicians on foot. In order to better understand the impact of subroute planning from a methodological point of view, this paper develops a branch-and-price algorithm and tailors an adaptive large neighborhood search method for the truck and trailer routing problem with time windows.

The truck and trailer routing problem (TTRP) [5] consists in determining a set of least cost or distance routes visiting a given number of customers with accessibility constraints. The vehicle fleet is composed of trucks which may or may not pull a trailer. One part of the customers can be visited by a truck pulling a trailer (denoted as trailer customers). The other part can only be visited by a truck alone (denoted as truck customers). In order to serve these two types of customers, three types of routes can be planned: truck only routes, i.e. routes that are carried out by a truck alone; complete vehicle routes that are carried out by a truck pulling a trailer without any intermediate decoupling; and complete vehicle routes that are carried out by a truck and a trailer containing truck-only subroutes. Each of these subroutes has to start and end at a given trailer customer where the trailer is decoupled and then re-coupled again at the end of the subroute. Both the trucks and the trailers have a capacity and shifting loads from the trailer to the truck before departing on truck-only subroutes is possible. An important aspect of this problem is that a trailer may not be picked up by a truck different from the one which decoupled it, and each customer may only be visited once. This implies that a decoupling point cannot be used more than once: whenever a customer is used as a decoupling point it has to be served. However, several consecutive subroutes may start and end at the same decoupling point.

The truck and trailer routing problem with time windows (TTRPTW) has first been stated and solved by Lin et al. [24]. The only difference with respect to the TTRP is that customers as well as the depot have time windows. Lin et al. [24] assume that trailer customers that serve as decoupling points have to be visited before their first truck-only subroute starts. Derigs et al. [10] do not make this restrictive assumption, i.e. the customer used to park the trailer may be visited before or after any of its truck-alone subroutes. They show that this leads to considerable improvements in terms of solution quality and it is the problem version considered in this paper.

In the following, we first give a brief overview of the related literature in Section 2. In Section 3 we define the considered problem in further detail and we propose a path-based model. In Section 4, we design a column generation scheme to solve its linear programming relaxation, which is embedded into a branch-and-bound tree. Several enhancements relying on ideas from the literature as well on new ideas are discussed. An adaptive large neighborhood search (ALNS) algorithm, relying on known destroy and repair operators, and its adaptation to the TTRPTW is described in Section 5. The main purpose of the ALNS is to populate the column pool in our branch-and-price scheme. However, it also achieves highly competitive results when compared to best known results on instances from the literature. These results are presented in Section 6, where we also evaluate the performance of the proposed branch-and-price scheme. For this purpose we solve instances with 25, 50 and 100 customers and we illustrate the impact of the proposed methodological enhancements. Section 7 concludes the paper and provides potential directions for future research.

2 Related work

Truck and trailer routing problems have only recently received more attention in the literature. From a methodological point of view, most of the works published on the TTRP are metaheuristic algorithms. Chao [5] and Scheuerer [34] propose tabu search algorithms, Lin et al. [23] a simulated annealing based method, Villegas et al. [39] a greedy randomized adaptive search procedure combined with path relinking, and Derigs et al. [10] apply their metaheuristic solution framework, combining attribute based hill climber and record-to-record travel with large neighborhood search and local search. Two groups of authors combine heuristic and exact ideas in the form of so-called matheuristics: Caramia and Guerriero [4] combine mathematical programming with a local search algorithm and Villegas et al. [40] use heuristic column generation. A multi-objective version has been addressed by Tan et al. [37] with a hybrid multi-objective evolutionary algorithm.

The TTRPTW has only been addressed by Lin et al. [24], who propose a simulated annealing algorithm, and by Derigs et al. [10] who apply their solution framework not only to the TTRP but also to the TTRPTW and to a problem version in which load transfers are not allowed. Much earlier, a real-world inspired problem involving trucks, trailers, time windows and a heterogeneous vehicle fleet had been addressed by Semet and Taillard [35].

The generalized truck and trailer routing problem (GTTRP) has been introduced by Drexl [13]. In this variant, additional decoupling (so-called transshipment) locations, time windows and a heterogeneous vehicle fleet are considered. Trucks and truck-and-trailer combinations do not only differ in terms of their capacities but also in terms of their fixed and distance-dependent costs. Drexl [13] proposes a branch-and-price algorithm to solve the GTTRP. The pricing subproblems are solved by a labeling algorithm based on dynamic programming, considering the following resources: cost, collected load, transferred load, time, visited locations and trailer in use; and a label may only dominate another label residing at the same node if the current trailer parking positions are equal. We will show that, in our case, dominance is also possible under different circumstances. Drexl [13] solves instances with up to 10 truck customers, 10 trailer customers and 10 transshipment locations to optimality with a time limit of 11,100 seconds. He also derives a heuristic algorithm from the proposed branch-and-price framework and solves some of the TTRP instances of Chao [5].

Very recently, Belenguer et al. [3] have solved the single truck and trailer routing problem with satellite depots by branch-and-cut. In this problem, a single truck towing a detachable trailer has to serve a given set of customers only accessible by truck. Appropriate satellite depots or transshipment locations have to be selected and are used to park the trailer and transfer load from the trailer to the truck. Several families of valid inequalities are proposed. The largest instance solved has 100 customers and 10 satellite depots.

Truck and trailer routing problems are also related to two-echelon VRPs (2E-VRPs). In the 2E-VRP goods are delivered from a depot to satellite facilities and from the satellite facilities they are distributed to the actual customers. Direct shipping from the depot to a customer is not possible. Split deliveries to satellite facilities are allowed but not to the customers. The number of customers that can be served by a satellite facility depends on the amount of goods shipped from the depot to this facility. The best performing exact algorithm is the method of Baldacci et al. [2] in which the problem is decomposed into several multi-depot VRPs with side-constraints. They solve almost all instances from the literature with 50 customers and up to 5 satellites.

In Drexl [14] a modeling framework is introduced for the vehicle routing problem with trailers and transshipments (VRPTT). No fixed assignment of trailers to trucks is considered and load transfers between vehicles are allowed. The author explains how, for example, the 2E-VRP can be modeled as a VRPTT. Drexl [15] proposes branch-and-cut algorithms for the VRPTT, considering several families of valid inequalities. The best performing method uses

combinatorial Benders cuts [6] and solves some instances with up to 8 customers, 8 potential transshipment locations and 8 vehicles.

Finally, both the 2E-VRP and the TTRP are related to the location routing problem. Recent surveys covering these problem classes can be found in Cuda et al. [8], Drexl and Schneider [16] and Prodhon and Prins [29].

3 Problem formulation

The TTRPTW can be modeled on a complete directed graph $G(V, A)$ where V is the set of all vertices which contains the origin depot d^+, the destination depot d^-, and the set of customers N; and A denotes the set of arcs. Each arc $(i, j) \in A$ is associated with a travel cost c_{ij} and a travel time t_{ij}. There is a fleet of m trucks available to serve the different customers and each truck may also be combined with a trailer. The trucks have a capacity of Q_{truck} and the trailers a capacity of $Q_{trailer}$. The set of customers can be divided into the set of truck customers N_{truck} and the set of trailer customers $N_{trailer}$, where $N_{truck} \cap N_{trailer} = \varnothing$ and $N = N_{truck} \cup N_{trailer}$. Each customer $i \in N$ has a demand q_i, a time window $[e_i, l_i]$, in which service has to start, and a service time s_i.

The travel costs of a route c_r are computed by summing over all travel costs c_{ij} associated with arcs (i, j) contained in route r. Furthermore, we use parameter b_{ir} equal to 1 to indicate that customer i is served by route r, and 0 otherwise. The set Ω contains all feasible routes which may be truck-only routes, trailer routes without subroutes, and trailer routes with subroutes, respecting capacity constraints of the trucks and the trailers, time window constraints and accessibility restrictions at the customers, i.e. a customer from the set N_{truck} may only be served on a truck-alone subroute or a truck-only main route. Finally, we define binary variables x_r taking value 1 if route r is used and 0, otherwise. Using the above notation, we formulate the TTRPTW as the following path-based model:

$$\min \sum_{r \in \Omega} c_r x_r \tag{1}$$

$$\sum_{r \in \Omega} b_{ir} x_r = 1 \qquad \forall i \in N \tag{2}$$

$$0 \leq \sum_{r \in \Omega} x_r \leq m \tag{3}$$

$$x_r \in \{0, 1\} \qquad \forall r \in \Omega. \tag{4}$$

The objective function (1) minimizes the total routing costs. Constraints (2) ensure that each customer is served by exactly one route. Constraints (3) make sure that at most m routes are used. In the variant considered in this paper, the number of trucks is not restricted, i.e. we can set $m = |N|$.

The above model can be solved by means of a branch-and-price algorithm which is described next.

4 Branch-and-price

Branch-and-price methods embed column generation into branch-and-bound trees. At the root node, the linear programming relaxation of the model must be solved. Since the set Ω is typically too large to be enumerated completely, column generation provides a means to dynamically generate only those routes (columns) which have the potential to be part of an optimal solution.

Branch-and-price algorithms have been the method of choice for vehicle routing problems with time window restrictions [12] and they have recently also shown to outperform branch-and-cut based approaches on the capacitated vehicle routing problem [28]. For an introduction to column generation we refer to the tutorial of Lübbecke and Desrosiers [25] and to the book by Desaulniers et al. [11]. As already explained, to populate the column pool Ω, promising x_r variables have to be generated. However, some initial columns can also be constructed to start the method. We generate two initial columns per customer, both corresponding to single customer routes, starting and ending at the depot. The first of these two columns has a high cost of M and does not consume a vehicle resource, i.e. it has a coefficient of 0 in constraint (3). The second has the actual routing cost of the single customer route but it consumes a vehicle resource. Furthermore, we also use a dummy column that corresponds to a giant tour serving all customers, uses all possible arcs and does not consume a vehicle resource. Using this set of initial columns guarantees that a feasible solution can always be generated, even after branching has been performed (as is described later). In order to identify additional promising columns, dual information from the so-called restricted master problem is used. The master problem in our case corresponds to the linear relaxation of model (1)–(4), i.e. the integrality restrictions on the x_r variables are relaxed to $x_r \geq 0$, and the restricted master problem is obtained by replacing the set Ω with a subset Ω' of generated columns. We denote by π_i the dual variable of constraint (2) for customer i and by β the dual variable of constraint (3). Then, a promising column is any route which has negative reduced cost. The reduced cost of a given route r is computed as follows:

$$c_r - \sum_{i \in N} a_{ir} \pi_i - \beta. \tag{5}$$

In order to identify the route with the lowest reduced cost, an (elementary) shortest path problem with resource constraints has to be solved. Elementary means that each customer may only appear at most once on this route. In our case, a decoupling point may appear more than once. Therefore, the elementary property only holds for customers which are not used as decoupling points. However, like in the general elementary case, $b_{ir} = 1$ no matter how often a decoupling point i actually appears along a route r. Then, column generation consists of the following steps. First, the restricted master problem is solved on Ω'. Second, the dual information is retrieved. Third, the subproblem is solved and fourth the column pool Ω' is updated with the newly identified columns. Steps one to four are repeated until no new promising routes or columns can be found. In this case, the optimal solution of the relaxed model (1)–(4) on Ω' corresponds to the optimal solution on the complete set Ω. If this solution is also integer, the optimal solution to model (1)–(4) has been found. Otherwise, we have to resort to branching and column generation is used to solve the linear programs at the nodes of the branch-and-bound tree.

The process of identifying promising columns in step three is usually referred to as pricing. In the following, we first describe how we solve the pricing subproblem. Thereafter, we discuss several enhancements to the pricing phase and we explain the employed branching scheme. Finally, we discuss how additional lower and upper bounds can be obtained.

4.1 Solving the pricing subproblem

Elementary shortest path problems can be solved by means of labeling algorithms [see, e.g., 18] where each label represents a partial path starting at the origin depot, and ending at the current *node*. A sketch of the labeling algorithm we use is given in Algorithm 1. It generates paths starting at the origin depot and ending at the end depot and it terminates as soon as at least 10 labels of negative reduced cost have been generated at the end depot or the list of unprocessed labels is empty.

Algorithm 1 A labeling algorithm

```
1: retrieve dual information and graph G(V, A)
2: generate label(s) at origin depot
3: put generated label(s) into list of unprocessed labels U
4: repeat
5:    pop first label L from U
6:    retrieve node of L
7:    if L is not dominated by another label at node then
8:       for each arc (i, j) ∈ A such that node = i do
9:          if feasible extension to j possible then
10:            extend label L along arc (i, j) and generate new label(s) at j
11:            put generated label(s) into list of unprocessed labels U
12:         end if
13:      end for
14:   end if
15: until U = ∅ or 10 labels of negative reduced costs have been generated at end depot
16: return all paths associated with negative reduced cost labels at end depot
```

In our labeling algorithm, each label carries the following information, in addition to a pointer to the parent node:

node	the node of the current label
cost	the travel cost until *node*
time	the time when leaving *node*
load	the total load along the tour including that of *node*
trailer	if a trailer is in use on the tour $(0, 1)$
sub	if the current node is on a subroute $(0, 1)$
loadsub	the total load along the current subroute
dec	the last decoupling point, if on a subroute
dec_visited	if the last decoupling point was already visited $(0, 1)$
$V_{visited}$	the set of customers visited along the tour
redcost	the reduced cost until *node*

We note that resource *dec_visited* is not necessary in the elementary pricing case: the information that the decoupling point was already visited or not is also stored in $V_{visited}$. However, in order to use the *ng*-route relaxation [1], which is described later, such a resource allows us to simply replace the set $V_{visited}$ by a different definition without any further changes to the labeling algorithm.

In the following, we first describe when a feasible extension along an arc (i, j) is possible (Line 9 of Algorithm 1) and how and how many new labels are generated at j. Then, we discuss enhancements to the labeling algorithm, which are followed by a description of the employed dominance rules (Line 7 of Algorithm 1).

4.1.1 Label extension

Label L can be extended along arc (i, j) if the following holds:

1. node j has not been visited before ($\notin V_{visited}(L)$) or it is the decoupling point ($j = dec(L)$)
2. if node j is a truck customer, no trailer is currently present (either $sub(L) = 1$ or $trailer(L) = 0$)
3. if node j was not visited before, capacity restrictions can be respected ($load(L) + q_j \leq Q_{truck} + trailer(L)Q_{trailer}$)

4. if node j was not visited before, its time window can be respected ($time(L) + t_{ij} \leq l_j$)
5. the return to the depot within the planning horizon is possible:
   (a) if j was not visited yet: $\max(time(L) + t_{ij}, e_j) + s_j + t_{jd^-} \leq l_{d^-}$
   (b) if j was already visited ($j = dec(\mathrm{L})$): $time(L) + t_{ij} + t_{jd^-} \leq l_{d^-}$
6. if we are currently on a subroute ($sub(L) = 1$), we also have to make sure:
   (a) if j is not the decoupling point and the decoupling point was not visited yet that it is possible to reach the decoupling point within its time window: $\max(time(L) + t_{ij}, e_j) + s_j + t_{j,dec(L)} \leq l_{dec(L)}$
   (b) if j is not the decoupling point that the truck capacity is not exceed: $loadsub(L)+q_j \leq Q_{truck}$
   (c) that j is not the end depot
7. if j is the end depot that negative reduced costs are possible: $redcost(L) + c_{id^-} < 0$

If L can be feasibly extended along arc (i, j), we distinguish four different types of label extensions: (A) decoupling and serving, (B) decoupling but not serving, (C) no re- or decoupling, and (D) re-coupling (and serving if not served previously). How many labels are generated depends on j:

1. if j is a truck customer or the end depot: only extension (C) is performed and one new label at node j is generated.
2. if j is a trailer customer and a trailer is currently present: extensions (A), (B) and (C) are performed and three new labels are generated at node j.
3. if j is the decoupling point and was not visited yet: extensions (A), (B) and (D) are performed and three new labels are generated at node j.
4. if j is the decoupling point and was already visited: extensions (B) and (D) are performed and two new labels at node j are generated.

We now describe the different extension rules in further detail:
(A) decoupling and serving. In this case the trailer is decoupled at j and j is also served. The new label L' at j is generated as follows:

$$
\begin{aligned}
&node(L') = j \\
&cost(L') = cost(L) + c_{ij} \\
&time(L') = \max(time(L) + t_{ij}, e_j) + s_j \\
&load(L') = load(L) + q_j \\
&trailer(L') = trailer(L) \\
&sub(L') = 1 \\
&loadsub(L') = 0 \\
&dec(L') = j \\
&dec\_visited(L') = 1 \\
&V_{visited}(L') = V_{visited}(L) \cup \{j\} \\
&redcost(L') = redcost(L) + c_{ij} - \pi_j
\end{aligned}
$$

(B) decoupling but not serving. In this case, the trailer is decoupled at j but j is not served and a new label L' is generated at j as follows:

$$
\begin{array}{l}
node(L') = j \\
cost(L') = cost(L) + c_{ij} \\
time(L') = time(L) + t_{ij} \\
load(L') = load(L) \\
trailer(L') = trailer(L) \\
sub(L') = 1 \\
loadsub(L') = 0 \\
dec(L') = j \\
dec\_visited(L') = dec\_visited(L) \\
V_{visited}(L') = V_{visited}(L) \\
redcost(L') = redcost(L) + c_{ij}
\end{array}
$$

(C) no re- or decoupling. In this case node j is visited but no de- or recoupling is performed. The corresponding new label L' at j is generated as follows:

$$
\begin{array}{l}
node(L') = j \\
cost(L') = cost(L) + c_{ij} \\
time(L') = \max(time(L) + t_{ij}, e_j) + s_j \\
load(L') = load(L) + q_j \\
trailer(L') = trailer(L) \\
sub(L') = sub(L) \\
loadsub(L') = \begin{cases} loadsub(L) + q_j, \text{ if } sub(L') = 1, \\ loadsub(L), \text{ otherwise} \end{cases} \\
dec(L') = dec(L) \\
dec\_visited(L') = dec\_visited(L) \\
V_{visited}(L') = V_{visited}(L) \cup \{j\} \\
redcost(L') = redcost(L) + c_{ij} - \pi_i
\end{array}
$$

(D) recoupling. In this case j is the decoupling point and if $dec\_visited = 0$, j is also served, otherwise it is not. Thus a new label L' is generated at j as follows:

$$
\begin{array}{l}
node(L') = j \\
cost(L') = cost(L) + c_{ij} \\
time(L') = \begin{cases} \max(time(L) + t_{ij}, e_j) + s_j, \text{ if } dec\_visited(L) = 0, \\ time(L) + t_{ij}, \text{ otherwise} \end{cases} \\
load(L') = \begin{cases} load(L) + q_j, \text{ if } dec\_visited(L) = 0, \\ load(L), \text{ otherwise} \end{cases} \\
trailer(L') = trailer(L) \\
sub(L') = 0 \\
loadsub(L') = 0 \\
dec(L') = -1 \\
dec\_visited(L') = 0 \\
V_{visited}(L') = \begin{cases} V_{visited}(L) \cup j, \text{ if } dec\_visited(L) = 0, \\ V_{visited}(L), \text{ otherwise} \end{cases} \\
redcost(L') = \begin{cases} redcost(L) + c_{ij} - \pi_i, \text{ if } dec\_visited(L) = 0, \\ redcost(L) + c_{ij}, \text{ otherwise} \end{cases}
\end{array}
$$

The labeling algorithm starts with the generation of two labels at the origin depot. In the first, no trailer is considered, i.e. $trailer = 0$, whereas in the second a trailer is assumed to be present, i.e. $trailer = 1$. All other label entries are initialized as follows: ($node$, $cost$, $time$, $load$, sub, $loadsub$, dec, $dec\_visited$, $V_{visited}$, $redcost$) $= (0, 0, e_0, 0, 0, 0, -1, 0, \{0\}, -\beta)$ in both labels.

4.1.2 Improving the label extension rules

It is possible to improve the label extension rules described above. The improvements we propose are based on the notion that, given that the triangle inequality holds for the cost and the time matrices, in an optimal solution no subroute exists that contains only trailer customers:

- a label L where $sub(L) = 1$ cannot be extended to $dec(L)$ as long as the subroute does not contain at least one truck customer. In order to keep track of whether or not a truck customer has been visited on the current subroute, we add the resource *truckcustvisited* to each label.
- if a label L is to be extended along arc (i, j) and $sub(L) = 1$, $truckcust\text{-}visited(L) = 0$ and j is a trailer customer but not the decoupling point, it can only be extended to j if it is still possible to reach at least one truck customer taking time and load considerations into account, i.e. there has to exist at least one truck customer k for which the following holds:
  - $k \notin V_{visited}(L)$
  - $loadsub(L) + q_j + q_k \leq Q_{truck}$
  - $load(L) + q_j + q_k \leq Q_{truck} + Q_{trailer}$
  - $\max\{e_j, time(L) + t_{ij}\} + s_j + t_{jk} \leq l_k$
  - case 1: decoupling point was not visited yet:
    $\max\{e_k, \max[e_j, time(L) + t_{ij}] + s_j + t_{jk}\} + s_k + t_{k,dec(L)} \leq l_{dec(L)}$
    $\max\{e_{dec(L)}, \max[e_k, \max(e_j, time(L) + t_{ij}) + s_j + t_{jk}] + s_k + t_{k,dec(L)}\} + s_{dec(L)} + t_{dec(L),d^-} \leq l_{d^-}$
    case 2: decoupling point was already visited:
    $\max\{e_k, \max(e_j, time(L) + t_{ij}) + s_j + t_{jk}\} + s_k + t_{k,dec(L)} + t_{dec(L),d^-} \leq l_{d^-}$
- a similar logic can be applied whenever a new subroute is started. This should only be possible from label L along arc (i, j), where j will become the decoupling point, if there exists at least one truck customer k for which the following holds:
  - $k \notin V_{visited}(L)$
  - case 1: j was not visited yet: $load(L) + q_j + q_k \leq Q_{truck} + Q_{trailer}$
    case 2: j was already visited: $load(L) + q_k \leq Q_{truck} + Q_{trailer}$
  - $time(L) + t_{ij} + t_{jk} \leq l_k$
  - $\max(e_k, time(L) + t_{ij} + t_{jk}) + s_k + t_{k,j} + t_{j,d^-} \leq l_{d^-}$

4.1.3 Dominance rules

Before a label is considered for extension, we check if it is dominated by an existing label at the same node. It is rather straightforward that a label L dominates a label L' if the following conditions hold and at least one inequality is strict (in the case of equality, we keep the label that was already extended):

$node(L) = node(L')$
$time(L) \leq time(L')$
$redcost(L) \leq redcost(L')$
$trailer(L) = trailer(L')$
$load(L) \leq load(L')$
$sub(L) = sub(L')$
$dec(L) = dec(L')$
$dec\_visited(L) = dec\_visited(L')$
$truckcustvisited(L) = truckcustvisited(L')$ or $truckcustvisited(L) = 1$
$loadsub(L) \leq loadsub(L')$
$V_{visited}(L) \subseteq V_{visited}(L')$

The first criterion states that both labels have to reside at the same node. The second and third criteria state that the current time and the reduced cost at L has to be smaller than or equal to the current time and the reduced cost at L', respectively. Furthermore, in both labels a trailer has to be either present or not. This guarantees that the total available capacity is the same on both paths represented by the labels. In combination with the criterion that the current load has to be smaller than or at most equal in the dominating label, we guarantee that the remaining capacity in the dominating label is at least as large as in the dominated one. The next three criteria take care of the subroute logic: labels are only comparable if both are currently on the main route or if both are currently on a subroute having the same decoupling point. In addition, whether or not the decoupling point was already visited must be equal in both labels. Regarding the requirement that at least one truck customer has to appear on each subroute, in the dominating label L this resource has to take value 1 if it also takes value 1 in the dominated one. Furthermore, the remaining capacity on the subroute also has to be at least equal in the dominating label. Finally, we make sure that the dominating label can reach at least as many nodes as the dominated one by comparing the sets $V_{visited}$ of L and L'. Otherwise, we cannot guarantee that label L has the same options regarding the collection of reduced costs as label L'; e.g. if the path leading to label L consists of customers 1, 2 and 3, and the path of label L' of 1, 2 and 4, then it is not guaranteed that when L is extended to 4 and L' to 3 that the label emanating from L will be associated with the lower reduced costs.

4.1.4 Improving the dominance rules

The above dominance rules do not allow a comparison of partial paths that are not in the same state in terms of being on a subroute starting at the same decoupling point. However, we observe that this restrictive assumption is not always necessary. Assume that for label L $sub(L) = 0$ and $trailer(L) = 0$. Then this label can dominate a label L' if $sub(L') = 1$ and $trailer(L') = 1$ in the following case:

$node(L) = node(L')$
$time(L) \leq time(L')$
$redcost(L) \leq redcost(L')$
$Q_{truck} - load(L) \geq Q_{truck} + Q_{trailer} - load(L')$ (remaining capacity)
$V_{visited}(L) \subseteq V_{visited}(L')$

The reason is that if the remaining capacity on a truck-only tour is at least as large as the remaining capacity on the trailer tour with a subroute and the same or more nodes are still reachable (i.e. $\notin V_{visited}$), then these nodes can be reached along the truck tour at least as fast as on the trailer tour.

Assume now the opposite situation. For label L, $sub(L) = 1$, $trailer(L) = 1$, $dec\_visited(L) = 1$, and $truckcustvisited = 1$. Then this label can dominate a label L' with $sub(L') = 0$ and $trailer(L') = 0$ if the following holds:

$$node(L) = node(L')$$
$$time(L) + t_{node(L),dec(L)} + t_{dec(L),node(L)} \leq time(L')$$
$$redcost(L) + c_{node(L),dec(L)} + c_{dec(L),node(L)} \leq redcost(L')$$
$$Q_{truck} + Q_{trailer} - load(L) \geq Q_{truck} - load(L') \text{ (remaining capacity)}$$
$$V_{visited}(L) \subseteq V_{visited}(L')$$

Furthermore, all nodes that are not reachable from the current node (i.e. all those for which conditions 3, 4 or 5 in Section 4.1.1 do not hold) can be inserted into set $V_{visited}$. A similar rule has also been employed by other authors, e.g., by Ropke and Cordeau [30].

4.2 General enhancements

We now describe several ideas that we use to speed up the pricing step in our column generation scheme.

4.2.1 Preprocessing

In order to avoid extensions along arcs that cannot lead to a feasible solution, we pre-process the graph and remove all arcs (i, j) from the set A that cannot be part of a feasible solution:

1. If i and j are both truck customers, we check if visiting customer i before customer j is feasible with respect to time windows. If not, arc (i, j) is removed from the graph.
2. If $q_i + q_j > Q_{truck}$ and both customers are truck customers, arcs (i, j) and (j, i) are removed from the graph.

Furthermore, we also remove the arc connecting the origin depot with the end depot and all arcs leading to the origin depot or leaving the end depot.

4.2.2 Leveled pricing

In a first set of experiments with our column generation scheme, we observed that in some cases the optimal solution does not contain any subroutes. This led us to the idea of a leveled pricing scheme to further speed up the exact labeling algorithm. Instead of considering all possibilities when it comes to label extension, we use a three-stage approach. In the first stage, we try to find paths of reduced costs for truck-only routes. If we fail to find any, we try to find paths of reduced costs for trailer routes, without considering the generation of subroutes. Only if this fails as well, do we try to generate paths with subroutes.

4.2.3 Heuristic pricing

The pricing subproblem does not necessarily have to be solved to optimality at every step. Promising columns may also be produced by a heuristic pricing algorithm. In our heuristic labeling algorithm we use ideas from [26]. However, we do not use a two-stage pricing scheme but we pre-compute a set of potential subroutes for each of the trailer customers. They are generated as follows. In a first step, for each trailer customer, we identify the six closest customers. If at least one of these customers is a truck customer, the respective trailer customer is considered as a potential starting point for a subroute. Then, for each of these potential starting points, we enumerate all sets of size two to six of these six closest customers, such that each set contains at least one truck customer. For each of these subsets, we generate all feasible routes visiting all customers in the set. For each route k we compute the minimum duration d_k, a time window $[e_k, l_k]$, the total load q_k and its total cost c_k; c_k includes the costs for going from the starting point to the first customer on the subroute and from the last customer on the subroute back to the starting point (d_k does not). In order to compute the minimum possible duration of the subroute, we compute the total waiting time w_k along the route as well as the forward time

slack f_k [33]. Using this information, the total duration and the time window of a subroute k are computed as follows:

$$\begin{aligned} d_k &= \sum_{(i,j)\in A(k)} t_{ij} + \sum_{i\in V(k)} s_i + \max\{0, w_k - f_k\} \\ e_k &= e_{i_f} + \min\{w_k, f_k\} \\ l_k &= e_{i_f} + f_k, \end{aligned}$$

where $A(k)$ is the arc set of subroute k, $V(k)$ the vertex set, and i_f the first customer along the subroute. Among all those feasible routes that are generated for a given set of customers, we only keep non-dominated ones. A route k dominates a route k' if $e_k < e_{k'}, l_k > l_{k'}, d_k < d_{k'}, c_k < c_{k'}$. Whenever the pricing algorithm is called, we go through all pre-computed subroutes once and check if they have negative reduced cost. If they do, they are considered in the pricing heuristic. Finally, we assume that in each trailer route, only one trailer customer can be used as a temporary depot and at most three times consecutively (i.e. at most three consecutive subroutes may be appended).

Based on the observations made above we also consider a version of our heuristic pricing algorithm that uses the leveled pricing idea. However, instead of considering three stages, we only consider two. In the first, we only generate truck routes and in the second we generate trailer routes with and without subroutes, appending complete subroutes from the pre-computed pool.

4.2.4 *ng*-route relaxation

Due to its success in column generation based algorithms for other vehicle routing problems, we also use the *ng*-route relaxation of Baldacci et al. [1]. In order to do so, we determine for each customer i the set N_i which contains the 10 closest customers to i as well as the customer itself. Then, instead of set $V_{visited}$, we consider the set Π which, in every extension step, is updated as follows: $\Pi(L') = (\Pi(L) \cap N_j) \cup \{j\}$.

4.2.5 Stabilization

In order to alleviate both the heading-in as well as the tailing-off effect in the root node, we relax the set-partitioning type constraints (2) to set-covering type constraints and we use interior point stabilization as introduced by Rousseau et al. [32]. It works as follows. In a first step, the original restricted master problem is solved and all columns that are part of the basis are identified. Their indices are stored in the set R^*. Furthermore, the set of nodes for which the covering constraint is not tight is stored in set $\bar{C}$. Then, the following LP is solved with different values for u_i which are randomly drawn in the (0,1) interval:

$$\min \sum_{r\in\Omega} c_r x_r \tag{6}$$

$$\sum_{r\in\Omega} b_{ir} x_r \geq u_i \qquad \forall i \in N \setminus \bar{C} \tag{7}$$

$$\sum_{r\in\Omega} b_{ir} x_r \geq -\infty \qquad \forall i \in \bar{C} \tag{8}$$

$$-m \leq \sum_{r\in\Omega} x_r \leq m \tag{9}$$

$$x_r \geq 0 \qquad \forall r \in \Omega \setminus R^* \tag{10}$$

$$x_r \text{ free} \qquad \forall r \in R^*. \tag{11}$$

We use 20 fixed samples and, like Rousseau et al. [32], we solve for both u and $-u$, i.e. we solve 40 LPs. Then, in order to obtain stabilized dual variable values, we average over the dual information from these 40 LPs.

Other stabilization schemes have been proposed in the literature [see, e.g., 17, 27]. However, in order to avoid parameter tuning issues, we employ the simple scheme described above.

4.2.6 Populating the column pool

In order to start with a column pool Ω' that already contains columns that may be part of a good solution, we propose an adaptive large neighborhood search (ALNS) algorithm. It is described in detail in Section 5. During the execution of the ALNS we collect all feasible routes. After termination, all these routes are transformed into columns and put into the set Ω'.

Furthermore, we also use a second means to populate Ω': we choose a seed customer and we identify all customers that are within a radius of $0.5c_{max}$ (with $c_{max} = \max_{(i,j)\in A} c_{ij}$). Then we set up a reduced size instance containing at most 15 nodes (including the start and the end depot). In the case where too many customers are within the chosen radius, we randomly remove customers. We solve the thus obtained restricted TTRPTW by means of truncated column generation: we use ALNS to populate the column pool and the labeling algorithm without any further enhancements. We stop if the objective value was not improved in five consecutive column generation iterations and we repeat for 30 randomly chosen seed customers. All columns generated in this way are appended to the column pool of the original instance. Considering at most 15 nodes in the restricted TTRPTW instances appears to offer a good compromise between the time required to solve the problem and the potential of the newly generated columns to be useful in the column pool of the original instance.

4.3 Branching

In several cases the solution obtained from solving the relaxed master problem is already integer. Whenever this is not the case, we resort to branching and we use the above described column generation scheme to solve the linear relaxation in each branch-and-bound node. It is widely known that applying binary branching directly on the x_r variables is impractical. Therefore, we first branch on the number of trucks. Let k denote the number of trucks currently in use, and let us assume that this number is fractional, then we obtain two child nodes, one in which the upper bound of constraint (3) is set to $\lfloor k \rfloor$ and one in which the lower bound of constraint (3) is set to $\lceil k \rceil$.

In the case where an integer number of trucks is used, we branch on arcs. In order to do so, in addition to which customer is served on a given route, we also store which arcs are used in a given route or column. We use parameter a_{ijr} which takes value 1 if arc (i, j) appears on route r, and 0 otherwise. Using this information as input and summing over all columns in the current basis, we can compute whether or not an arc is used a fractional number of times. Since each arc can only appear at most once in a solution to the TTRPTW, we obtain values between 0 and 1 for each arc. In the case where one or more of these values is not integer-valued, we branch on the arc with the most fractional value (the one that is closest to 0.5). Let us denote this arc by (i, j). We then obtain two child nodes, one in which arc (i, j) is forbidden and one in which arc (i, j) is enforced. In the first node, in the general case, we simply remove the respective arc from the network and (since we keep a global column pool) we deactivate all columns that contain this arc (as well as all other arcs that have been forbidden in previous branching decisions leading to the current node). In the second node, we remove all arcs $(l, j)|l \in N \cup \{0\}, l \neq i$ and $(i, l)|l \in N \cup \{n + 1\}, l \neq j$ from the network and we deactivate all columns that contain any of the removed arcs.

We note that the above network modification rules can only be applied in the case where both i and j are truck customers. In the case where i is a truck customer and j a trailer

customer, forbidding arc (i, j) is straightforward as before. However, enforcing arc (i, j) is not so obvious because in addition to arc (i, j) there may be other arcs leading to j in a feasible solution to the TTRPTW. In order to account for this, we only remove all arcs leaving i (except the one leading to j) from the network but we do not remove arcs leading to j. In the case where i is a trailer customer and j is a truck customer, we do the reverse. Whenever both i and j are trailer customers, arc removal is not problematic but the network modifications to enforce the usage of an arc cannot be employed. Therefore, whenever i or j (or both) are trailer customers, in addition to all feasible network modifications, we also add the following new constraint to the master program which makes sure that arc (i, j) is used:

$$\sum_{r \in \Omega'} a_{ijr} x_r = 1. \tag{12}$$

The dual information associated with constraints of this type has to be passed to the subproblem. This is done by considering a reduced cost matrix in the label extension rules for resource *redcost*, which is initially equal to the original cost matrix. As soon as branching constraints of the above form are present, we subtract their dual variable values from the respective arc costs.

In order to select the next node to branch on we use best bound search, i.e. the node with the lowest lower bound is processed next.

4.4 Upper and lower bounds

In branch-and-bound algorithms the notion of bounds is used to prune the tree: whenever the obtained lower bound at a given node exceeds the current best upper bound, the corresponding node can be pruned. There exists a trade-off between the quality of the lower bound and the time needed to compute this bound. In our exact labeling algorithm, in order to save time, we do not solve the pricing subproblem to optimality but we stop as soon as at least 10 negative reduced cost columns have been found. However, whenever fewer than 10 labels of reduced cost are generated at the end depot, we actually solve the subproblem to optimality. This means that we can exploit the information provided by the label associated with the lowest reduced cost path: adding its reduced cost to the objective value of the current master problem provides a valid lower bound [25, 38]. Thus, in order to save time, when the solution to the current master problem is not integer but the obtained lower bound is higher than the parent lower bound, we use it, i.e. we stop pricing and resort to branching.

During initial experiments we sometimes encountered the problem that we were not able to generate a valid lower bound at the root node within the maximum time limit. In order to circumvent this issue we exploit again the observation that in some cases the optimal solutions of the TTRPTW and the VRPTW are rather similar or even coincide. At the root node, if more than 20% of the total allotted run time has already passed, we switch to a pricing strategy that solves a relaxation of the TTRPTW. This relaxation corresponds to solving the linear relaxation of a VRPTW: all customers are considered as truck customers and the capacity of the truck is set to $Q_{truck} + Q_{trailer}$. We use the standard pricing scheme to solve this relaxation and we stop the labeling algorithm as soon as at least 100 columns of negative reduced cost have been generated. In many cases, this bound will be weaker than the TTRPTW bound. However, arc branching may help to make the subproblems at the child nodes faster to solve. In the case where the root node solution corresponds to such a VRPTW solution and arc based branching should be performed, we compute the average arc usage of the solution corresponding to the VRPTW bound and the solution to the master problem before the switch to VRPTW pricing was performed. In the case where this average value is not fractional, we randomly choose an arc from the solution to branch on.

In order to start with a good upper bound which can help to prune bad branches early during the search, we use the upper bound generated by means of the truncated ALNS used to populate the column pool. It is described next.

5 Adaptive large neighborhood search

Adaptive large neighborhood search (ALNS) [31] is among the best performing metaheuristic frameworks for a wide range of routing problems [see, e.g. 9, 19, 21]. For this reason and because of its rather simple and easy to implement structure, we also use it to solve the TTRPTW. In ALNS, in each iteration, the current incumbent solution s is first destroyed (i.e. customers are removed from their routes) and then repaired (i.e. customers are inserted into the routes). In the case where the new solution s' is better than the incumbent solution or it meets other acceptance criteria, it replaces s. In the case where it is better than the best solution s_{best} encountered so far, it replaces also s_{best}. The first incumbent solution s_{init} is generated by means of a greedy insertion algorithm. Then, in each iteration, like in Kovacs et al. [22], a destroy-repair operator pair is chosen. The probability of an operator being selected depends on the operator's performance during past iterations.

5.1 Adaptive mechanism

We adopt the adaptive approach of Kovacs et al. [22] which is based on the one proposed by Ropke and Pisinger [31]. It works as follows: destroy-repair operator pairs collect performance scores during segments of 100 iterations. At the end of a segment, their weights are updated according to the collected scores. In our implementation, like in [22, 31], if an operator pair generates a new best solution, its score is increased by 33 and it is increased by 9 if the current incumbent solution is improved and the respective solution was not visited before. All initial weights are set to 1 and weight updates are done as follows: Let w_{dr} denote the weight of operator pair dr, s_{dr} the score obtained during the last segment, and S_{dr} the total number of times operator pair dr was used during the last segment, then w_{dr} is set equal to $0.9w_{dr} + 0.1s_{dr}/S_{dr}$. In contrast to Kovacs et al. [22] and also Ropke and Pisinger [31], the generation of worse solutions that are accepted as new incumbent solutions is not rewarded.

5.2 Acceptance scheme

Non-improving solutions are accepted using a simulated annealing type acceptance criterion, i.e. the probability of accepting a deteriorating solution is given by $e^{-(f(s')-f(s)/t}$, where t is the current temperature which is initially set to $t = p/\ln(0.5)f(s_{init})$. The temperature t is decreased in each iteration as follows: $t = 0.99975t$ and $p = 0.005$. Furthermore, if no new best solution is identified for 5000 iterations, the new solution becomes the new incumbent and the simulated annealing temperature is reset to $t = p/\ln(0.5)f(s)$.

5.3 Destroy operators

With respect to destroy operators, we use a random removal, a related removal, a worst removal and a cluster removal operator. All of them have also been employed by Kovacs et al. [22]. In the related removal operator, relatedness of two customers i and j is measured by $10|B_i - B_j| + 3c_{ij} + 4|q_i - q_j|$, where B_i denotes the beginning of service at customer i. In the cluster removal operator, only customers on the main route are considered in the generation of clusters. In all other operators, all currently inserted customers are considered. Whenever a customer is removed that also serves as the depot of one or several subroutes, all customers of these subroutes are removed as well. Whenever a route does not contain any subroutes and the total demand of a route is lower than or equal to the truck capacity, a previously present trailer is removed from the route. The number of customers to be removed in each iteration is randomly chosen between 5% and 50% of all currently inserted customers. The related and the worst removal operators are randomized like in Kovacs et al. [22] and Ropke and Pisinger [31]. Given a list L of customers sorted according to the chosen criterion, the customer with rank $|L|y^p$

is removed, where y is a random number in the $[0, 1)$ interval. In the worst removal operator, $p = 3$ while in the related removal operator, $p = 6$.

5.4 Repair operators

In terms of repair operators, we use a greedy insertion and four q-regret insertion heuristics (with $q \in \{2, 3, 4, m\}$). All operators are used in a deterministic as well as in a randomized fashion. In the randomized version we add a noise term to the insertion costs. This noise term is randomly chosen in $[-0.025c_{max}, 0.025c_{max}]$, where c_{max} denotes the maximum distance related cost between any pair of nodes. The choice to use noise or not in the current iteration is controlled in the same way as the selection of the operator pairs.

Whenever a trailer is already present on a route, only trailer customers can be inserted into the main route and we try to insert both types of customers into subroutes. However, the generation of a new subroute is only tried in the case of truck customers. If the current route is a truck route and no truck customers are currently present, the transformation into a trailer route is attempted if the capacity constraint cannot be met otherwise. Whenever we try to generate a new subroute at a trailer customer and it would become its first subroute, in a first step, we try to service the trailer customer before the new subroute. If this is not feasible, we try to move the visit to the trailer customer to after the subroute. Whenever we try to insert a customer into an existing subroute, no feasible scheduling is possible and the depot customer of this subroute is currently served directly before this subroute, we check if by moving the visit to the depot customer to after the subroute feasibility can be attained. Only if this fails as well, do we consider that no feasible insertion position into the current subroute exists. Note that if the decoupling point is not visited directly before the current subroute, we do not attempt to move its visit to after the current subroute.

6 Computational results

The proposed algorithms have been implemented in C++ and CPLEX 12.51 was used as linear programming solver. All tests were carried out on 2.67 GHz Intel Xeon CPUs. RAM was limited to 9G and only a single thread was used. In the following, we first describe the test instances. Thereafter we compare the performance of the proposed ALNS to the current state of the art and we evaluate the impact of the proposed enhancements on the performance of our branch-and-price algorithm.

6.1 Test instances

In order to test our algorithms we use three data sets with different characteristics and with different problem sizes. They are all derived from the VRPTW data set of Solomon [36]. Therefore, they all follow a similar scheme but they differ in terms of the total number of customers and in which and how many are considered as truck customers. The first data set was kindly provided by Vincent Yu and contains 50 and 100 customer instances derived from the C101, C201, R101, R201, C101, and C201 instances of Solomon. The second data set was kindly provided by Ulrich Derigs and contains 100-customer instances that are based on the same instances. The respective best known results are reported in Lin et al. [24] and Derigs et al. [10]. In order to evaluate the performance of our branch-and-price algorithm on instances with various characteristics, we generated an additional data set with 25-customer instances, taking each of the Solomon instances as a basis. In order to determine the truck customers, we employed the same scheme as Lin et al. [24]: for each customer we identify the nearest neighboring customer and we sort them according to the distance from their nearest neighbor, in increasing order. Then for each 25-customer Solomon instance we generated three

new instances, in which the first 25%, 50%, and 75% from the list are considered as truck customers. The remaining customers are considered as trailer customers.

6.2 Evaluating the performance of the proposed ALNS

In order to determine the performance of the proposed ALNS, we use the 100-customer instances of Lin et al. [24] and Derigs et al. [10] and compare our average and best results out of five random runs for 50,000 iterations to the results reported in the respective papers. We note here that in contrast to Lin et al. [24], we consider that trailer customers serving as decoupling points may be visited before or after each of their subroutes, which leads to additional flexibility in the planning. This flexibility is also considered by Derigs et al. [10]. In Table 1 we compare our results to the results of Lin et al. [24]. We report the following information: the best known solution values (BKS), the average results reported by Lin et al. [24], and their deviations from the BKS (dev), the average computation time reported by Lin et al. [24] (time), our average (Avg) and best results (Best) as well as their deviations from the BKS produced by the proposed ALNS and the corresponding information for the proposed ALNS followed by the solution of a set partitioning model on the set of all feasible routes generated during the execution of the ALNS. Finally, we also report the best objective values we encountered during all parameter tuning tests and their respective deviations from the best known results. In Table 2 we report similar information for the instances of Derigs et al. [10]. However, since solving the set partitioning problem on all generated routes did not lead to a considerable improvement in terms of solution quality, we do not report such results for the data set of Derigs et al. [10]. Both tables show that our method is very fast and it produces better results than the other two methods on average. Therefore, our ALNS appears to be a good choice for generating initial columns and for producing an initial upper bound.

6.3 Evaluating the performance of the branch-and-price algorithm

In order to evaluate the performance of the proposed branch-and-price algorithm we first test different parameter combinations on the newly generated data set with 25 customers, using a run time limit of 2 hours. Table 3 provides an overview of the different abbreviations used to indicate which parameter setting is currently tested.

Pricing schemes	s	standard exact pricing
	l	leveled exact pricing
	h	heuristic pricing
	L	leveled heuristic pricing
Enhancements	G	ng-route relaxation
	A	initial columns and bound by ALNS (10K iterations)
	M	initial columns by solving instances of reduced size
	S	stabilization (only at root node)
	V	VRPTW pricing (only at root node)

Table 3: Abbreviations used in tables

In Table 4 we compare the different pricing schemes. We report for how many instances (out of the total number of instances per class) valid lower bounds can be generated at the root node within a run time limit of 2 hours. These results indicate that the leveled pricing scheme (l) works better than the standard scheme (s) and it also suggests that a combination of leveled heuristic pricing and leveled exact pricing (Ll) performs better than non-leveled heuristic pricing in combination with standard pricing (hs). The best values per row are bold-faced unless all settings solve the same number of instances. In order to get a clearer picture of whether

		Lin et al.			ALNS (50K iterations)					ALNS plus MIP (50K iterations)						
	BKS	Avg (of 5)	dev (%)	time** (m)	Avg (of 5)	dev (%)	Best (of 5)	dev (%)	avg t* (m)	Avg (of 5)	dev (%)	Best (of 5)	dev (%)	avg t* (m)	Best seen	dev (%)
C101-25	971.82	971.92	0.01	52.71	905.71	-6.80	905.02	-6.87	1.30	905.02	-6.87	905.02	-6.87	1.31	903.84	-6.99
C101-50	1106.90	1111.72	0.44	62.00	1023.94	-7.49	1019.46	-7.90	1.25	1010.39	-8.72	1004.39	-9.26	1.45	1004.39	-9.26
C101-75	1154.80	1166.35	1.00	58.19	1107.04	-4.14	1081.25	-6.37	1.02	1118.92	-3.11	1068.92	-7.44	0.98	1058.85	-8.31
C201-25	671.31	685.14	2.06	74.70	676.95	0.84	671.37	0.01	1.65	674.70	0.51	671.37	0.01	1.61	671.37	0.01
C201-50	713.06	723.68	1.49	47.72	709.07	-0.56	709.07	-0.56	1.46	709.90	-0.44	709.07	-0.56	1.69	709.07	-0.56
C201-75	711.45	711.45	0.00	44.61	711.46	0.00	711.46	0.00	1.45	712.80	0.19	711.46	0.00	1.61	711.46	0.00
R101-25	1651.16	1659.96	0.53	69.89	1645.04	-0.37	1644.64	-0.39	1.01	1644.64	-0.39	1644.64	-0.39	1.18	1644.64	-0.39
R101-50	1644.64	1660.73	0.98	60.13	1646.06	0.09	1644.64	0.00	1.02	1644.64	0.00	1644.64	0.00	1.12	1644.64	0.00
R101-75	1644.64	1645.68	0.06	49.29	1645.10	0.03	1644.64	0.00	1.02	1644.64	0.00	1644.64	0.00	1.14	1644.64	0.00
R201-25	1166.64	1193.16	2.27	56.52	1160.00	-0.57	1154.21	-1.07	1.34	1157.67	-0.77	1152.25	-1.23	1.38	1147.80	-1.61
R201-50	1148.17	1172.90	2.15	53.65	1155.57	0.64	1149.17	0.09	1.24	1155.96	0.68	1149.79	0.14	1.37	1147.80	-0.03
R201-75	1147.80	1157.80	0.87	43.78	1150.74	0.26	1148.48	0.06	1.25	1148.95	0.10	1147.88	0.01	1.34	1147.80	0.00
RC101-25	1717.01	1739.52	1.31	70.86	1697.26	-1.15	1688.41	-1.67	1.25	1657.48	-3.47	1654.39	-3.65	1.36	1654.39	-3.65
RC101-50	1770.40	1793.11	1.28	62.75	1769.61	-0.04	1760.35	-0.57	1.09	1730.44	-2.26	1724.23	-2.61	1.40	1710.32	-3.39
RC101-75	1784.06	1794.70	0.60	50.12	1793.33	0.52	1789.89	0.33	0.93	1782.38	-0.09	1779.61	-0.25	1.31	1779.45	-0.26
RC201-25	1277.05	1287.16	0.79	60.95	1276.75	-0.02	1266.11	-0.86	1.25	1267.78	-0.73	1265.56	-0.90	1.37	1265.56	-0.90
RC201-50	1273.69	1289.75	1.26	45.84	1269.07	-0.36	1266.11	-0.60	1.28	1273.71	0.00	1266.11	-0.60	1.37	1265.56	-0.64
RC201-75	1266.11	1276.67	0.83	38.34	1268.93	0.22	1266.11	0.00	1.30	1272.31	0.49	1266.11	0.00	1.37	1265.98	-0.01
Average	1267.82	1280.08	1.00	55.67	1256.20	-1.05	1251.13	-1.46	1.23	1250.68	-1.38	1245.00	-1.87	1.35	1243.20	-2.00
Max			2.27			0.84		0.33			0.68		0.14			0.01

** CPU time on Intel Pentium 4 1.5 GHz computer

* CPU time on 2.67 GHz Intel Xeon computer

Table 1: Comparison to results of Lin et al. [24]

		Derigs et al. (100K)			Derigs et al. (220K)			ALNS (50K iterations)				
	BKS	Avg (of 5)	dev (%)	time (m)	Best (of 5)	dev (%)	avg t (m)	Avg (of 5)	dev (%)	Best (of 5)	dev (%)	avg t* (m)
C101-25-D	905.35	908.71	0.37	34.76	905.35	0.00	65.79	908.12	0.31	896.69	-0.96	6.53
C101-50-D	1009.96	1019.02	0.90	43.34	1009.96	0.00	72.56	987.54	-2.22	983.23	-2.65	5.66
C101-75-D	1081.61	1102.86	1.96	40.09	1081.61	0.00	69.81	1040.03	-3.84	1031.36	-4.65	5.52
C201-25-D	683.88	687.98	0.60	26.61	683.88	0.00	41.41	678.10	-0.85	671.37	-1.83	6.42
C201-50-D	700.63	711.46	1.55	18.03	700.63	0.00	38.84	698.30	-0.33	695.78	-0.69	6.51
C201-75-D	711.46	711.46	0.00	18.14	711.46	0.00	39.35	710.93	-0.08	710.06	-0.20	6.20
R101-25-D	1647.15	1648.77	0.10	30.39	1647.15	0.00	57.78	1645.30	-0.11	1645.30	-0.11	4.63
R101-50-D	1646.00	1649.38	0.21	35.01	1646.00	0.00	67.02	1645.17	-0.05	1644.64	-0.08	4.42
R101-75-D	1645.79	1647.62	0.11	36.84	1645.79	0.00	78.68	1644.90	-0.05	1644.64	-0.07	4.59
R201-25-D	1152.25	1161.92	0.84	16.57	1152.25	0.00	33.14	1162.73	0.91	1154.72	0.21	5.40
R201-50-D	1147.81	1152.22	0.38	14.01	1147.81	0.00	31.02	1150.62	0.24	1147.88	0.01	5.36
R201-75-D	1148.88	1154.32	0.47	19.90	1148.88	0.00	36.80	1150.93	0.18	1148.17	-0.06	5.53
RC101-25-D	1702.09	1710.61	0.50	33.23	1702.09	0.00	65.46	1685.23	-0.99	1680.02	-1.30	4.86
RC101-50-D	1739.10	1747.62	0.49	40.53	1739.10	0.00	78.06	1727.39	-0.67	1712.72	-1.52	4.44
RC101-75-D	1741.93	1748.60	0.38	39.85	1741.93	0.00	76.70	1742.18	0.01	1721.06	-1.20	4.91
RC201-25-D	1265.56	1275.48	0.78	14.50	1265.56	0.00	26.00	1268.95	0.27	1266.11	0.04	5.65
RC201-50-D	1266.11	1282.31	1.28	13.55	1266.11	0.00	31.10	1273.30	0.57	1266.11	0.00	5.63
RC201-75-D	1267.27	1282.69	1.22	15.27	1267.27	0.00	35.54	1269.49	0.18	1265.56	-0.13	5.47
Avg	1247.94	1255.72	0.67	27.26	1247.94	0.00	52.50	1243.84	-0.36	1238.08	-0.84	5.43
Max			1.96			0.00			0.91		0.21	

* original CPU time multiplied by factor 3.839 to make it comparable to CPU time of Derigs et al. [7]

Table 2: Comparison to results of Derigs et al. [10]

Data set	s	l	hs	Ll
C1	23/27	23/27	23/27	**24**/27
C2	19/24	**20**/24	19/24	19/24
R1	36/36	36/36	36/36	36/36
R2	25/33	**27**/33	25/33	25/33
RC1	24/24	24/24	24/24	24/24
RC2	15/24	**17**/24	16/24	**17**/24

Table 4: Comparison of pricing strategies for solving root node relaxation in terms of how many valid lower bounds can be computed within the run time limit ($n = 25$)

Data set	l-A	l-M	l-S	l-G	l-V	Ll-A	Ll-M	Ll-S	Ll-G	Ll-V
C1	**25**/27	24/27	24/27	25/27	27/27	24/27	**25**/27	24/27	**26**/27	27/27
C2	19/24	19/24	20/24	19/24	24/24	**20**/24	**20**/24	20/24	19/24	24/24
R1	36/36	36/36	36/36	36/36	36/36	36/36	36/36	36/36	36/36	36/36
R2	26/33	25/33	26/33	30/33	33/33	26/33	**26**/33	26/33	30/33	33/33
RC1	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24
RC2	18/24	17/24	**13**/24	18/24	**23**/24	**19**/24	17/24	11/24	**19**/24	22/24

Table 5: Comparison of pricing strategies plus enhancements for solving root node relaxation in terms of how many valid lower bounds can be computed within the run time limit ($n = 25$)

setting l or Ll should be used in combination with the different enhancements, we compute the root node lower bound for each of the two settings in turn in combination with initial columns produced by ALNS (A), initial columns produced by solving subproblems (M), stabilization (S), the *ng*-route relaxation (G), and switching to VRPTW bound generation after 20% of the maximum run time (V). The obtained results are reported in Table 5. Performing pairwise comparisons between l and Ll in combination with each of the proposed enhancements, we use bold face to denote better values. Doing so shows that, in general, Ll leads to more valid bounds than l. Evaluating each of the enhancements, they all appear to be beneficial for at least one instance class. However, they sometimes also reduce the number of valid lower bounds obtained in the root node by one or two for some data sets. Stabilization (S) even leads to a considerable reduction for instance class RC2: six fewer instances can be solved with Ll-S than with Ll. The reason is that for this instance class with stabilized dual prices proving that no additional column of negative reduced cost exists becomes cumbersome for the labeling algorithm.

Given this picture, we use Ll in combination with all of the different enhancements and with all but stabilization (S) in our final set of experiments with the branch-and-price algorithm. We use again a run time limit of 2 hours and we apply the algorithm to the newly generated instances with $n = 25$ customers, the instances of Lin et al. [24] with $n = 50$ and $n = 100$ customers, and the instances of Derigs et al. [10] with $n = 100$ customers. In Tables 6–11, we report the results for the 25-customer instances, in Tables 12 and 13 the results for the instances of Lin et al. [24] and in Table 14 for the instances of Derigs et al. [10]. For each of the two parameter settings Ll-A-M-S-G-V and Ll-A-M-G-V we report the following information for each instance: the upper bound obtained by means of ALNS, the lower bound at the root node (RLB), the final lower bound (LB), the final upper bound (UB), the percentage ratio of the lower bound with respect to the upper bound (a value of 100 indicates that the instance was solved to optimality), the total number of columns generated (col), the total number of nodes generated (nod), and the total run time in seconds. Even if the advantage of using stabilization was not obvious in our first set of experiments, when using all the different components together, it has a positive impact. The total number of instances solved to optimality increases from 21 to 22 for instance class C1 with 25 customers (see Table 6) and it also increases by one for instance class C2 with 25 customers (see Table 7), where we solve 11 with stabilization and only 10 without. Taking a

	Ll-A-M-S-G-V								Ll-A-M-G-V							
	ALNS	RLB	LB	UB	$\frac{LB}{UB}$%	col	nod	time	ALNS	RLB	LB	UB	$\frac{LB}{UB}$%	col	nod	time
c101-25	205.21	205.21	205.21	205.21	100.0	1872	1	3	205.21	205.21	205.21	205.21	100.0	1957	1	3
c101-50	222.97	216.37	219.58	219.58	100.0	3114	3	4	222.97	214.59	219.58	219.58	100.0	3102	3	3
c101-75	235.18	223.68	235.18	235.18	100.0	4261	39	21	235.18	221.68	235.18	235.18	100.0	4125	41	20
c102-25	203.68	203.68	203.68	203.68	100.0	3331	1	177	203.68	190.74	203.68	203.68	100.0	5881	3	2364
c102-50	218.34	211.99	218.34	218.34	100.0	4458	3	594	218.34	213.79	218.34	218.34	100.0	4585	3	949
c102-75	235.18	222.22	228.16	235.18	97.0	64599	147	7200	235.18	220.98	227.81	235.18	96.9	77036	57	7200
c103-25	203.68	203.68	203.68	203.68	100.0	4087	1	3266	203.68	190.74	190.74	203.68	93.6	6272	3	7200
c103-50	215.88	212.10	215.88	215.88	100.0	5252	3	7014	215.88	212.08	215.88	215.88	100.0	5362	3	4438
c103-75	235.18	222.10	222.10	235.18	94.4	6679	2	7200	235.18	220.64	220.64	235.18	93.8	6239	2	7200
c104-25	200.39	187.45	187.45	200.39	93.5	9540	2	7200	200.39	187.45	187.45	200.39	93.5	9534	2	7200
c104-50	213.11	187.45	187.45	212.89	88.0	11272	2	7201	213.11	187.45	209.37	212.89	98.3	8921	4	7200
c104-75	232.71	219.39	219.39	232.71	94.3	5458	2	7200	232.71	219.39	219.39	232.71	94.3	5409	2	7200
c105-25	204.92	204.92	204.92	204.92	100.0	2638	1	4	204.92	204.92	204.92	204.92	100.0	2769	1	8
c105-50	226.20	215.74	218.95	218.95	100.0	3276	5	11	226.20	215.74	218.95	218.95	100.0	3232	5	8
c105-75	235.18	223.68	235.18	235.18	100.0	4837	59	33	235.18	218.99	235.18	235.18	100.0	4754	51	37
c106-25	205.21	205.21	205.21	205.21	100.0	2304	1	3	205.21	205.21	205.21	205.21	100.0	2479	1	4
c106-50	222.97	216.37	219.58	219.58	100.0	3280	3	5	222.97	210.54	219.58	219.58	100.0	3266	3	4
c106-75	235.18	223.68	235.18	235.18	100.0	4101	31	25	235.18	222.25	235.18	235.18	100.0	4071	31	20
c107-25	204.92	204.92	204.92	204.92	100.0	2965	1	5	204.92	204.92	204.92	204.92	100.0	3113	1	14
c107-50	208.94	208.94	208.94	208.94	100.0	3634	1	4	208.94	208.94	208.94	208.94	100.0	3707	1	5
c107-75	227.08	216.99	227.08	227.08	100.0	5097	9	166	227.08	213.70	227.08	227.08	100.0	5302	9	154
c108-25	204.92	204.92	204.92	204.92	100.0	3331	1	29	204.92	204.92	204.92	204.92	100.0	3498	1	42
c108-50	208.94	208.94	208.94	208.94	100.0	4168	1	9	208.94	208.94	208.94	208.94	100.0	4322	1	22
c108-75	227.08	216.90	227.08	227.08	100.0	6330	47	688	227.08	210.63	227.08	227.08	100.0	6226	41	663
c109-25	221.14	204.33	204.33	204.33	100.0	4078	1	172	221.14	204.33	204.33	204.33	100.0	4333	1	1315
c109-50	227.08	208.33	208.33	208.33	100.0	5117	1	68	227.08	208.33	208.33	208.33	100.0	5290	1	125
c109-75	227.08	216.79	227.08	227.08	100.0	7462	81	2822	227.08	212.62	227.08	227.08	100.0	7115	61	2963
Solved								22								21

Table 6: Branch-and-price results for instance set C1 with $n = 25$

look at the instance set RC2 where in our initial experiments stabilization led to a performance decrease, we are able to solve the same number of instances to optimality with and without stabilization. However, it can be observed that, especially for this instance class, run times are generally higher with stabilization than without (see Table 11). Interestingly, in the case of instance set R2 with $n = 25$ (see Table 9), instance r204-75 can be solved with stabilization in place while it cannot be solved without. The reverse is true for instance r208-50: it cannot be solved with stabilization but it can be solved without. Hence, the same total number of instances is solved by both configurations but not the exact same instances.

For the larger instances of Lin et al. [24] and Derigs et al. [10] with 50 and 100 customers, we observe a very similar performance for both parameter settings. In Table 12, we report our results for the 18 50-customer instances of Lin et al. [24]. With both parameter settings, 13 of them can be solved to optimality. Two of the RC instances cannot be solved due to memory problems (indicated by "M" in column "time") and for the third RC instance (RC101-050_3) our maximum node limit of 20,000 nodes is reached (indicated by "N" in column "time").

In Table 13, we report our results for the 100-customer instances of Lin et al. [24]. For this data set we only solve 3 out of 18 instances to optimality (all 3 instances are derived from instance R101 of Solomon [36]) and almost no difference between using stabilization or not can be observed.

Finally, in Table 14, we present our results for the 100-customer instances of Derigs et al. [10]. For this data set, we solve 4 out of 18 instances to optimality with both parameter settings. In addition to those based on the R101 instance of Solomon [36], instance c101-25-D is solved. For this instance, setting Ll-A-M-G-V (no stabilization) is almost twice as fast as setting Ll-A-M-S-G-V (with stabilization).

Summarizing the above, we observe that taking all enhancements together, we are able to generate valid lower bounds for all instances. Since in some cases only the VRPTW-based lower bound can be solved in the 2-hour time limit, the gap between lower and upper bounds remains relatively large with about 17% for the largest instances with 100 customers and clustered customer locations.

	Ll-A-M-S-G-V								Ll-A-M-G-V							
	ALNS	RLB	LB	UB	$\frac{LB}{UB}$%	col	nod	time	ALNS	RLB	LB	UB	$\frac{LB}{UB}$%	col	nod	time
c201-25	227.29	220.63	223.07	223.07	100.0	4006	11	21	227.29	220.63	223.07	223.07	100.0	4217	13	25
c201-50	223.07	220.63	223.07	223.07	100.0	3752	11	10	223.07	220.63	223.07	223.07	100.0	2491	7	7
c201-75	227.29	220.63	227.29	227.29	100.0	4874	21	9	227.29	220.63	227.29	227.29	100.0	7454	21	17
c202-25	220.52	216.50	217.48	220.52	98.6	4890	4	7200	220.52	217.99	218.29	220.52	99.0	4597	4	7200
c202-50	220.52	218.24	218.29	220.52	99.0	4720	5	7200	220.52	218.24	218.29	220.52	99.0	4857	5	7200
c202-75	220.52	217.96	220.52	220.52	100.0	9687	17	920	220.52	218.24	220.40	220.52	99.9	85277	12	7200
c203-25	220.52	215.54	215.54	220.52	97.7	6509	2	7203	220.52	215.54	215.54	220.52	97.7	5399	2	7203
c203-50	220.52	218.17	218.24	220.52	99.0	7287	4	7201	220.52	215.54	217.46	220.52	98.6	5049	4	7200
c203-75	220.52	218.17	219.99	220.52	99.8	68191	15	7200	220.52	218.17	220.40	220.52	99.9	13197	18	7200
c204-25	220.32	212.73	212.73	220.32	96.6	7423	2	7200	220.32	212.73	212.73	220.32	96.6	7552	2	7200
c204-50	220.32	212.73	212.73	220.32	96.6	8382	2	7200	220.32	212.73	212.73	220.32	96.6	6776	2	7200
c204-75	220.32	212.73	212.73	220.32	96.6	6776	2	7200	220.32	212.73	212.73	220.32	96.6	6548	2	7200
c205-25	225.66	220.47	223.07	223.07	100.0	5412	33	296	225.66	220.47	223.07	223.07	100.0	7046	35	358
c205-50	225.66	220.47	223.07	223.07	100.0	7418	33	59	225.66	220.47	223.07	223.07	100.0	6312	33	59
c205-75	225.66	220.47	225.66	225.66	100.0	13358	137	66	225.66	220.47	225.66	225.66	100.0	12228	115	54
c206-25	225.66	220.47	223.07	223.07	100.0	9376	45	2329	225.66	220.47	223.07	223.07	100.0	9044	47	1955
c206-50	223.07	220.47	223.07	223.07	100.0	8534	35	377	223.07	220.47	223.07	223.07	100.0	10250	39	321
c206-75	225.66	220.47	225.66	225.66	100.0	23320	179	218	225.66	220.47	225.66	225.66	100.0	18148	151	154
c207-25	225.45	215.34	215.34	225.45	95.5	4316	3	7200	225.45	215.34	215.34	225.45	95.5	3984	2	7201
c207-50	225.45	219.34	220.40	225.45	97.8	5105	9	7200	225.45	219.12	220.43	225.45	97.8	6301	11	7200
c207-75	225.45	219.34	225.39	225.45	100.0	57391	252	7200	225.45	219.34	225.34	225.45	100.0	36505	233	7200
c208-25	225.49	220.22	220.43	225.49	97.8	22011	5	7200	225.49	220.22	220.43	225.49	97.8	70836	5	7200
c208-50	225.49	220.22	220.43	225.49	97.8	201551	5	7200	225.49	220.22	220.43	225.49	97.8	174866	5	7200
c208-75	225.49	220.22	225.49	225.49	100.0	28168	271	795	225.49	220.22	225.49	225.49	100.0	30009	259	822
Solved								11								10

Table 7: Branch-and-price results for instance set C2 with $n = 25$

7 Discussion and outlook

In this paper, we have proposed a branch-and-price algorithm for the truck and trailer routing problem with time windows. Initial columns are computed by means of an adaptive large neighborhood search algorithm tailored to the problem under study. It obtains highly competitive results when compared to two existing metaheuristic algorithms from the literature. Additional initial columns are generated by applying column generation to subinstances of the original instances. The pricing algorithm takes advantage of several problem-specific ideas and a heuristic pricing scheme using pre-computed subroutes is designed. Furthermore, the *ng*-route relaxation of Baldacci et al. [1] and the stabilization scheme of Rousseau et al. [32] are employed. The latter turns out to be most helpful for clustered instances but not so for random-and-clustered instances. We show how the truck and trailer routing problem with time windows can be relaxed to the vehicle routing problem with time windows and we exploit this observation in order to speed up lower bound computations at the root node. We solve most of the instances with 25 and 50 customers within a run time limit of two hours. The largest instances that can be solved have 100 customers. In most cases, instances with fewer truck customers take longer to solve than those with more truck customers. For clustered instances the construction of subroutes appears to be most beneficial and they seem to be more difficult to solve than random and random-and-clustered instances, especially with long planning horizons.

Additional enhancements to the proposed method could be envisaged such as replacing the rather simple VRPTW pricing scheme by more advanced bounding procedures as proposed by Baldacci et al. [1]. Furthermore, valid inequalities, such as subset-row inequalities [20] could be used to obtain tighter bounds. Parallelization is also a promising direction for future research

Acknowledgments We wish to thank Vincent Yu and Ulrich Derigs for having provided us with their data sets. The first author was supported by the Austrian Science Fund (FWF): T514-N13. This support is gratefully acknowledged.

	Ll-A-M-S-G-V								Ll-A-M-G-V							
	ALNS	RLB	LB	UB	$\frac{LB}{UB}$%	col	nod	time	ALNS	RLB	LB	UB	$\frac{LB}{UB}$%	col	nod	time
r101-25	618.33	618.33	618.33	618.33	100.0	1522	1	2	618.33	618.33	618.33	618.33	100.0	1522	1	2
r101-50	618.33	618.33	618.33	618.33	100.0	1361	1	2	618.33	618.33	618.33	618.33	100.0	1361	1	2
r101-75	618.33	618.33	618.33	618.33	100.0	1465	1	2	618.33	618.33	618.33	618.33	100.0	1466	1	2
r102-25	548.11	547.40	548.11	548.11	100.0	2436	3	2	548.11	547.40	548.11	548.11	100.0	2436	3	2
r102-50	548.11	547.40	548.11	548.11	100.0	2189	3	2	548.11	547.40	548.11	548.11	100.0	2189	3	2
r102-75	548.11	547.40	548.11	548.11	100.0	2154	3	2	548.11	547.40	548.11	548.11	100.0	2154	3	2
r103-25	455.70	455.70	455.70	455.70	100.0	3063	1	4	455.70	455.70	455.70	455.70	100.0	3099	1	4
r103-50	455.70	455.70	455.70	455.70	100.0	2982	1	4	455.70	455.70	455.70	455.70	100.0	2996	1	3
r103-75	455.70	455.70	455.70	455.70	100.0	2825	1	3	455.70	455.70	455.70	455.70	100.0	2837	1	2
r104-25	417.96	417.96	417.96	417.96	100.0	3575	1	13	417.96	417.96	417.96	417.96	100.0	3605	1	17
r104-50	417.96	417.96	417.96	417.96	100.0	3203	1	5	417.96	417.96	417.96	417.96	100.0	3239	1	5
r104-75	417.96	417.96	417.96	417.96	100.0	3238	1	4	417.96	417.96	417.96	417.96	100.0	3255	1	3
r105-25	531.54	531.54	531.54	531.54	100.0	2335	1	2	531.54	531.54	531.54	531.54	100.0	2335	1	2
r105-50	531.54	531.54	531.54	531.54	100.0	2175	1	2	531.54	531.54	531.54	531.54	100.0	2175	1	2
r105-75	531.54	531.54	531.54	531.54	100.0	2231	1	2	531.54	531.54	531.54	531.54	100.0	2231	1	2
r106-25	466.48	461.58	466.48	466.48	100.0	3142	3	7	466.48	461.58	466.48	466.48	100.0	3110	3	5
r106-50	466.48	461.58	466.48	466.48	100.0	3175	3	4	466.48	461.58	466.48	466.48	100.0	3175	3	4
r106-75	466.48	461.58	466.48	466.48	100.0	3082	3	4	466.48	461.58	466.48	466.48	100.0	3103	3	3
r107-25	429.20	427.52	429.20	429.20	100.0	3404	3	14	429.20	427.52	429.20	429.20	100.0	3399	3	12
r107-50	429.20	427.52	429.20	429.20	100.0	3359	3	9	429.20	427.52	429.20	429.20	100.0	3370	3	8
r107-75	429.20	427.52	429.20	429.20	100.0	3199	3	5	429.20	427.52	429.20	429.20	100.0	3201	3	3
r108-25	404.28	403.45	404.28	404.28	100.0	3974	11	2006	404.28	403.45	404.28	404.28	100.0	4000	11	2109
r108-50	404.28	403.45	404.28	404.28	100.0	3808	11	1207	404.28	403.45	404.28	404.28	100.0	3813	9	1312
r108-75	404.28	403.45	404.28	404.28	100.0	4014	11	1061	404.28	403.45	404.28	404.28	100.0	4001	11	1209
r109-25	442.62	442.62	442.62	442.62	100.0	2899	1	3	442.62	442.62	442.62	442.62	100.0	2916	1	2
r109-50	442.62	442.62	442.62	442.62	100.0	2721	1	2	442.62	442.62	442.62	442.62	100.0	2724	1	2
r109-75	442.62	442.62	442.62	442.62	100.0	2850	1	2	442.62	442.62	442.62	442.62	100.0	2854	1	2
r110-25	445.18	440.62	445.18	445.18	100.0	3784	13	12	445.18	440.62	445.18	445.18	100.0	3755	7	6
r110-50	447.47	439.98	445.18	445.18	100.0	3707	5	4	447.47	441.03	445.18	445.18	100.0	3743	9	4
r110-75	445.18	441.03	445.18	445.18	100.0	3453	5	3	445.18	441.03	445.18	445.18	100.0	3455	5	2
r111-25	429.70	428.81	429.70	429.70	100.0	3634	3	11	429.70	428.81	429.70	429.70	100.0	3630	3	8
r111-50	429.70	428.81	429.70	429.70	100.0	3617	3	6	429.70	428.81	429.70	429.70	100.0	3617	3	4
r111-75	429.70	428.81	429.70	429.70	100.0	3233	3	4	429.70	428.81	429.70	429.70	100.0	3240	3	3
r112-25	403.60	393.87	402.85	402.85	100.0	4563	29	621	403.60	393.87	402.85	402.85	100.0	4511	29	432
r112-50	403.60	394.15	402.85	402.85	100.0	4665	31	260	403.60	394.15	402.85	402.85	100.0	4729	33	221
r112-75	403.60	394.15	402.85	402.85	100.0	4211	33	171	403.60	394.15	402.85	402.85	100.0	4218	31	153
Solved								36								36

Table 8: Branch-and-price results for instance set R1 with $n = 25$

	Ll-A-M-S-G-V								Ll-A-M-G-V							
	ALNS	RLB	LB	UB	$\frac{LB}{UB}$%	col	nod	time	ALNS	RLB	LB	UB	$\frac{LB}{UB}$%	col	nod	time
r201-25	464.38	461.30	464.38	464.38	100.0	3086	3	8	464.38	461.30	464.38	464.38	100.0	3140	3	7
r201-50	464.38	461.30	464.38	464.38	100.0	3081	3	5	464.38	461.30	464.38	464.38	100.0	3087	3	4
r201-75	464.38	461.30	464.38	464.38	100.0	2656	3	3	464.38	461.30	464.38	464.38	100.0	2667	3	2
r202-25	411.49	411.49	411.49	411.49	100.0	3280	1	225	411.49	411.49	411.49	411.49	100.0	3487	1	134
r202-50	411.49	411.49	411.49	411.49	100.0	3345	1	47	411.49	411.49	411.49	411.49	100.0	3750	1	39
r202-75	411.49	411.49	411.49	411.49	100.0	3221	1	7	411.49	411.49	411.49	411.49	100.0	3506	1	9
r203-25	419.07	387.09	387.09	392.33	98.7	5167	3	7201	419.07	387.09	387.09	392.33	98.7	4907	4	7201
r203-50	406.24	387.09	392.33	392.33	100.0	4036	7	2609	406.24	387.09	392.33	392.33	100.0	4152	7	2779
r203-75	400.40	387.09	392.33	392.33	100.0	4167	7	488	400.40	387.09	392.33	392.33	100.0	4173	7	713
r204-25	371.36	350.30	350.30	371.36	94.3	3908	2	7203	371.36	350.30	350.30	371.36	94.3	3943	2	7204
r204-50	360.62	350.30	350.32	356.61	98.2	4587	5	7201	360.62	350.30	350.56	355.89	98.5	4284	4	7201
r204-75	371.36	350.30	355.89	355.89	100.0	7080	13	6136	371.36	350.30	354.82	355.89	99.7	6313	16	7200
r205-25	394.06	391.67	394.06	394.06	100.0	3581	3	175	394.06	391.67	394.06	394.06	100.0	3547	3	157
r205-50	401.50	391.67	394.06	394.06	100.0	3594	3	29	401.50	391.67	394.06	394.06	100.0	3701	3	26
r205-75	401.50	391.67	394.06	394.06	100.0	3280	3	8	401.50	391.67	394.06	394.06	100.0	3305	3	6
r206-25	375.48	374.62	375.06	375.48	99.9	3815	4	7201	375.48	374.62	375.06	375.48	99.9	3507	4	7201
r206-50	375.48	374.62	375.48	375.48	100.0	3455	5	2365	375.48	374.62	375.48	375.48	100.0	3577	5	2344
r206-75	378.18	374.62	375.48	375.48	100.0	3330	5	365	378.18	374.62	375.48	375.48	100.0	3461	5	278
r207-25	363.27	361.14	361.14	363.27	99.4	3928	2	7203	363.27	361.14	361.14	363.27	99.4	3466	2	7202
r207-50	363.27	361.14	361.14	362.64	99.6	4769	3	7201	363.27	361.14	361.14	362.64	99.6	4029	3	7200
r207-75	363.27	361.14	362.64	362.64	100.0	3692	3	4871	363.27	361.14	362.64	362.64	100.0	3950	3	2418
r208-25	329.33	328.60	328.60	329.33	99.8	13486	3	7203	329.33	328.60	328.89	329.33	99.9	16827	3	7205
r208-50	329.33	328.60	328.60	329.33	99.8	19863	3	7203	329.33	328.60	329.33	329.33	100.0	18792	3	3834
r208-75	329.33	328.60	329.33	329.33	100.0	18797	3	772	329.33	328.60	329.33	329.33	100.0	18797	3	670
r209-25	390.15	365.03	365.03	371.56	98.2	3407	3	7200	390.15	365.03	365.03	371.56	98.2	4027	3	7200
r209-50	373.08	365.03	365.03	371.56	98.2	3778	3	7200	373.08	365.03	368.94	373.08	98.9	11506	6	7200
r209-75	373.08	365.03	371.56	371.56	100.0	3975	9	74	373.08	365.03	371.56	371.56	100.0	4276	9	76
r210-25	410.60	405.12	405.48	405.48	100.0	4203	3	5023	410.60	405.12	405.48	405.48	100.0	4026	3	4633
r210-50	410.60	405.12	405.48	405.48	100.0	4190	3	768	410.60	405.12	405.48	405.48	100.0	4100	3	803
r210-75	410.60	405.12	405.48	405.48	100.0	3764	3	151	410.60	405.12	405.48	405.48	100.0	3736	3	217
r211-25	353.76	341.77	341.77	353.76	96.6	4871	2	7200	353.76	341.77	341.77	353.76	96.6	4830	2	7200
r211-50	352.80	341.77	341.77	352.80	96.9	5315	2	7200	352.80	341.77	341.77	352.80	96.9	4998	2	7200
r211-75	365.07	341.77	348.81	353.03	98.8	14086	28	7200	365.07	341.77	346.01	359.09	96.4	42123	15	7200
Solved								20								20

Table 9: Branch-and-price results for instance set R2 with $n = 25$

	Ll-A-M-S-G-V								Ll-A-M-G-V							
	ALNS	RLB	LB	UB	$\frac{LB}{UB}$%	col	nod	time	ALNS	RLB	LB	UB	$\frac{LB}{UB}$%	col	nod	time
rc101-25	466.53	412.17	465.92	465.92	100.0	9162	7315	362	466.53	412.11	465.92	465.92	100.0	9779	9741	523
rc101-50	481.92	419.30	471.11	471.11	100.0	7483	7769	342	481.92	419.72	471.11	471.11	100.0	7524	7765	342
rc101-75	535.14	431.46	487.76	487.76	100.0	4860	2001	54	535.14	434.08	487.76	487.76	100.0	5030	2223	62
rc102-25	367.99	363.86	363.86	363.86	100.0	3092	1	8	367.99	363.86	363.86	363.86	100.0	3120	1	4
rc102-50	377.87	369.68	369.68	369.68	100.0	3320	1	7	377.87	369.68	369.68	369.68	100.0	3345	1	4
rc102-75	524.74	395.21	391.38	391.38	100.0	2837	3	5	524.74	391.38	391.38	391.38	100.0	2830	1	2
rc103-25	347.76	346.51	346.51	346.51	100.0	3075	1	12	347.76	346.51	346.51	346.51	100.0	3186	1	13
rc103-50	355.54	354.12	354.12	354.12	100.0	3708	1	17	355.54	354.12	354.12	354.12	100.0	3751	1	8
rc103-75	427.46	373.28	373.28	373.28	100.0	4041	1	9	427.46	373.28	373.28	373.28	100.0	4063	1	4
rc104-25	326.12	320.61	320.61	320.61	100.0	3628	1	99	326.12	320.61	320.61	320.61	100.0	3848	1	71
rc104-50	350.34	342.02	342.02	342.02	100.0	4683	1	88	350.34	342.02	342.02	342.02	100.0	4722	1	34
rc104-75	373.28	361.07	361.07	361.07	100.0	3707	1	10	373.28	361.07	361.07	361.07	100.0	3743	1	6
rc105-25	412.56	412.56	412.56	412.56	100.0	2548	1	3	412.56	412.56	412.56	412.56	100.0	2565	1	3
rc105-50	472.63	419.72	419.72	419.72	100.0	2782	3	4	472.63	419.72	419.72	419.72	100.0	2780	3	3
rc105-75	528.43	434.35	434.35	434.35	100.0	2645	1	5	528.43	434.35	434.35	434.35	100.0	2656	1	2
rc106-25	402.93	355.32	355.32	355.32	100.0	2762	1	5	402.93	355.32	355.32	355.32	100.0	2801	1	3
rc106-50	366.48	361.25	361.25	361.25	100.0	3239	1	5	366.48	361.25	361.25	361.25	100.0	3244	1	3
rc106-75	415.72	388.71	388.71	388.71	100.0	3539	3	4	415.72	388.71	388.71	388.71	100.0	3545	3	3
rc107-25	319.44	318.45	318.45	318.45	100.0	3454	1	25	319.44	318.45	318.45	318.45	100.0	3563	1	23
rc107-50	334.21	333.23	333.23	333.23	100.0	3769	1	15	334.21	333.23	333.23	333.23	100.0	3793	1	8
rc107-75	397.68	345.55	345.55	345.55	100.0	4128	1	8	397.68	345.55	345.55	345.55	100.0	4135	1	5
rc108-25	364.18	314.64	314.64	314.64	100.0	4154	1	97	364.18	314.64	314.64	314.64	100.0	4378	1	243
rc108-50	380.79	331.24	331.24	331.24	100.0	4673	1	188	380.79	331.24	331.24	331.24	100.0	4712	1	64
rc108-75	349.39	345.55	345.55	345.55	100.0	4609	1	26	349.39	345.55	345.55	345.55	100.0	4618	1	16
Solved								24								24

Table 10: Branch-and-price results for instance set RC1 with $n = 25$

	Ll-A-M-S-G-V								Ll-A-M-G-V							
	ALNS	RLB	LB	UB	$\frac{LB}{UB}$%	col	nod	time	ALNS	RLB	LB	UB	$\frac{LB}{UB}$%	col	nod	time
rc201-25	361.24	361.24	361.24	361.24	100.0	2339	1	20	361.24	361.24	361.24	361.24	100.0	2566	1	4
rc201-50	361.24	361.24	361.24	361.24	100.0	2340	1	13	361.24	361.24	361.24	361.24	100.0	2548	1	4
rc201-75	361.24	361.24	361.24	361.24	100.0	2275	1	4	361.24	361.24	361.24	361.24	100.0	2478	1	2
rc202-25	338.82	338.82	338.82	338.82	100.0	3119	1	1448	338.82	338.82	338.82	338.82	100.0	3067	1	779
rc202-50	338.82	338.82	338.82	338.82	100.0	2857	1	1446	338.82	338.82	338.82	338.82	100.0	3025	1	485
rc202-75	338.82	338.82	338.82	338.82	100.0	2384	1	240	338.82	338.82	338.82	338.82	100.0	2869	1	18
rc203-25	357.05	327.69	327.69	327.69	100.0	3518	1	1498	357.05	327.69	327.69	327.69	100.0	4594	1	1852
rc203-50	327.69	327.69	327.69	327.69	100.0	3137	1	1494	327.69	327.69	327.69	327.69	100.0	3880	1	1496
rc203-75	357.05	327.69	327.69	327.69	100.0	4076	1	1455	357.05	327.69	327.69	327.69	100.0	4137	1	570
rc204-25	334.86	300.23	300.23	305.59	98.2	9361	2	7200	334.86	300.23	300.23	300.98	99.8	9603	2	7200
rc204-50	313.32	300.23	300.23	300.23	100.0	3750	1	1568	313.32	300.23	300.23	300.23	100.0	4409	1	1460
rc204-75	313.32	300.23	300.23	300.23	100.0	4514	1	1552	313.32	300.23	300.23	300.23	100.0	5794	1	1480
rc205-25	338.93	338.93	338.93	338.93	100.0	2392	1	1057	338.93	338.93	338.93	338.93	100.0	2932	1	102
rc205-50	338.93	338.93	338.93	338.93	100.0	2457	1	323	338.93	338.93	338.93	338.93	100.0	2831	1	29
rc205-75	338.93	338.93	338.93	338.93	100.0	2335	1	42	338.93	338.93	338.93	338.93	100.0	2758	1	4
rc206-25	325.10	325.10	325.10	325.10	100.0	2955	1	1445	325.10	325.10	325.10	325.10	100.0	2712	1	127
rc206-50	325.10	325.10	325.10	325.10	100.0	2217	1	526	325.10	325.10	325.10	325.10	100.0	2916	1	28
rc206-75	344.93	325.10	325.10	325.10	100.0	2151	1	158	344.93	325.10	325.10	325.10	100.0	2847	1	8
rc207-25	298.95	298.95	298.95	298.95	100.0	2481	1	1602	298.95	298.95	298.95	298.95	100.0	3196	1	1324
rc207-50	308.57	298.95	298.95	298.95	100.0	3381	1	1463	308.57	298.95	298.95	298.95	100.0	2625	1	462
rc207-75	311.93	298.95	298.95	298.95	100.0	2885	1	1448	311.93	298.95	298.95	298.95	100.0	2930	1	68
rc208-25	269.57	269.57	269.57	269.57	100.0	4740	1	2131	269.57	269.57	269.57	269.57	100.0	5899	1	1713
rc208-50	269.57	269.57	269.57	269.57	100.0	6131	1	2345	269.57	269.57	269.57	269.57	100.0	6186	1	1855
rc208-75	269.57	269.57	269.57	269.57	100.0	4110	1	2198	269.57	269.57	269.57	269.57	100.0	7568	1	1909
Solved								23								23

Table 11: Branch-and-price results for instance set RC2 with $n = 25$

	Ll-A-M-S-G-V								Ll-A-M-G-V							
	ALNS	RLB	LB	UB	$\frac{LB}{UB}$%	col	nod	time	ALNS	RLB	LB	UB	$\frac{LB}{UB}$%	col	nod	time
C101-050_1	396.02	396.02	396.02	396.02	100.0	3076	1	21	396.02	396.02	396.02	396.02	100.0	3267	1	54
C101-050_2	430.17	420.95	426.08	426.08	100.0	9054	3	6420	430.17	420.00	426.08	426.08*	100.0	8712	3	6590
C101-050_3	457.48	438.17	453.39	453.39	100.0	10774	101	1506	457.48	432.73	453.39	453.39*	100.0	11599	103	2069
C201-050_1	404.67	361.80	361.80	397.95	90.9	29223	3	7201	404.67	361.80	361.80	397.95	90.9	26196	3	7200
C201-050_2	412.47	397.90	401.57	404.67	99.2	10125	11	7200	412.47	395.27	401.26	404.67	99.2	11282	11	7200
C201-050_3	404.67	398.88	403.86	403.86	100.0	20074	21	3231	404.67	397.81	403.86	403.86	100.0	19690	19	3008
R101-050_1	1049.50	1046.70	1046.70	1046.70	100.0	2809	1	6	1049.50	1046.70	1046.70	1046.70	100.0	2814	1	5
R101-050_2	1046.70	1046.70	1046.70	1046.70	100.0	3121	1	6	1046.70	1046.70	1046.70	1046.70	100.0	3126	1	5
R101-050_3	1046.70	1046.70	1046.70	1046.70	100.0	2786	1	6	1046.70	1046.70	1046.70	1046.70*	100.0	2788	1	5
R201-050_1	804.53	794.34	794.34	794.34	100.0	4447	1	685	804.53	794.34	794.34	794.34	100.0	5221	1	914
R201-050_2	800.85	794.34	794.34	794.34	100.0	3655	1	152	800.85	794.34	794.34	794.34	100.0	4652	1	169
R201-050_3	802.60	794.34	794.34	794.34	100.0	3689	1	44	802.60	794.34	794.34	794.34	100.0	4187	1	59
RC101-050_1	975.66	858.59	909.60	975.66	93.2	17043	19519	M	975.66	857.79	909.33	975.66	93.2	17236	19158	M
RC101-050_2	986.00	864.92	915.99	986.00	92.9	14350	18376	M	986.00	866.22	916.31	986.00	92.9	16786	17980	M
RC101-050_3	996.13	907.19	965.67	979.31	98.6	13657	20001	N	996.13	905.96	965.16	996.13	96.9	13125	20001	N
RC201-050_1	686.31	686.31	686.31	686.31	100.0	4145	1	1453	686.31	686.31	686.31	686.31	100.0	4538	1	665
RC201-050_2	686.31	686.31	686.31	686.31	100.0	2253	1	796	686.31	686.31	686.31	686.31	100.0	4031	1	259
RC201-050_3	714.97	686.31	686.31	686.31	100.0	2268	1	263	714.97	686.31	686.31	686.31	100.0	3434	1	36
Solved								13								13

*These values are higher than the best ever seen values given in Lin et al. [24]. However, these authors kindly confirmed that these values were not correctly reported.

Table 12: Branch-and-price results for the instances of Lin et al. [24] with $n = 50$

	Ll-A-M-S-G-V								Ll-A-M-G-V							
	ALNS	RLB	LB	UB	$\frac{LB}{UB}$%	col	nod	time	ALNS	RLB	LB	UB	$\frac{LB}{UB}$%	col	nod	time
C101-100_1	922.04	891.40	891.40	922.04	96.7	20647	2	7200	922.04	892.05	892.05	922.04	96.7	18180	2	7200
C101-100_2	1020.29	967.68	969.01	1020.29	95.0	26679	4	7200	1020.29	969.33	969.33	1020.29	95.0	26025	2	7200
C101-100_3	1127.90	1026.06	1026.06	1127.90	91.0	36392	2	7200	1127.90	1017.98	1017.98	1127.90	90.3	43033	2	7200
C201-100_1	671.76	591.56	591.56	671.76	88.1	128099	2	7200	671.76	591.56	591.56	671.76	88.1	78566	2	7201
C201-100_2	709.07	591.56	591.56	709.07	83.4	69878	2	7202	709.07	591.56	591.56	709.07	83.4	116730	2	7200
C201-100_3	711.46	591.56	591.56	711.46	83.1	92904	2	7200	711.46	591.56	591.56	711.46	83.1	74272	2	7200
R101-100_1	1646.67	1639.46	1644.64	1644.64	100.0	6920	27	93	1646.67	1637.56	1644.64	1644.64	100.0	6977	31	99
R101-100_2	1647.60	1638.77	1644.64	1644.64	100.0	7750	33	62	1647.60	1635.54	1644.64	1644.64	100.0	7778	33	57
R101-100_3	1645.30	1640.23	1644.64	1644.64	100.0	7143	23	34	1645.30	1634.25	1644.64	1644.64	100.0	7100	25	34
R201-100_1	1158.43	1142.41	1142.41	1157.79	98.7	14369	2	7200	1158.43	1142.41	1142.41	1157.79	98.7	23450	2	7202
R201-100_2	1166.84	1142.41	1142.41	1150.68	99.3	20359	2	7202	1166.84	1142.41	1142.41	1166.84	97.9	21377	2	7203
R201-100_3	1162.15	1142.41	1142.41	1162.15	98.3	16895	2	7202	1162.15	1142.41	1142.41	1162.15	98.3	18835	2	7202
RC101-100_1	1695.85	1620.08	1632.04	1695.85	96.2	14438	286	7200	1695.85	1619.37	1631.96	1695.85	96.2	15044	335	7200
RC101-100_2	1783.45	1666.59	1686.63	1783.45	94.6	14022	487	7200	1783.45	1666.41	1687.29	1783.45	94.6	14053	521	7200
RC101-100_3	1811.71	1732.57	1758.61	1811.71	97.1	13714	3574	7200	1811.71	1728.79	1758.17	1811.71	97.0	13924	3688	7200
RC201-100_1	1298.04	1258.34	1258.34	1266.11	99.4	25750	2	7202	1298.04	1259.45	1259.45	1266.11	99.5	36123	2	7200
RC201-100_2	1281.90	1259.45	1259.45	1278.15	98.5	19168	2	7203	1281.90	1259.45	1259.45	1267.62	99.4	20308	2	7204
RC201-100_3	1278.08	1259.45	1259.45	1267.06	99.4	23191	2	7204	1278.08	1259.45	1259.45	1278.08	98.5	21914	2	7204
Solved								3								3

Table 13: Branch-and-price results for the instances of Lin et al. [24] with $n = 100$

	Ll-A-M-S-G-V								Ll-A-M-G-V							
	ALNS	RLB	LB	UB	$\frac{LB}{UB}$%	col	nod	time	ALNS	RLB	LB	UB	$\frac{LB}{UB}$%	col	nod	time
c101_25-D	896.69	891.26	893.78	893.78	100.0	6283	5	4108	896.69	890.75	893.78	893.78	100.0	6034	3	2310
c101_50-D	1035.70	947.52	947.52	1035.70	91.5	21713	2	7200	1035.70	947.27	947.27	1035.70	91.5	16756	2	7200
c101_75-D	1068.67	985.51	985.51	1068.67	92.2	23059	4	7200	1068.67	979.30	979.30	1068.67	91.6	22440	2	7200
c201_25-D	687.97	591.56	591.56	687.97	86.0	82421	2	7200	687.97	591.56	591.56	687.97	86.0	76513	2	7200
c201_50-D	693.33	591.56	591.56	693.33	85.3	69874	2	7201	693.33	591.56	591.56	693.33	85.3	71749	2	7201
c201_75-D	710.06	591.56	591.56	710.06	83.3	71292	2	7200	710.06	591.56	591.56	710.06	83.3	86394	2	7200
r101_25-D	1647.39	1639.57	1644.64	1644.64	100.0	6846	29	96	1647.39	1639.22	1644.64	1644.64	100.0	6850	29	100
r101_50-D	1646.67	1639.46	1644.64	1644.64	100.0	6920	27	90	1646.67	1637.56	1644.64	1644.64	100.0	6977	31	109
r101_75-D	1645.30	1638.39	1644.64	1644.64	100.0	6348	33	85	1645.30	1641.13	1644.64	1644.64	100.0	6336	31	88
r201_25-D	1152.55	1142.41	1142.41	1152.55	99.1	13872	2	7202	1152.55	1142.41	1142.41	1152.55	99.1	26275	2	7201
r201_50-D	1158.43	1142.41	1142.41	1157.79	98.7	14369	2	7200	1158.43	1142.41	1142.41	1157.79	98.7	22974	2	7202
r201_75-D	1172.53	1142.41	1142.41	1172.53	97.4	17620	2	7202	1172.53	1142.41	1142.41	1172.53	97.4	24848	2	7203
rc101_25-D	1695.81	1613.80	1623.90	1695.81	95.8	15700	267	7200	1695.81	1614.48	1627.22	1695.81	96.0	15987	287	7200
rc101_50-D	1747.70	1643.86	1658.86	1747.70	94.9	16397	429	7200	1747.70	1644.47	1658.82	1747.70	94.9	16320	410	7200
rc101_75-D	1747.47	1664.60	1683.21	1747.47	96.3	13692	456	7200	1747.47	1664.73	1683.07	1747.47	96.3	13738	420	7200
rc201_25-D	1266.11	1258.34	1258.34	1266.11	99.4	24141	2	7202	1266.11	1259.45	1259.45	1266.11	99.5	31974	2	7200
rc201_50-D	1268.07	1258.34	1258.34	1268.07	99.2	29707	2	7203	1268.07	1259.45	1259.45	1268.07	99.3	25544	2	7201
rc201_75-D	1267.88	1258.34	1258.34	1267.88	99.2	28638	2	7202	1267.88	1259.45	1259.45	1267.88	99.3	23533	2	7202
Solved								4								4

Table 14: Branch-and-price results for the instances of Derigs et al. [10] with $n = 100$

References

[1] Baldacci, R., Mingozzi, A., and Roberti, R. (2011). New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, 59(5):1269–1283.

[2] Baldacci, R., Mingozzi, A., Roberti, R., and Calvo, R. W. (2013). An exact algorithm for the two-echelon capacitated vehicle routing problem. *Operations Research*, 61(2):298–314.

[3] Belenguer, J. M., Benavent, E., Martínez, A., Prins, C., Prodhon, C., and Villegas, J. G. (2015). A branch-and-cut algorithm for the single truck and trailer routing problem with satellite depots. *Transportation Science*, in press, available online (`http://dx.doi.org/10.1287/trsc.2014.0571`).

[4] Caramia, M. and Guerriero, F. (2010). A heuristic approach for the truck and trailer routing problem. *Journal of the Operational Research Society*, 61:1168–1180.

[5] Chao, I.-M. (2002). A tabu search method for the truck and trailer routing problem. *Computers & Operations Research*, 29(1):33–51.

[6] Codato, G. and Fischetti, M. (2006). Combinatorial Benders' cuts for mixed-integer linear programming. *Operations Research*, 54(4):756–766.

[7] CPU-benchmark (2015). `https://www.cpubenchmark.net`. accessed: May 2015.

[8] Cuda, R., Guastaroba, G., and Speranza, M. (2015). A survey on two-echelon routing problems. *Computers & Operations Research*, 55:185–199.

[9] Demir, E., Bektaş, T., and Laporte, G. (2012). An adaptive large neighborhood search heuristic for the pollution-routing problem. *European Journal of Operational Research*, 223(2):346–359.

[10] Derigs, U., Pullmann, M., and Vogel, U. (2013). Truck and trailer routing—problems, heuristics and computational experience. *Computers & Operations Research*, 40(2):536–546.

[11] Desaulniers, G., Desrosiers, J., and Solomon, M. M., editors (2006). *Column Generation*, volume 5. Springer, New York.

[12] Desaulniers, G., Madsen, O. B., and Røpke, S. (2014). The vehicle routing problem with time windows. In Toth, P. and Vigo, D., editors, *Vehicle Routing: Problems, Methods, and Applications*, pages 119–159. SIAM, Philadelphia.

[13] Drexl, M. (2011). Branch-and-price and heuristic column generation for the generalized truck-and-trailer routing problem. *Revista de Metodos Cuantitativos para la economia y la empresa*, 12:5–38.

[14] Drexl, M. (2013). Applications of the vehicle routing problem with trailers and transshipments. *European Journal of Operational Research*, 227(2):275–283.

[15] Drexl, M. (2014). Branch-and-cut algorithms for the vehicle routing problem with trailers and transshipments. *Networks*, 63(1):119–133.

[16] Drexl, M. and Schneider, M. (2015). A survey of variants and extensions of the location-routing problem. *European Journal of Operational Research*, 241(2):283–308.

[17] Du Merle, O., Villeneuve, D., Desrosiers, J., and Hansen, P. (1999). Stabilized column generation. *Discrete Mathematics*, 194(1):229–237.

[18] Feillet, D., Dejax, P., Gendreau, M., and Gueguen, C. (2004). An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229.

[19] Hemmelmayr, V. C., Cordeau, J.-F., and Crainic, T. G. (2012). An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. *Computers & Operations Research*, 39(12):3215–3228.

[20] Jepsen, M., Petersen, B., Spoorendonk, S., and Pisinger, D. (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56(2):497–511.

[21] Kovacs, A. A., Golden, B. L., Hartl, R. F., and Parragh, S. N. (2015). The generalized consistent vehicle routing problem. *Transportation Science*, in press, available online (`http://dx.doi.org/10.1287/trsc.2014.0529`).

[22] Kovacs, A. A., Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2012). Adaptive large neighborhood search for service technician routing and scheduling problems. *Journal of Scheduling*, 15(5):579–600.

[23] Lin, S.-W., Yu, V. F., and Chou, S.-Y. (2009). Solving the truck and trailer routing problem based on a simulated annealing heuristic. *Computers & Operations Research*, 36:1683–1692.

[24] Lin, S.-W., Yu, V. F., and Lu, C.-C. (2011). A simulated annealing heuristic for the truck and trailer routing problem with time windows. *Expert Systems with Applications*, 38(12):15244 – 15252.

[25] Lübbecke, M. E. and Desrosiers, J. (2005). Selected topics in column generation. *Operations Research*, 53(6):1007–1023.

[26] Muter, I., Cordeau, J.-F., and Laporte, G. (2014). A branch-and-price algorithm for the multidepot vehicle routing problem with interdepot routes. *Transportation Science*, 48(3):425–441.

[27] Pessoa, A., Sadykov, R., Uchoa, E., and Vanderbeck, F. (2013). In-out separation and column generation stabilization by dual price smoothing. In Bonifaci, V., Demetrescu, C., and Marchetti-Spaccamela, A., editors, *Experimental Algorithms*, volume 7933 of *Lecture Notes in Computer Science*, pages 354–365. Springer, Berlin.

[28] Poggi, M. and Uchoa, E. (2014). New exact algorithms for the capacitated vehicle routing problem. In Toth, P. and Vigo, D., editors, *Vehicle Routing: Problems, Methods, and Applications*, pages 59–86. SIAM, Philadelphia.

[29] Prodhon, C. and Prins, C. (2014). A survey of recent research on location-routing problems. *European Journal of Operational Research*, 238(1):1–17.

[30] Ropke, S. and Cordeau, J.-F. (2009). Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, 43(3):267–286.

[31] Ropke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472.

[32] Rousseau, L.-M., Gendreau, M., and Feillet, D. (2007). Interior point stabilization for column generation. *Operations Research Letters*, 35(5):660–668.

[33] Savelsbergh, M. W. (1992). The vehicle routing problem with time windows: Minimizing route duration. *ORSA Journal on Computing*, 4(2):146–154.

[34] Scheuerer, S. (2006). A tabu search heuristic for the truck and trailer routing problem. *Computers & Operations Research*, 33:894–909.

[35] Semet, F. and Taillard, E. (1993). Solving real-life vehicle routing problems efficiently using tabu search. *Annals of Operations Research*, 41(4):469–488.

[36] Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265.

[37] Tan, K. C., Chew, Y. H., and Lee, L. (2006). A hybrid multi-objective evolutionary algorithm for solving truck and trailer vehicle routing problems. *European Journal of Operational Research*, 172:855–885.

[38] Vanderbeck, F. (2005). Implementing mixed integer column generation. In *Column Generation*, pages 331–358. Springer, New York.

[39] Villegas, J. G., Prins, C., Prodhon, C., Medaglia, A. L., and Velasco, N. (2011). A grasp with evolutionary path relinking for the truck and trailer routing problem. *Computers & Operations Research*, 38:1319–1334.

[40] Villegas, J. G., Prins, C., Prodhon, C., Medaglia, A. L., and Velasco, N. (2013). A matheuristic for the truck and trailer routing problem. *European Journal of Operational Research*, 230(2):231–244.