

On the One-to-One Pickup-and-Delivery Problem with Time Windows and Trailers

Technical Report LM-2018-05, Revised Version

Michael Drexl

Faculty of Applied Natural Sciences and Industrial Engineering,
Deggendorf Institute of Technology, Deggendorf, Germany
and

Chair of Logistics Management, Gutenberg School of Management and Economics,
Johannes Gutenberg University, Mainz, Germany
E-mail: michael.drexl@th-deg.de

27th April 2020

Abstract

This paper studies an extension of the well-known one-to-one pickup-and-delivery problem with time windows. In the latter problem, requests to transport goods from pickup to delivery locations must be fulfilled by a set of vehicles with limited capacity subject to time window constraints at locations. The goods are not interchangeable; what is picked up at one particular location must be delivered to one particular other location. The extension discussed here consists in the consideration of a heterogeneous vehicle fleet comprising lorries with detachable trailers. Trailers are advantageous as they increase the overall vehicle capacity. However, some locations may be accessible only by a lorry without a trailer. Therefore, special locations are available where trailers can be parked while lorries visit accessibility-constrained locations. This induces a nontrivial tradeoff between an enlarged vehicle capacity and the necessity of scheduling detours for parking and reattaching a trailer.

The contribution of the present paper is threefold: (i) It studies a practically relevant generalization of the one-to-one pickup-and-delivery problem with time windows. (ii) It develops an exact amortized constant-time procedure for testing the feasibility of an insertion of a transport task into a given route with regard to time windows and lorry and trailer capacities. (iii) The paper provides a comprehensive set of new benchmark instances on which the running time of the constant-time test is compared with a naïve one that requires linear time by embedding both tests in an adaptive large neighbourhood search algorithm. The results of computational experiments show that the constant-time test outperforms its linear-time counterpart by one order of magnitude on average.

Keywords: Vehicle routing; Pickup-and-delivery; Trailers; Insertion heuristic; Constant-time feasibility test.

1 Introduction

The one-to-one pickup-and-delivery problem with time windows and trailers (PDPTWT) can be described as follows. There is a set of requests or tasks to transport specified amounts of goods between paired pickup and delivery locations. To fulfil the tasks, a set of capacitated vehicles consisting of *single lorries* and *lorry-trailer combinations (LTCs)* is available. Each vehicle has a given start and a given end location. The start location of a vehicle may differ from the vehicle's end location. A trailer has the same start and the same end location as its associated lorry. Each single lorry and each LTC has a fixed cost, incurred only if it fulfils at least one task, and a travel cost for moving from one location to another. Fixed and travel costs may differ between vehicles; for LTCs, travelling between two locations with the trailer attached may be more expensive than without. Capacities may also differ between vehicles. LTCs have a lorry capacity and a trailer

capacity. After picking up and before delivering the goods of a certain task, vehicles may visit other pickup and/or delivery locations. All pickup and all delivery locations can be visited by a single lorry and by an LTC lorry without its trailer. However, some pickup and some delivery locations may have accessibility constraints in the sense that they cannot be visited by an LTC lorry when the trailer is coupled. Because of these accessibility restrictions, there are also *parking and transshipment locations (PTLs)*. At PTLs, trailers can be decoupled, parked, and re-coupled, and load can be transshipped between an LTC lorry and its trailer. In this paper, a fixed lorry-trailer assignment is assumed. This means that each trailer can be pulled only by one lorry, and only this lorry can transfer load to or from the trailer. All task locations, i.e., all pickup or delivery locations, can be visited by any lorry, all locations designated as reachable by trailer can be visited by any trailer, and PTLs can be visited by all LTC lorries and trailers. Each task location is visited exactly once, whereas PTLs can be visited more than once by the same or different LTCs. The load to be picked up at a task location can be split arbitrarily between a lorry and its trailer if the location is visited by an LTC.

Each location has a single, hard time window that may be equal to the length of the planning horizon and thus nonrestrictive. Arrival at a location before the start of its time window is allowed and incurs waiting time but no cost. Waiting time is not limited. There are fixed service times at all task locations and all PTLs. At PTLs, there are two service times, one for the decoupling and one for the re-coupling operation. Travel times between locations and service times are independent of the current vehicle, of its current load and, for LTCs, of whether or not the trailer is attached. Travel and service times as well as fixed and travel costs are time-independent. All vehicles are available throughout the complete planning horizon.

An LTC route may visit any location and is partitioned into the *main route*, which is the part of the route where the lorry pulls its trailer, and zero or more *subroutes* that start and end at a PTL where the lorry parks its trailer while visiting one or more task locations. An LTC lorry may perform several consecutive subroutes starting and ending at the same PTL before finally pulling away its trailer. If a delivery location is visited on a subroute and the corresponding pickup location has been visited before this subroute, it must be ensured that the entire amount of goods bound for this delivery location is on the lorry at the start of the subroute. This may require a load transfer from a trailer to its lorry at a PTL.

There is no congestion at PTLs: arbitrarily many trailers can be parked at a PTL at the same time. Without loss of generality, it is assumed that a load transfer, if any, between an LTC lorry and its trailer takes place only directly before a decoupling operation, not when re-coupling. The duration (service time) of a decoupling operation includes time for a potential load transfer.

The problem is static and deterministic, i.e., all data are known in advance.

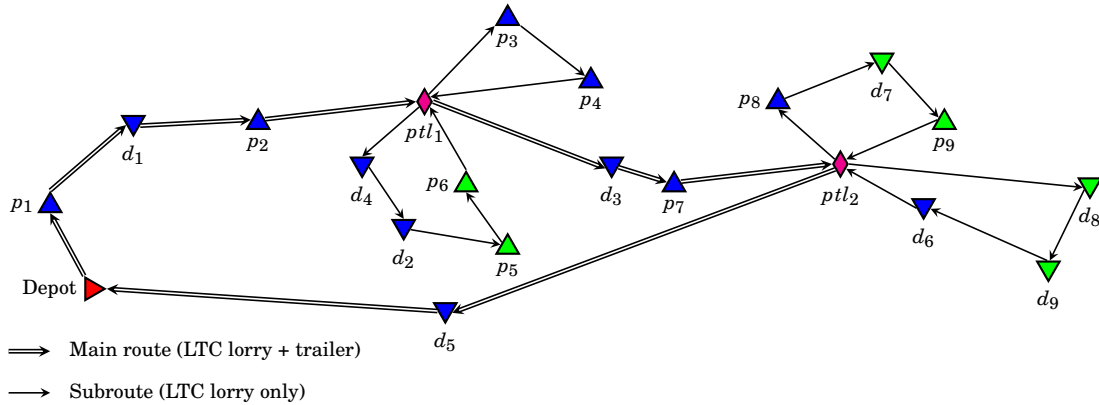
The objective of the PDPTWT is to find a feasible solution with a minimal (or, at least, low) sum of fixed and travel costs. A feasible solution consists of a set of feasible routes, one for each single lorry and one for each LTC, so that each task is covered by exactly one vehicle (single lorry or LTC). A route is feasible if and only if it starts at the start depot of the vehicle that performs the route, fulfils zero or more tasks, and ends at the vehicle's end depot, while maintaining all time windows, accessibility constraints, and lorry and trailer capacities. In a feasible solution, the following nine cases are possible with regard to accessibility constraints:

| | | Pickup can be visited with a trailer | | | | | |
|---|-----|--------------------------------------|-----|-----|----|-----------------------------------|---------------------------------------|
| | | yes | yes | no | no | | |
| Delivery can be visited with a trailer | yes | 1 | 2 | | | yes no yes no | Pickup is visited on main route |
| | yes | 3 | 4 | 5 | 6 | | |
| | no | | 7 | | | | |
| | no | | 8 | | 9 | | |
| | | yes | no | yes | no | Delivery is visited on main route | |

Figure 1 shows an example LTC route that fulfils the nine tasks t_1, \dots, t_9 . For $i = 1, \dots, 9$, p_i and d_i respectively denote the pickup and the delivery location of task t_i . The route starts and ends

at the depot bottom left and performs four subroutes, two each at the parking and transshipment locations ptl_1 and ptl_2 . In the figure, task t_i corresponds to case i of the above table for $i = 1, \dots, 9$. Blue triangles represent locations that can be visited with a trailer; green ones can only be visited without. Triangles pointing upwards represent pickups, those pointing downwards represent deliveries.

Figure 1: Example LTC route



There is no lack of practical applications of the PDPTWT. This author has seen use cases in the supply of supermarkets, beverage stores, and apparel stores, in the transport of ready-mixed concrete garages and commercial waste bins, and, most notably, in the less-than-truckload business, where containers, swap-body platforms, and smaller collective consignments are transported by LTCs.

The contribution of this paper is threefold: (i) It studies a practically relevant extension of the one-to-one pickup-and-delivery problem with time windows. Put differently, it generalizes vehicle routing problems (VRPs, i.e., problems where either all pickups or all deliveries take place at a central depot) with trailers to pickup-and-delivery problems. (ii) It develops an exact amortized constant-time procedure for testing the feasibility of an insertion of a task into a given PDPTWT route concerning time windows and lorry and trailer capacities. ‘Exact’ means that the testing procedure will declare the insertion as feasible if and only if the route resulting from the insertion is feasible. ‘Amortized constant-time’ means that the test itself takes constant time and is independent of the number of tasks (or, equivalently, the number of locations visited) on the route, but that the test uses auxiliary data which must be computed in a preprocessing step which does not run in constant time. (iii) The paper provides a comprehensive set of new benchmark instances and empirically compares the running time of the constant-time test on these instances with a naïve one that requires linear time by embedding both tests in an adaptive large neighbourhood search algorithm for the heuristic solution of the problem. The results of computational experiments show that the constant-time test outperforms its linear-time counterpart by one order of magnitude on average.

The rest of the paper is structured as follows. The next section gives a brief review of related literature. Section 3 presents the adaptive large neighbourhood search procedure used to solve the PDPTWT. In Section 4, the insertion feasibility tests regarding time and capacity are described. Section 5 presents the newly created benchmark instances and discusses the computational results obtained on them. Finally, Section 6 gives a conclusion and proposes topics for further research.

2 Related Work

This section briefly reviews pertinent literature, focussing on works concerned with pickup-and-delivery problems, routing problems with trailers, and efficient feasibility tests in heuristics for routing problems.

Pickup-and-delivery problems (without trailers) exist in several variants (one-to-one, one-to-many-to-one, many-to-many, simultaneous delivery and pickup) and have been extensively studied in the last decades. Important surveys are presented by Parragh et al. [26, 27], Doerner and Salazar-González [9], and Battarra et al. [2]. These works also provide classification schemes for the different variants. The static, deterministic, multi-vehicle, one-to-one variant with time windows is the most widely studied type. Exact algorithms for this problem are presented by Ropke et al. [34], Ropke and Cordeau [33], and Baldacci et al. [1]. According to Battarra et al. [2], the most successful heuristic procedures, by Bent and Van Hentenryck [3] and Ropke and Pisinger [35], are both based on large neighbourhood search.

Routing problems with trailers have also attracted a lot of interest from researchers. The surveys by Prodhon and Prins [32] and Cuda et al. [7] contain sections on VRPs with trailers, which are commonly referred to as truck-and-trailer routing problems (TTRPs). Most works on TTRPs consider no time windows. Exact algorithms for TTRPs with time windows (TTRPTWs) are presented by Parragh and Cordeau [25] and Rothenbächer et al. [36]. Heuristics for TTRPTWs are described by Drexler [10] (heuristic column generation), Lin et al. [22] (simulated annealing), Derigs et al. [8] (hybrid local and large neighbourhood search, attribute-based hill climber), and Parragh and Cordeau [25] (adaptive large neighbourhood search).

Pickup-and-delivery problems with time windows and trailers are less well studied. Most papers on this topic consider approaches for problems where vehicles consisting of a tractor and a semi-trailer are employed to perform full-load tasks, i.e., where a vehicle can transport only one task at a time. Examples are the problems examined by Cheung et al. [6] (attribute-decision model), Xue et al. [46] (tabu search) and Tilk et al. [42] (branch-and-price-and-cut). Concerning the PDPTWT version studied here, this author is aware of only one paper: Bürckert et al. [5] describe a holonic multi-agent system heuristic for a generalization of the PDPTWT in the context of long-distance transport. The authors take into account eight types of resource: driver, lorry with loading capacity, lorry without loading capacity, tractor, trailer, semi-trailer, chassis, and swap-body. Adequate combinations of these resources must be created to fulfil tasks.

Seminal works on *efficient feasibility tests* for insertion or local search procedures for different types of VRPs and PDPs are the ones by Savelsbergh [37, 38, 39], Kindervater and Savelsbergh [20], Funke et al. [13], Irnich et al. [18], Irnich [16, 17], Masson et al. [24], Vidal et al. [43], and Grangier et al. [14]. None of these, however, considers routing problems with trailers.

3 Adaptive Large Neighbourhood Search for the PDPTWT

Adaptive large neighbourhood search (ALNS) is a very widely and successfully used metaheuristic, in particular for, but not limited to, many different types of routing problem. ALNS was first used by Ropke and Pisinger [35] and extends the large neighbourhood search principle introduced by Shaw [40] by adding different removal and reinsertion operators and an adaptive operator selection scheme. Pisinger and Ropke [30] present a tutorial and a literature survey on (A)LNS. The basic idea of large neighbourhood search is to repeatedly perform the following steps. Given an incumbent solution, some of its elements are removed and reinserted to create a new solution that replaces the current incumbent if it either improves the best solution found so far or fulfils some other acceptance criterion.

The ALNS used for the computational experiments described in this paper follows the set-up described by Ropke and Pisinger [35] for the PDPTW without trailers. All removal and reinsertion operators described by Ropke and Pisinger [35] (random, worst and Shaw removal, basic greedy, regret-2, -3, -4 and -M reinsertion) are applied, a roulette wheel procedure with adaptive weight adjustment is employed for selecting the removal and reinsertion operators in each iteration, and a simulated annealing acceptance criterion is used. In addition, the following three removal strategies are built into the ALNS. In the *arc frequency history removal* heuristic, proposed by Masson et al. [23], the aim is to remove tasks that seem to be at bad positions compared to the best known solutions. The heuristic keeps track of how often each arc (connection between two locations) appears in any one of the solutions contained in a fixed-size set composed of the

best solutions found so far. In each ALNS iteration, if a solution enters or leaves this set, the frequencies of the arcs in this solution are incremented or decremented accordingly. When the arc frequency history removal heuristic is selected, a frequency value is computed for each task by summing up the frequencies of the arcs over which the pickup and the delivery locations of the task are reached and left in the current solution. Then, with a certain amount of randomness as proposed by Ropke and Pisinger [35], the tasks with the lowest frequency values are removed. The *zero-split removal* heuristic, proposed by Parragh et al. [28], removes sequences of task locations where the vehicle is empty when reaching the first location and when leaving the last. Longer sequences are preferred, and the removed tasks are reinserted one by one. Finally, the *subroute removal* heuristic, as its name implies, removes entire subroutes, which are selected at random. ‘Removing a subroute’ means that all tasks with at least one location on the subroute are removed. The removed tasks are reinserted one by one in this heuristic, too.

The worst and Shaw removal heuristics exist in a static and a dynamic version. In the static versions, the removal criteria are computed anew only once in an ALNS iteration, in the dynamic versions, they are updated after each removal of a single task. The removal criterion for a task in the worst removal heuristic is the difference in the costs of the current solution with and without the task. The Shaw removal operator uses, for each pair of tasks, a relatedness measure that takes into account the distances between the pickup locations, the distances between the delivery locations, the overlap of the time windows of the pickup locations, the overlap of the time windows of the delivery locations, and the difference between the capacity requirements of the two tasks. In lieu of the noise mechanism used by Ropke and Pisinger [35], the ALNS uses *insertion preference strategies*. In each iteration of a reinsertion heuristic, one of the following five strategies is randomly selected and applied before deciding which task to insert into which route: (i) make the insertion of tasks where the pickup location can be visited with a trailer more attractive; (ii) similar for tasks where this is not the case; (iii) make the insertion into single lorry routes more attractive; (iv) similar for LTC routes; (v) make it more attractive to insert tasks where the pickup location can be visited with a trailer into LTC routes. This is achieved by appropriately modifying the insertion costs of tasks into particular routes.

When, in an insertion step, the creation of a new subroute must be tested, which is necessary if a location not reachable by trailer is to be inserted directly after a location that is left with the trailer coupled, a suitable PTL must be selected. The ALNS does not necessarily choose the PTL closest to the task location in question. Instead, similar to what is done in the removal heuristics, a certain degree of randomness is introduced, with closer PTLs being selected with higher probability.

Apart from that, the decisive modification to the ALNS as described by Ropke and Pisinger [35] is that the time window and capacity feasibility tests described in the next section are used; these take into account trailers and accessibility restrictions.

4 Feasibility Tests

In the following, techniques are proposed to test the temporal and capacitive feasibility of task insertions into routes performed by single lorries or LTCs in constant time, given appropriate auxiliary data computed in a preprocessing step. (In a slight abuse of terminology, ‘amortized constant time’ is abbreviated by ‘constant time’ here and in what follows.) As will be shown, the preprocessing to determine or update the necessary auxiliary data for a route to test time window as well as capacity feasibility takes time quadratic in the number of tasks fulfilled or locations visited on the route, but it is performed only once for a given solution. The resulting data are then used for all feasibility tests, i.e., for testing all potential insertion positions of all unplanned tasks. The routines are embedded in the ALNS metaheuristic described in the previous section. They could, however, also be used within other metaheuristic or local search approaches.

In this section, the following notation is used. Each task t from pickup location p to delivery location d is denoted by $t = (p, d)$ and has a capacity requirement $q^t > 0$, which means that q^t units of load must be picked up at p and $-q^t$ units must be delivered at d . The capacity requirement

at each location u is denoted by q_u . Hence, $q_p > 0$ for each pickup location p , $q_d < 0$ for each delivery location d , and $q_u = 0$ for each vehicle depot or PTL u . Each location u has a single, hard time window $[a_u, b_u]$, $0 \leq a_u \leq b_u \leq T$, where T is the length of the planning horizon. The depot locations have a time window of $[0, T]$. Each task location u has a unique service time (duration) s_u , and each PTL u has a decoupling duration (including a fixed time for a potential load transfer) of s_u^{dec} and a coupling duration of s_u^{coup} . For each pair (u, v) of locations, t_{uv} denotes the travel time from u to v . Each single lorry, each LTC lorry, and each trailer has a specified one-dimensional capacity, denoted by Q_k^l and Q_k^t respectively. For a single lorry k , $Q_k^t = 0$. The symbol ‘= $=$ ’ serves as equality operator, ‘= $=$ ’ is the assignment operator, and ‘ $x += y$ ’ is used as shortcut for ‘ $x = x + y$ ’.

The descriptions assume that feasibility of an insertion of a task $t = (p, d)$ into an existing route $r = (0, 1, \dots, n-1, n)$, with p to be inserted directly after position (zero-based index of the route) h and d to be inserted directly after position i , is to be tested. If p cannot be reached with a trailer, r is performed by an LTC, and the trailer is attached upon leaving h , a location triple $\tilde{p} = ptl_p \rightarrow p \rightarrow ptl_p$ corresponding to a new subroute is inserted after h ; similar for i and d . ptl_p is a suitable trailer parking location; similar for d . Note that p , d , ptl_p , ptl_d , \tilde{p} , and \tilde{d} are locations, whereas h and i are indices on a route. To simplify notation, when referring to a location visited at a certain position on a route, only the index is used: for example, the start of the time window of index i , i.e., of the i th location visited on a route, is denoted by a_i , and the travel time between index i and a to-be-inserted location v is denoted by t_{iv} etc.

Indices h and i indicate positions in the route *before* p and d are inserted. Hence, if $h = i$, then d is to be inserted directly after p , or, if a triple $\tilde{p} = ptl_p \rightarrow p \rightarrow ptl_p$ is to be inserted, directly after the triple. If, however, d cannot be reached with a trailer and p is left with the trailer attached or a triple \tilde{p} is to be inserted, then a triple $\tilde{d} = ptl_d \rightarrow d \rightarrow ptl_d$ is inserted. In principle, if $h = i$ and p or d must be surrounded by a decouple-couple pair, it would also be possible to surround both p and d by one pair. This might be beneficial for instances where many pickups are close to their deliveries. For simplicity of exposition, this additional possibility is not considered in the present paper. When this option is used, constant-time feasibility tests are just as well possible with the auxiliary data structures described in the following subsections; the formal description, though, is tedious. Moreover, in the course of an ALNS, configurations where it is beneficial that the pickup and the delivery of a task are surrounded by a decouple-couple pair will often be achieved automatically as a result of the removal steps.

Several consecutive subroutes by one LTC lorry at the same PTL are modelled by inserting a decouple-couple pair for each subroute. It is assumed that the fixed service times at PTLs are incurred also in such cases.

4.1 Time Windows

In this paper, neither route duration constraints nor time-dependent costs are considered and thus there is no need to strive for minimization of route duration. Under these conditions, it is optimal regarding feasibility to consider only as-early-as-possible schedules, i.e., to assume that a vehicle always leaves a location at the earliest possible point in time; this provides the maximum possible flexibility at subsequent positions on the vehicle’s route.

Testing time-window feasibility of an insertion in linear time is trivial: the locations of the to-be-inserted task are tentatively inserted (including PTLs for decoupling and coupling, if necessary); the route is traversed, starting at the depot at time zero; travel, service and waiting times are added; finally, the resulting earliest possible starts of service are compared with the location time windows.

Testing time-window feasibility in constant time is a little more involved. To do so, Savelsbergh [39] introduced the concept of forward time slack (FTS). The FTS at a position on a route indicates by how much the earliest possible start of service at this position can be postponed without violating a time window at this or a subsequent position on the route. This idea is adapted to test the

feasibility of the insertion of a pickup-and-delivery task (p, d) into a PDPTWT route $r = (0, \dots, n)$ as follows.

First, note that a triple $\tilde{u} = ptl_u \rightarrow u \rightarrow ptl_u$ can be regarded as a *meta-location* or *segment* (cf. Irnich [16], Vidal et al. [43]) and handled as if it were a single location. Hence, whenever the insertion of a triple \tilde{u} needs to be *tested* because the task location u cannot be reached with a trailer, the time window of the corresponding meta-location is tested. (However, when an insertion of a triple for a location u is to be actually *performed*, the sequence $ptl_u \rightarrow u \rightarrow ptl_u$ must be inserted, because the new subroute created by inserting the triple might be enlarged by an insertion of a task location in a later iteration.) The time window $[a_{\tilde{u}}, b_{\tilde{u}}]$ of a meta-location need be precomputed only once, before the start of the ALNS, for each task location u and each PTL ptl . This can be done by setting

$$\begin{aligned} a_{\tilde{u}} &= \max(a_{ptl}, a_u - t_{ptl,u} - s_{ptl}^{dec}), \\ b_{\tilde{u}} &= \min(b_u - t_{ptl,u} - s_{ptl}^{dec}, b_{ptl} - t_{u,ptl} - s_u - t_{ptl,u} - s_{ptl}^{dec}). \end{aligned}$$

If $a_{\tilde{u}} > b_{\tilde{u}}$, then ptl cannot serve as parking location for visiting u . Otherwise, the service time $s_{\tilde{u}}$ of a meta-location \tilde{u} is set to

$$s_{\tilde{u}} = s_{ptl_u}^{dec} + t_{ptl_u,u} + s_u + t_{u,ptl_u} + s_{ptl_u}^{coup}.$$

The travel times to and from a meta-location \tilde{u} are those to and from ptl_u . The travel costs to \tilde{u} are those to ptl_u for a lorry with its trailer plus those from ptl_u to u plus those from u to ptl_u , both for a lorry without its trailer. The travel costs from \tilde{u} are those from ptl_u for a lorry with its trailer.

Second, the following auxiliary data are used:

- e_i : Earliest point in time at which service at index i can begin.
- w_i : Waiting time at index i , i.e., time period between arrival and beginning of service at i .
- w_{ij} : Cumulated waiting time between i and j , i.e., sum of waiting times at indices i, \dots, j .
- sl_i : Slack time from 0 to i , i.e., maximal amount of time by which the departure at the start depot can be postponed from e_0 without violating any time window between 0 and i .
- f_i : Forward time slack from i to n , i.e., maximal amount of time by which e_i can be postponed without violating any time window from i up to the end of the route.

The first four quantities are computed for each route in a preprocessing step as follows:

$$\begin{aligned} e_0 &= a_0; & e_i &= \max(a_i, e_{i-1} + s_{i-1} + t_{i-1,i}); & i &= 1, \dots, n \\ w_0 &= 0; & w_i &= \max(0, a_i - (e_{i-1} + s_{i-1} + t_{i-1,i})); & i &= 1, \dots, n \\ w_{00} &= 0; & w_{0i} &= w_{0,i-1} + w_i; & i &= 1, \dots, n \\ sl_0 &= b_0 - e_0; & sl_i &= \min(sl_{i-1}, b_i - e_i + w_{0,i}); & i &= 1, \dots, n \end{aligned}$$

The FTS can then be computed as $f_i = \min_{j=i, \dots, n}(sl_j)$ for $i = 0, \dots, n$. The computation or update of the first four auxiliary data structures requires linear time in n ; the FTS computation time is quadratic in n . Given these data, time-window feasibility of an insertion can be tested as described in Algorithm 1 (cf. Masson et al. [24]).

Note that it is sufficient to execute lines 1–7 of Algorithm 1 only once for each h with a given PTL ptl_p . If TestTimeWindows returns false in line 7, it makes no sense to test further insertion positions for d with h as insertion position for p or \bar{p} , because neither p nor \bar{p} can be inserted after h or later on r ; hence, the next position for inserting p can be considered.

Due to the limited planning horizon, if a task location not reachable by trailer is to be inserted at a certain position on a main route, i.e., when a new subroute must be created, in principle all PTLs must be tested for whether an insertion at this position is possible. This, of course, increases the running time of an insertion heuristic. However, if only a subset of all PTLs is considered,

an insertion heuristic may miss some feasible solutions, and the solution quality of the overall algorithm may deteriorate. The time window feasibility test described in Algorithm 1 receives as input a particular choice of PTL for the pickup and for the delivery location. Therefore, the test is exact in the sense that it will correctly consider the insertion of a specific triple $ptl_v \rightarrow v \rightarrow ptl_v$ feasible if and only if the insertion of this specific triple is feasible. If several PTLs shall be considered, Algorithm 1 must be embedded in a loop over these PTLs.

Algorithm 1 TestTimeWindows($p, \tilde{p}, d, \tilde{d}, r, h, i$)

Input: Pickup-and-delivery task $t = (p, d)$

Route $r = (0, 1, 2, \dots, n)$

Indices of insertion positions h, i with $0 \leq h \leq i \leq n - 1$

Meta-locations $\tilde{p} = ptl_p \rightarrow p \rightarrow ptl_p$ and $\tilde{d} = ptl_d \rightarrow d \rightarrow ptl_d$ for p and d respectively, for a specific PTL ptl_p for p and a specific PTL ptl_d for d (only if r is performed by an LTC)

Result: Returns **true** iff inserting p or \tilde{p} into r directly after index h and d or \tilde{d} directly after i (or, iff $h = i$, after p or \tilde{p}) is feasible regarding all time windows, **false** otherwise

```

1  $u = p$ 
2 if  $p$  cannot be reached with trailer and trailer is currently attached
3   |  $u = \tilde{p}$ 
4 // Test feasibility of inserting  $u$  after  $h$ 
5  $e_u = \max(a_u, e_h + s_h + t_{h,u})$ 
6 if  $e_u > b_u$ 
7   | return false
8 //  $\Delta_{h+1}$  is the time shift at  $h + 1$ , the increase of  $e_{h+1}$ , caused by inserting
    $u$ 
9  $\Delta_{h+1} = \max(0, e_u + s_u - e_{h+1} + t_{u,h+1})$ 
10 if  $\Delta_{h+1} > f_{h+1}$ 
11   | return false
12  $v = d$ 
13 if  $d$  cannot be reached with trailer and trailer is currently attached
14   |  $v = \tilde{d}$ 
15 // Test feasibility of inserting  $v$  after  $i$ 
16 if  $i > h$  // Delivery not directly after pickup
17   |  $e_v = \max(a_v, a_i + \max(0, \Delta_{h+1} - w_{h+1,i}) + s_i + t_{i,v})$ 
18   | if  $e_v > b_v$ 
19     | return false
20   | //  $\Delta_{i+1}$  is the time shift at  $i + 1$  caused by inserting  $v$ 
21   |  $\Delta_{i+1} = \max(0, e_v + s_v - e_{i+1} + t_{v,i+1})$ 
22   | if  $\Delta_{i+1} > f_{i+1}$ 
23     | return false
24 else //  $i = h$ , i.e., delivery directly after pickup
25   |  $e_v = \max(a_v, e_u + s_u + t_{u,v})$ 
26   | if  $e_v > b_v$ 
27     | return false
28   |  $\Delta_{i+1} = \max(0, e_v + s_v + t_{v,i+1} - e_{i+1})$ 
29   | if  $\Delta_{i+1} > f_{i+1}$ 
30     | return false
31   | return true

```

4.2 Capacities

Time-window tests are the same for single lorry as well as LTC routes: at each position on a route, the earliest start of service must lie within the time window of the respective location. By contrast,

the presence of trailers requires additional capacity tests for LTC routes compared to single lorry routes. In this section, it is first described verbally what must be tested in linear- and constant-time capacity tests. Afterwards, the linear- and constant-time test routines are presented.

At each position of *single lorry routes and main routes of LTCs*, the *total load balance*, which is the difference between the load picked up on the route so far minus the load delivered so far, must be less than or equal to the lorry plus the trailer capacity.

For capacity considerations on *subroutes*, the following two quantities are relevant:

- The *minimal lorry load at decoupling*, i.e., the minimal load that must inevitably be in the lorry upon leaving the decoupling location. This load is equal to the maximum of the following two values:
 - The difference between the total load balance at the decoupling location and the trailer capacity.
 - The sum of the capacity requirements incurred by the deliveries on the subroute whose pickups lie before the subroute. (This value is nonnegative, so that the minimal lorry load at the decoupling location is nonnegative as well.)
- The *subroute load balance* at each position, i.e., the difference between the sum of the load in the lorry at the start of the subroute plus the load picked up on this subroute so far minus the load delivered on this subroute so far. (The subroute load balance can be positive, zero, or negative.)

A subroute is capacity-feasible if and only if the first quantity is less than or equal to the lorry capacity and the value of the second is nonnegative and less than or equal to the lorry capacity at each position.

4.2.1 Testing Capacities in Linear Time

To test capacity-feasibility of an insertion in linear time, the procedure detailed in Algorithm 2 is used. For simplicity, the vehicle index k is omitted: Q^l and Q^t are used instead of Q_k^l and Q_k^t to denote the lorry and the trailer capacity.

Testing capacity in linear time for single-lorry routes is simple: the to-be-inserted task is tentatively inserted, one pass over the route is performed, and the capacity requirement at each position is added to the total load and compared with the lorry capacity (lines 2–6).

Testing capacity for LTC routes is not entirely straightforward even in linear time. As discussed above, it must be known at the start of a subroute how much load must be in the lorry to be able to perform the deliveries whose pickups are not on this subroute. This information is gathered in one forward pass over the route (lines 11–16). (In reality, it is of course not enough to have this amount of load in the lorry at the start of a subroute. It is also necessary to have the *right* commodities aboard the lorry, those that must be delivered on this subroute. This, however, has to be ensured by the driver. For algorithmic planning, it is sufficient to test whether enough loading capacity is available on the lorry.) The second pass (lines 20–38) then performs the actual capacity test on main routes and subroutes (total load at all positions, minimal load at decoupling positions, subroute load balance at all positions on subroutes).

4.2.2 Testing Capacities in Constant Time

To test the feasibility of the insertion of a pickup-and-delivery task (p, d) in constant time, the following data, computed for each route in a preprocessing step, can be used.

1. `TrailerAttached`: An array of boolean values. `TrailerAttached[i]` indicates whether or not the trailer is attached upon leaving (the location corresponding to) index i .
2. `MaxTotalLoadOfSegment`: A two-dimensional array of nonnegative integers. `MaxTotalLoadOfSegment[i][offset]` stores, for an index i on a route, the maximal load balance from the start of the route at any index from i up to and including $i + \text{offset}$. In particular, `MaxTotalLoadOfSegment[i][0]` stores the overall load picked up but not delivered yet from the start depot to and including the location at index i .

Algorithm 2 TestCapacityLinear(r, k)

Input: Route $r = (0, 1, 2, \dots, n)$ with to-be-inserted task tentatively inserted, including decoupling and coupling locations where necessary, and capacity requirements $q_v \in \mathbb{Z}$, $v = 0, 1, 2, \dots, n$

Vehicle k (single lorry or LTC) with lorry and trailer capacities Q^l and Q^t ; for single lorries, $Q^t == 0$

Result: Returns **true** iff lorry and, if applicable, trailer capacity of k are maintained at each index of r , **false** otherwise

```
1 TotalLoad = 0
2 if  $Q^t == 0$  // Test for single lorries
3   for  $v = 0, 1, 2, \dots, n$ 
4     TotalLoad +=  $q_v$ 
5     if TotalLoad >  $Q^l$ 
6       return false
7 else // Test for LTCs
8   LoadDeliveredButNotPickedUpOnSubroute =
9     array of integers of length  $n + 1$ , initialized to 0
10  IndexOfLastDecouple = 0
11  for  $v = 0, 1, 2, \dots, n$ 
12    if  $v$  corresponds to a decoupling location
13      IndexOfLastDecouple =  $v$ 
14    if Trailer is not attached upon leaving  $v$ 
15      if  $v$  corresponds to a delivery and associated pickup is before current subroute
16        LoadDeliveredButNotPickedUpOnSubroute[IndexOfLastDecouple] +=  $(-1) \cdot q_v$ 
17  MinLorryLoadSinceLastDecouple = 0
18  MaxLorryLoad = 0
19  MaxLorryLoadSinceLastDecouple = 0
20  for  $v = 0, 1, 2, \dots, n$ 
21    TotalLoad +=  $q_v$ 
22    if TotalLoad >  $Q^l + Q^t$ 
23      return false
24    MaxLorryLoad =  $\min(\text{TotalLoad}, Q^l)$ 
25    if  $v$  corresponds to a decoupling location
26      MinLorryLoadSinceLastDecouple =
27         $\max(\text{TotalLoad} - Q^t, \text{LoadDeliveredButNotPickedUpOnSubroute}[v])$ 
28      if MinLorryLoadSinceLastDecouple >  $Q^l$ 
29        return false
30      MinLorryLoadSinceLastDecouple =  $\max(\text{MinLorryLoadSinceLastDecouple}, 0)$ 
31      MaxLorryLoadSinceLastDecouple = MaxLorryLoad
32    if Trailer is not attached upon leaving  $v$ 
33      MinLorryLoadSinceLastDecouple =  $\max(\text{MinLorryLoadSinceLastDecouple} + q_v, 0)$ 
34      if MinLorryLoadSinceLastDecouple >  $Q^l$ 
35        return false
36      MaxLorryLoadSinceLastDecouple =  $\min(\text{MaxLorryLoadSinceLastDecouple} + q_v, Q^l)$ 
37      if  $q_v < 0$  and MaxLorryLoadSinceLastDecouple < 0
38        return false
39 return true
```

For example, consider the following route:

| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|----------------------|---|-----|-----|---|-----|-----|-----|----|-----|---|-----|-----|----|----|
| Capacity requirement | 0 | +40 | +10 | 0 | +10 | +20 | -40 | +5 | -10 | 0 | -10 | -20 | -5 | 0 |

This route contains one subroute, which starts at index 3 and ends at index 9, i.e., the zero value at index 3 corresponds to a decoupling process at some PTL, and the zero value at index 9 represents the associated coupling process at this PTL. The load balances at indices 2 to 6 are +50, +50, +60, +80, and +40; thus, $\text{MaxTotalLoadOfSegment}[2][4] = +80$. Moreover, $\text{MaxTotalLoadOfSegment}[8][0] = +35$.

3. **TotalLoadDeliveredButNotPickedUpOnSubroute:** An array of nonnegative integers. If i is an index corresponding to a decoupling location, $\text{TotalLoadDeliveredButNotPickedUpOnSubroute}[i]$ stores the overall load delivered but not picked up on the respective subroute.
4. **LoadBalanceFromStartOfSubroute:** An array of integers. $\text{LoadBalanceFromStartOfSubroute}[i]$ stores, for an index i on a subroute, the positive, negative or zero load balance from the start of the subroute up to and including i .

In the above example route, $\text{LoadBalanceFromStartOfSubroute}[7] = -5 = 10 + 20 - 40 + 5$.

5. **MaxLoadBalanceFromStartOfSubroute:** A two-dimensional array of nonnegative integers. $\text{MaxLoadBalanceFromStartOfSubroute}[i][\text{offset}]$ stores, for an index i on a subroute, the maximum of zero and the largest load balance from the start of the subroute to any index from i up to and including $i + \text{offset}$.

In the above example, $\text{MaxLoadBalanceFromStartOfSubroute}[6][2] = 0 = \max(0, -10, -5, -15) = \max(0, \text{LoadBalanceFromStartOfSubroute}[7])$.

6. **IndexOfLastPrecedingDecouple:** An array of nonnegative integers. $\text{IndexOfLastPrecedingDecouple}[i]$ stores the index where the last decoupling that precedes i on the route occurs.
7. **OffsetOfNextCoupling:** An array of nonnegative integers. $\text{OffsetOfNextCoupling}[i]$ stores the number of positions on the route from i until the next index of a coupling process.

$\text{MaxTotalLoadOfSegment}$ and $\text{MaxLoadBalanceFromStartOfSubroute}$ can be filled using a nested forward pass, i.e., by iterating over all indices $j \geq i$ for each index i on the route. All other data structures described above can be filled or updated by passing through a route once. This means that all necessary preprocessing data for a route can be computed in quadratic time in the number of tasks on the route.

Given these data, the capacity feasibility of an insertion of a task $t = (p, d)$ into an existing route r , with p to be inserted directly after position (zero-based index of the route) h and d to be inserted directly after position i , can be tested as described in Algorithm 3. It is evident that the algorithm itself runs in constant time, i.e., its running time is independent of the number of tasks or the number of locations visited on route r .

Note that, to test the capacity constraints, it is irrelevant whether or not the pickup and/or the delivery location of the task to be inserted must be surrounded by a decouple-couple pair for insertion at the position in question, as decoupling and coupling processes have a capacity requirement of zero.

Note further that, similar to the situation in Algorithm 1, if $\text{TestCapacityConstant}$ returns false from line 3 or line 18, it is unnecessary to consider further potential insertion positions for d for the current insertion position of p . Instead, the next position for inserting p can be considered. Hence, it is sufficient here to execute lines 2–18 of Algorithm 3 only once for each h .

5 Computational Experiments

5.1 Benchmark Instances

To this author's knowledge, there are no benchmark instances for the PDPTWT as studied in this paper. Therefore, a set of instances has been created to perform computational experiments with

Algorithm 3 TestCapacityConstant(p, d, r, h, i, k)

Input: Pickup-and-delivery task $t = (p, d)$ with capacity requirement $q > 0$

Route $r = (0, 1, 2, \dots, n)$

Indices of insertion positions h, i with $0 \leq h \leq i \leq n - 1$

Vehicle k (single lorry or LTC) with lorry and trailer capacities Q^l and Q^t ; for single lorries, $Q^t == 0$

Result: Returns **true** iff inserting p or a triple $\tilde{p} = ptl_p \rightarrow p \rightarrow ptl_p$ into r directly after index h and d or a triple $\tilde{d} = ptl_d \rightarrow d \rightarrow ptl_d$ directly after i (or, iff $h == i$, after p or \tilde{p}) is feasible regarding lorry and trailer capacity, **false** otherwise

```
1 // Evaluate feasibility of insertion regarding total capacity
2 if  $Q^l + Q^t < \text{MaxTotalLoadOfSegment}[h][i - h] + q$ 
3   | return false
4 else if  $Q^t == 0$ 
5   | return true
6 // Evaluate feasibility of insertion regarding lorry capacity
7 if  $i > h$  // Delivery not directly after pickup
8   | // Evaluate feasibility of insertion of pickup
9   | if  $\text{TrailerAttached}[h] == \text{false}$  // Trailer not attached when leaving  $h$ 
10  |   |  $ind = \text{IndexOfLastPrecedingDecouple}[h]$ 
11  |   |  $\text{MinLoadAtDecouple} = \max(\text{MaxTotalLoadOfSegment}[ind][0] - Q^t,$ 
12  |   |   |  $\text{TotalLoadDeliveredButNotPickedUpOnSubroute}[ind])$ 
13  |   |  $\text{LoadAfterPickup} = \text{MinLoadAtDecouple} + \text{LoadBalanceFromStartOfSubroute}[h] + q$ 
14  |   |  $offset = \text{OffsetOfNextCoupling}[h + 1]$ 
15  |   | if  $h + \text{OffsetOfNextCoupling}[h] \geq i$ 
16  |   |   |  $offset = i - h$ 
17  |   | if  $\text{LoadAfterPickup} + \text{MaxLoadBalanceFromStartOfSubroute}[h + 1][\max(0, offset - 1)] > Q^l$ 
18  |   |   | return false
19  | // Evaluate feasibility of insertion of delivery
20  | if  $\text{TrailerAttached}[i] == \text{false}$ 
21  |   | if  $i - h \geq \text{OffsetOfNextCoupling}[h]$  // Delivery not on same subroute as pickup
22  |   |   |  $ind = \text{IndexOfLastPrecedingDecouple}[i]$ 
23  |   |   |  $\text{MinLoadAtDecouple} = \max(\text{MaxTotalLoadOfSegment}[ind][0] - Q^t,$ 
24  |   |   |   |  $\text{TotalLoadDeliveredButNotPickedUpOnSubroute}[ind])$ 
25  |   |   | if  $\text{MinLoadAtDecouple} + \text{MaxLoadBalanceFromStartOfSubroute}[ind][i - ind] + q > Q^l$ 
26  |   |   |   | return false
27 else //  $i == h$ , i.e., delivery directly after pickup
28   | if  $\text{TrailerAttached}[h] == \text{false}$ 
29   |   |  $ind = \text{IndexOfLastPrecedingDecouple}[h]$ 
30   |   |  $\text{MinLoadAtDecouple} = \max(\text{MaxTotalLoadOfSegment}[ind][0] - Q^t,$ 
31   |   |   |  $\text{TotalLoadDeliveredButNotPickedUpOnSubroute}[ind])$ 
32   |   | if  $\text{MinLoadAtDecouple} + \text{LoadBalanceFromStartOfSubroute}[h] + q > Q^l$ 
33   |   |   | return false
34 return true
```

solution procedures. A well-known and widely used set of benchmark instances for pickup-and-delivery problems with time windows and without trailers has been proposed by Li and Lim [21] and is available at www.sintef.no/projectweb/top/pdptw/li-lim-benchmark. This set comprises six classes of instances, with 100, 200, 400, 600, 800, and 1,000 task locations, and thus with 50, 100, 200, 300, 400, and 500 tasks respectively. The instances have been derived from the Solomon instances for the vehicle routing problem with time windows (Solomon [41]), and in analogy to the original data, the Li and Lim instances are also partitioned into six classes LC1, LC2, LR1, LR2, LRC1, and LRC2 according to structural characteristics as follows: ‘C’ stands for geographically clustered tasks which, for the PDPTW and the PDPTWT, also means that the pickup and the delivery location of a task are close together; ‘R’ stands for geographically randomly distributed tasks; ‘1’ stands for restrictive time windows so that only few tasks per route are possible; and ‘2’ stands for less restrictive time windows and a longer planning horizon, which makes longer routes (routes covering more tasks) possible. Each instance has a homogeneous fleet, and start and end depot location of the vehicles coincide.

As pointed out by Derigs et al. [8], p. 544, some benchmark instances for vehicle routing problems with trailers are constructed such that there is no need to use lorry-trailer combinations at all, because the capacity of the lorry is high enough for transporting the entire demand and/or the time windows are so restrictive that a vehicle cannot serve many customers. This has also been observed when trying to modify the Li and Lim instances for use with trailers. Therefore, the benchmark instances for the PDPTWT have two vehicle classes: lorry-trailer combinations and single lorries. The single lorries have artificially high fixed cost, so that they are used only when necessary to ensure that all tasks are covered. Such cases can occur when the time windows of a task are so tight that there is not enough time to decouple the trailer to visit the pickup or the delivery task.

With this in mind, the Li and Lim instances have been adapted to the PDPTWT as follows:

- Every even-numbered location (as listed in the original Li and Lim instance file) is reachable by trailer, i.e., locations 0, 2, 4, 6...; the odd-numbered ones are not.
- Starting with location 0 (the depot), every second location that is reachable by trailer may be used for parking and transshipment; i.e., for locations 0, 4, 8, 12..., a PTL is created. This means that the number of PTLs is approximately half the number of tasks.
- As mentioned, the time windows of task locations are generally too short in the Li and Lim instances, so that no LTCs are used. Therefore, each original time window $[a_u, b_u]$ of a task location u is enlarged to $a_u = \max(0, a_u - TWShift)$ and $b_u = \min(b_u + TWShift, T)$, where $TWShift = \lfloor 100 + (AvgPickupTime + AvgDeliveryTime)/2 \rfloor$, and $AvgPickupTime$ and $AvgDeliveryTime$ respectively indicate the arithmetic mean of the service times at pickup and at delivery locations as indicated in the original files, rounded down to the nearest integer.
- The time windows of parking and transshipment locations are set to the complete planning horizon, i.e., to the time window of the depot. According to the author’s practical experience, this is a mild and realistic assumption.
- The decoupling and coupling service times at PTLs are set to $AvgPickupTime$ and $AvgDeliveryTime$ respectively.
- The number of single lorries as well as the number of LTCs is considered unlimited.
- Single lorries are assigned a fixed cost of 1,000; LTCs have no fixed cost.
- As in the Li and Lim instances, Euclidean distances are used for travel times as well as travel costs. For LTCs, travel times and costs are the same whether or not the trailer is currently attached.
- Capacities of single and LTC lorries are set to the vehicle capacity specified in the respective original instance; trailer capacities are set to 150 % of the lorry capacity.

There is an arc between two locations u and v , i.e., a location v can be visited directly after a location u , unless $a_u + t_u^s + t_{uv} > b_v$, where $t_0^s = 0$ for the depot location 0, $t_u^s = s_u$ for all task locations u , and $t_u^s = \min(AvgPickupTime, AvgDeliveryTime)$ for all PTLs u .

Table 1 shows the distribution of the number of instances of the different types and basic instance characteristics. Note that the number of tasks differs slightly between instances of the same size

Table 1: Instance characteristics

| Size class | Type | No. instances | Average | | | | Average | | | | | |
|--------------|------|---------------|-----------|---------------|----------|-----------|-------------------------|--------------------|---------------------|-----------------------|----------------------|----------------|
| | | | No. tasks | No. locations | No. PTLs | No. arcs | Length planning horizon | Length time window | Pickup service time | Delivery service time | Capacity requirement | Lorry capacity |
| 100 | LC1 | 9 | 53 | 135 | 27 | 16,309 | 1,236 | 584 | 80 | 90 | 19 | 200 |
| | LC2 | 8 | 51 | 130 | 26 | 13,964 | 3,390 | 1,131 | 86 | 90 | 18 | 700 |
| | LR1 | 12 | 53 | 134 | 27 | 17,633 | 230 | 209 | 9 | 10 | 15 | 200 |
| | LR2 | 11 | 51 | 129 | 26 | 15,492 | 1,000 | 562 | 10 | 10 | 15 | 1,000 |
| | LRC1 | 8 | 53 | 136 | 27 | 17,961 | 240 | 220 | 9 | 10 | 17 | 200 |
| | LRC2 | 8 | 51 | 130 | 26 | 15,528 | 960 | 501 | 10 | 10 | 17 | 1,000 |
| | All | 56 | 52 | 132 | 27 | 16,222 | 1,100 | 513 | 32 | 34 | 16 | 543 |
| 200 | LC1 | 10 | 105 | 266 | 53 | 63,713 | 1,351 | 628 | 81 | 90 | 18 | 200 |
| | LC2 | 10 | 101 | 256 | 51 | 54,852 | 3,598 | 1,191 | 87 | 90 | 19 | 700 |
| | LR1 | 10 | 105 | 264 | 53 | 63,138 | 634 | 341 | 9 | 10 | 17 | 200 |
| | LR2 | 10 | 101 | 255 | 51 | 55,630 | 2,535 | 959 | 10 | 10 | 17 | 1,000 |
| | LRC1 | 10 | 105 | 266 | 53 | 64,724 | 634 | 312 | 9 | 10 | 18 | 200 |
| | LRC2 | 10 | 101 | 255 | 51 | 55,989 | 2,535 | 755 | 10 | 10 | 18 | 1,000 |
| | All | 60 | 103 | 260 | 52 | 59,674 | 1,881 | 698 | 34 | 37 | 18 | 550 |
| 400 | LC1 | 10 | 210 | 527 | 106 | 248,962 | 1,501 | 652 | 82 | 90 | 18 | 200 |
| | LC2 | 10 | 203 | 510 | 102 | 217,304 | 3,693 | 1,177 | 87 | 90 | 19 | 700 |
| | LR1 | 10 | 209 | 525 | 105 | 241,818 | 804 | 384 | 9 | 10 | 18 | 200 |
| | LR2 | 10 | 202 | 507 | 102 | 218,047 | 3,213 | 1,135 | 10 | 10 | 18 | 1,000 |
| | LRC1 | 10 | 208 | 523 | 105 | 246,120 | 765 | 337 | 9 | 10 | 18 | 200 |
| | LRC2 | 10 | 203 | 509 | 102 | 220,785 | 3,060 | 827 | 10 | 10 | 18 | 1,000 |
| | All | 60 | 206 | 517 | 104 | 232,172 | 2,173 | 752 | 34 | 37 | 18 | 550 |
| 600 | LC1 | 10 | 315 | 791 | 158 | 551,278 | 1,496 | 653 | 81 | 90 | 18 | 200 |
| | LC2 | 10 | 305 | 764 | 153 | 483,976 | 3,815 | 1,197 | 87 | 90 | 19 | 700 |
| | LR1 | 10 | 314 | 788 | 158 | 512,648 | 1,206 | 476 | 9 | 10 | 18 | 200 |
| | LR2 | 10 | 303 | 759 | 152 | 476,558 | 4,823 | 1,527 | 10 | 10 | 18 | 1,000 |
| | LRC1 | 10 | 314 | 788 | 158 | 517,217 | 1,206 | 391 | 9 | 10 | 18 | 200 |
| | LRC2 | 10 | 303 | 760 | 152 | 472,827 | 4,823 | 1,056 | 10 | 10 | 18 | 1,000 |
| | All | 60 | 309 | 775 | 155 | 502,417 | 2,895 | 883 | 34 | 37 | 18 | 550 |
| 800 | LC1 | 10 | 419 | 1,051 | 210 | 946,974 | 1,676 | 676 | 82 | 90 | 18 | 200 |
| | LC2 | 10 | 406 | 1,018 | 204 | 855,818 | 3,811 | 1,200 | 87 | 90 | 19 | 700 |
| | LR1 | 10 | 418 | 1,049 | 210 | 871,608 | 1,688 | 573 | 9 | 10 | 18 | 200 |
| | LR2 | 10 | 404 | 1,012 | 203 | 836,128 | 6,751 | 1,989 | 10 | 10 | 18 | 1,000 |
| | LRC1 | 10 | 418 | 1,048 | 210 | 848,080 | 1,573 | 431 | 9 | 10 | 18 | 200 |
| | LRC2 | 10 | 404 | 1,012 | 203 | 815,230 | 6,289 | 1,224 | 10 | 10 | 18 | 1,000 |
| | All | 60 | 412 | 1,032 | 207 | 862,306 | 3,631 | 1,016 | 34 | 37 | 18 | 550 |
| 1,000 | LC1 | 10 | 525 | 1,314 | 263 | 1,420,904 | 1,824 | 684 | 82 | 90 | 18 | 200 |
| | LC2 | 10 | 508 | 1,272 | 255 | 1,325,717 | 3,914 | 1,210 | 87 | 90 | 19 | 700 |
| | LR1 | 10 | 523 | 1,309 | 262 | 1,325,967 | 1,925 | 617 | 9 | 10 | 18 | 200 |
| | LR2 | 10 | 504 | 1,263 | 253 | 1,290,588 | 7,697 | 2,195 | 10 | 10 | 18 | 1,000 |
| | LRC1 | 10 | 524 | 1,312 | 263 | 1,228,887 | 1,821 | 453 | 9 | 10 | 18 | 200 |
| | LRC2 | 8 | 505 | 1,266 | 253 | 1,252,281 | 7,284 | 1,492 | 10 | 10 | 18 | 1,000 |
| | All | 58 | 515 | 1,290 | 258 | 1,309,291 | 3,967 | 1,095 | 35 | 38 | 18 | 534 |
| All | All | 354 | 267 | 670 | 134 | 497,857 | 2,617 | 828 | 34 | 36 | 18 | 546 |

class in the Li and Lim instances. Therefore, the values in the columns from ‘No. locations’ to ‘No. Arcs’ are averages, too. By construction of the instances, the column ‘No. Tasks’ also indicates the number of task locations reachable by trailer. Lorry capacities are the same for all instances of the same size class for each type.

5.2 Results

The code was programmed in C++ and compiled with Microsoft Visual Studio Enterprise 2017, Version 15.5.3. The experiments were run on a workstation with the Windows 10 Education operating system, an Intel Xeon E5-1660 v3 @ 3.00 GHz CPU, and 64 GB RAM in single-thread mode. The parameters used in the ALNS are listed in Table 3 in the Appendix.

To assess the relative performance of the linear- and the constant-time test, 10,000 ALNS iterations were performed with both tests for all instances of size classes 100, 200, and 400, i.e., those with at most 200 tasks. For the larger instances, computation times using the linear-time test became too long, so that, for size classes 600, 800, and 1,000, only the constant-time test was used. For the linear-time test, the time windows are tested together with the capacities in the loop of line 3 or 19 in Algorithm 2. The constant-time test first examines time window feasibility, then capacities. Aggregated results are shown in Table 2; detailed results by instance are given in Tables 4–9 in the Appendix.

The most important finding that can be read from Table 2 is that the speedup of the constant-time test compared to the linear one is considerable for all instance types and ranges from a factor of nine to a factor of 142, with an average of 38. This demonstrates that the effort of implementing the constant-time test is well justified.

Further insights that can be obtained from the data in Table 2 are:

- The larger the instance, the higher is the iteration number where the best solution was found.
- The number of routes in the best solution found can differ significantly between instances of the same size class and type.
- As LTC routes have no fixed cost, most routes are actually LTC routes. This also shows that the instances are a suitable test bed for routing problems with trailers (remember the comment on page 13).
- In particular for the larger instances with long planning horizon and wide time windows, the number of subroutes greatly exceeds the number of LTC routes, meaning that the average LTC route performs more than one subroute. Most PTLs are used only once.
- The running times for the instances with more tasks per route, i.e., fewer routes, are consistently higher than those for the other instances.
- The speedup for the LR and LRC instances increases with increasing instance size; for the LC instances, this is not the case.
- The speedup is significantly greater for the instances with fewer routes (classes with ‘2’).

6 Conclusions and Outlook

This paper has studied the PDPTWT, a routing problem which aims at fulfilling a set of transport tasks between pickup and delivery locations, subject to time window constraints and accessibility restrictions, by means of a fleet consisting of single lorries and lorry-trailer combinations. Procedures to test the temporal and capacitive feasibility of inserting a task into an existing route have been presented. Given adequate data computed in a preprocessing step, these procedures run in constant time. They have been embedded in an adaptive large neighbourhood search algorithm for the heuristic solution of the PDPTWT. A comprehensive set of benchmark instances has also been created. The results of computational experiments are presented which show significant speedups that can be realized with the constant-time feasibility test.

Topics for further research abound.

As the focus of the research presented here was on efficient feasibility testing, not on solution quality, many options exist regarding *algorithmic refinements* to improve solution quality of the ALNS. First of all, local and/or very large-scale neighbourhood search routines could be added, as done, e.g., by Derigs et al. [8] and Gschwind and Drexl [15]. Also, matheuristic components, e.g., solving a set-covering problem with all generated routes at the end of the ALNS, cf. Parragh and Schmid [29], Villegas et al. [45], could be helpful. Another refinement would be to add a splitting

Table 2: Aggregated computational results (minimum / average / maximum)

| Size class | Type | Iteration where best solution was found | No. routes | No. LTC routes | No. sub-routes | No. PTLs used | Running time ALNS with constant-time test | Ratio running time linear / constant |
|--------------|------|---|------------|----------------|----------------|---------------|---|--------------------------------------|
| 100 | LC1 | 1,850/5,505/9,249 | 10/ 11/ 11 | 10/11/ 11 | 10/10/ 11 | 6/ 8/10 | 42/ 44/ 47 | 10/ 11/ 12 |
| | LC2 | 282/3,956/8,699 | 4/ 4/ 4 | 4/ 4/ 4 | 3/ 4/ 4 | 2/ 3/ 4 | 71/ 78/ 90 | 35/ 37/ 39 |
| | LR1 | 4,794/8,111/9,850 | 11/ 11/ 11 | 11/11/ 11 | 10/11/ 12 | 6/ 7/ 9 | 39/ 42/ 45 | 9/ 9/ 10 |
| | LR2 | 414/5,977/9,916 | 2/ 3/ 4 | 2/ 3/ 4 | 2/ 3/ 5 | 1/ 3/ 5 | 75/ 112/ 156 | 33/ 47/ 58 |
| | LRC1 | 4,938/8,062/9,916 | 11/ 12/ 12 | 11/12/ 12 | 11/12/ 14 | 6/ 8/11 | 39/ 42/ 43 | 9/ 9/ 10 |
| | LRC2 | 2,474/5,334/9,999 | 3/ 3/ 4 | 3/ 3/ 4 | 3/ 4/ 5 | 2/ 3/ 4 | 74/ 92/ 110 | 32/ 39/ 43 |
| | All | 282/6,276/9,999 | 2/ 7/ 12 | 2/ 7/ 12 | 2/ 7/ 14 | 1/ 5/11 | 39/ 68/ 156 | 9/ 25/ 58 |
| 200 | LC1 | 5,116/8,530/9,980 | 20/ 21/ 21 | 20/21/ 21 | 20/21/ 22 | 12/14/16 | 148/ 157/ 169 | 11/ 12/ 12 |
| | LC2 | 2,762/5,676/9,606 | 6/ 7/ 8 | 6/ 7/ 8 | 6/ 7/ 9 | 4/ 6/ 8 | 240/ 280/ 350 | 36/ 39/ 41 |
| | LR1 | 4,497/7,663/9,645 | 8/ 10/ 11 | 8/10/ 11 | 8/12/ 16 | 7/ 9/11 | 175/ 198/ 229 | 19/ 22/ 26 |
| | LR2 | 294/7,799/9,949 | 3/ 4/ 5 | 3/ 4/ 5 | 5/ 8/ 15 | 3/ 7/13 | 375/ 617/ 987 | 69/ 90/121 |
| | LRC1 | 6,133/8,675/9,878 | 9/ 11/ 12 | 9/11/ 12 | 10/12/ 15 | 7/10/13 | 175/ 190/ 230 | 18/ 21/ 27 |
| | LRC2 | 212/6,869/9,759 | 4/ 5/ 6 | 4/ 5/ 6 | 5/10/ 18 | 5/ 9/17 | 314/ 482/ 657 | 53/ 69/ 82 |
| | All | 212/7,535/9,980 | 3/ 9/ 21 | 3/ 9/ 21 | 5/12/ 22 | 3/ 9/17 | 148/ 321/ 987 | 11/ 42/121 |
| 400 | LC1 | 7,750/9,320/9,998 | 36/ 39/ 42 | 36/39/ 42 | 35/40/ 42 | 19/25/29 | 468/ 494/ 521 | 11/ 11/ 12 |
| | LC2 | 5,247/8,429/9,977 | 12/ 13/ 14 | 12/13/ 14 | 13/14/ 15 | 9/12/14 | 735/ 854/1,048 | 34/ 36/ 38 |
| | LR1 | 6,366/8,456/9,987 | 15/ 18/ 21 | 15/18/ 21 | 17/21/ 26 | 13/18/20 | 538/ 630/ 763 | 19/ 24/ 31 |
| | LR2 | 4,193/8,404/9,961 | 5/ 7/ 9 | 5/ 7/ 9 | 13/19/ 25 | 12/17/22 | 1,198/1,943/3,432 | 76/101/142 |
| | LRC1 | 6,771/8,677/9,989 | 15/ 21/ 24 | 15/21/ 24 | 17/23/ 27 | 14/18/21 | 493/ 554/ 681 | 18/ 21/ 29 |
| | LRC2 | 3,927/8,600/9,968 | 7/ 9/ 12 | 7/ 9/ 12 | 13/22/ 27 | 13/19/23 | 940/1,394/2,434 | 59/ 78/108 |
| | All | 3,927/8,648/9,998 | 5/ 18/ 42 | 5/18/ 42 | 13/23/ 42 | 9/18/29 | 468/ 978/3,432 | 11/ 45/142 |
| 600 | LC1 | 8,262/9,523/9,958 | 57/ 61/ 64 | 57/61/ 64 | 55/61/ 65 | 34/40/45 | 778/ 820/ 868 | |
| | LC2 | 8,330/9,425/9,964 | 19/ 21/ 22 | 19/21/ 22 | 20/24/ 32 | 17/22/28 | 1,092/1,259/1,505 | |
| | LR1 | 3,047/8,285/9,981 | 18/ 25/ 31 | 17/24/ 31 | 23/34/ 47 | 21/30/37 | 1,010/1,163/1,452 | |
| | LR2 | 7,062/8,658/9,915 | 7/ 10/ 12 | 7/10/ 12 | 28/42/ 60 | 25/35/48 | 2,216/3,584/5,803 | |
| | LRC1 | 7,960/9,290/9,991 | 18/ 28/ 32 | 18/28/ 32 | 25/32/ 36 | 22/26/31 | 876/ 994/1,391 | |
| | LRC2 | 5,731/9,177/9,995 | 8/ 12/ 17 | 8/12/ 17 | 29/44/ 61 | 25/37/53 | 1,742/2,394/4,247 | |
| | All | 3,047/9,060/9,995 | 7/ 26/ 64 | 7/26/ 64 | 20/40/ 65 | 17/32/53 | 778/1,702/5,803 | |
| 800 | LC1 | 7,969/9,154/9,965 | 73/ 81/ 87 | 70/80/ 87 | 70/79/ 84 | 47/53/58 | 1,081/1,141/1,202 | |
| | LC2 | 7,271/9,407/9,942 | 27/ 28/ 30 | 24/27/ 30 | 31/36/ 48 | 23/30/39 | 1,484/1,754/1,923 | |
| | LR1 | 7,632/9,187/9,998 | 22/ 32/ 39 | 19/27/ 36 | 28/41/ 57 | 26/37/48 | 1,406/1,628/2,083 | |
| | LR2 | 7,592/9,470/9,942 | 7/ 13/ 17 | 7/12/ 16 | 30/58/ 81 | 26/50/71 | 2,816/4,581/7,905 | |
| | LRC1 | 8,637/9,521/9,989 | 26/ 40/ 47 | 24/36/ 40 | 35/43/ 50 | 32/37/43 | 1,116/1,286/1,743 | |
| | LRC2 | 6,887/8,658/9,992 | 12/ 16/ 21 | 12/15/ 20 | 35/68/103 | 32/58/79 | 2,040/2,893/4,923 | |
| | All | 6,887/9,233/9,998 | 7/ 35/ 87 | 7/33/ 87 | 28/54/103 | 23/44/79 | 1,081/2,214/7,905 | |
| 1,000 | LC1 | 5,994/8,963/9,999 | 92/103/110 | 88/95/103 | 88/94/108 | 62/68/78 | 1,310/1,477/1,597 | |
| | LC2 | 7,300/9,098/9,988 | 34/ 36/ 38 | 32/35/ 36 | 40/48/ 58 | 33/42/52 | 1,956/2,196/2,581 | |
| | LR1 | 8,048/9,504/9,997 | 27/ 40/ 50 | 22/32/ 40 | 37/54/ 68 | 34/48/55 | 1,822/2,151/2,800 | |
| | LR2 | 8,723/9,477/9,910 | 10/ 16/ 21 | 10/14/ 17 | 43/69/ 99 | 42/62/83 | 3,275/5,497/1,002 | |
| | LRC1 | 7,727/9,518/9,998 | 33/ 51/ 59 | 29/41/ 48 | 50/57/ 71 | 42/48/60 | 1,698/1,888/2,665 | |
| | LRC2 | 8,358/9,499/9,934 | 14/ 20/ 24 | 13/18/ 21 | 56/89/118 | 50/75/97 | 2,937/4,035/6,626 | |
| | All | 5,994/9,338/9,999 | 10/ 45/110 | 10/40/103 | 37/68/118 | 33/56/97 | 1,310/2,834/1,002 | |
| All | All | 212/8,366/9,999 | 2/ 23/110 | 2/22/103 | 2/34/118 | 1/28/97 | 39/1,359/1,002 | 9/ 38/142 |

procedure based on dynamic programming that finds optimal PTLs for given routes, cf. Prins [31] and Villegas et al. [44].

Regarding *modelling extensions*, many additional practically relevant constraints could be taken into account. Two particularly interesting extensions are loading constraints such as last-in-first-out, and the impossibility of transferring load between an LTC lorry and its trailer. Of special relevance in connexion with constant-time feasibility tests are limits on route duration and on the time or the number of intermediate stops between the pickup and the delivery of a task. Load-dependent service times require an optimization of the load transfer amounts from lorry to trailer at decoupling and coupling locations, a considerable additional intricacy. Time-dependent costs (and route duration constraints, too) lead to the difficult situation that an as-early-as-possible schedule need no longer be optimal (Savelsbergh [39]), thus violating a fundamental assumption on which the feasibility tests described in the present paper are based.

Also *other variants of pickup-and-delivery problems*, such as one-to-many-to-one problems (also called vehicle routing problems with backhauls, Irnich et al. [19]), many-to-many, and simultaneous PDPs (Battarra et al. [2]), lend themselves to consider a fleet containing trailers.

Furthermore, in many pickup-and-delivery applications, the possibility or even the requirement to split tasks exists (cf. the survey by Drexel [11] and the more recent papers by Masson et al. [24] and Grangier et al. [14]). This means that a task $t = (p, d)$ can be decomposed into two subtasks or legs, (p, tl) and (tl, d) at transshipment locations tl . The legs of split tasks can be performed by different vehicles, and this creates an interdependence between routes: changes in one route may make one or several or all other routes infeasible. This interdependence requires a synchronization regarding time and load and, when trailers are considered, leads to the *PDPTWT with synchronization*.

Of course, all of the above extensions and variants can also be considered in a *dynamic and/or stochastic context*, where some information becomes known only after execution of a route plan has begun and/or some data are known only in the form of random variables, cf. Berbeglia et al. [4] and Flatberg et al. [12].

Finally, there is yet no *exact algorithm* for solving the PDPTWT. Computing optimal solutions to larger PDPTWT instances is surely a challenging but worthwhile endeavour.

References

- [1] Baldacci R, Bartolini E, Mingozzi A (2011): *An Exact Algorithm for the Pickup and Delivery Problem with Time Windows*. *Operations Research* 59(2): 414–426.
- [2] Battarra M, Cordeau JF, Iori M (2014): *Pickup-and-Delivery Problems for Goods Transportation*. In: Toth P, Vigo D (eds): *Vehicle Routing: Problems, Methods, and Applications*. Society for Industrial and Applied Mathematics, Philadelphia. 161–191.
- [3] Bent R, Van Hentenryck P (2006): *A Two-Stage Hybrid Algorithm for Pickup and Delivery Vehicle Routing Problems with Time Windows*. *Computers & Operations Research* 33: 875–893.
- [4] Berbeglia G, Cordeau JF, Laporte G (2010): *Dynamic Pickup and Delivery Problems*. *European Journal of Operational Research* 202(1): 8–15.
- [5] Bürckert H, Fischer K, Vierke G (2000): *Holonic Transport Scheduling with TELETRUCK*. *Applied Artificial Intelligence* 14: 697–725.
- [6] Cheung R, Shi N, Powell W, Simão H (2008): *An Attribute-Decision Model for Cross-Border Drayage Problem*. *Transportation Research Part E* 44(2): 217–234.
- [7] Cuda R, Guastaroba G, Speranza M (2015): *A Survey on Two-Echelon Routing Problems*. *Computers & Operations Research* 55: 185–199.
- [8] Derigs U, Pullmann M, Vogel U (2013): *Truck and Trailer Routing—Problems, Heuristics and Computational Experience*. *Computers & Operations Research* 40(2): 536–546.

- [9] Doerner K, Salazar-González J (2014): *Pickup-and-Delivery Problems for People Transportation*. In: Vigo D, Toth P (eds): *Vehicle Routing: Problems, Methods, and Applications*. Society for Industrial and Applied Mathematics, Philadelphia. 193–212.
- [10] Drexel M (2011): *Branch-and-Price and Heuristic Column Generation for the Generalized Truck-and-Trailer Routing Problem*. *Journal of Quantitative Methods for Economics and Business Administration* 12: 5–38.
- [11] Drexel M (2012): *Synchronization in Vehicle Routing—A Survey of VRPs with Multiple Synchronization Constraints*. *Transportation Science* 46(3): 297–316.
- [12] Flatberg T, Hasle G, Kloster O, Nilssen E, Riise A (2005): *Dynamic and Stochastic Aspects in Vehicle Routing – A Literature Survey*. Technical Report STF90A05413, SINTEF.
- [13] Funke B, Grünert T, Irnich S (2005): *Local Search for Vehicle Routing and Scheduling Problems: Review and Conceptual Integration*. *Journal of Heuristics* 11(4): 267–306.
- [14] Grangier P, Gendreau M, Lehuédé F, Rousseau LM (2016): *An Adaptive Large Neighborhood Search for the Two-Echelon Multiple-Trip Vehicle Routing Problem with Satellite Synchronization*. *European Journal of Operational Research* 254(1): 80–91.
- [15] Gschwind T, Drexel M (2019): *Adaptive Large Neighborhood Search with a Constant-Time Feasibility Test for the Dial-a-Ride Problem*. *Transportation Science* 53(2): 480–491.
- [16] Irnich S (2008): *Resource Extension Functions: Properties, Inversion, and Generalization to Segments*. *OR Spectrum* 30(1): 113–148.
- [17] Irnich S (2008): *A Unified Modeling and Solution Framework for Vehicle Routing and Local Search-Based Metaheuristics*. *INFORMS Journal on Computing* 20(2): 270–287.
- [18] Irnich S, Funke B, Grünert T (2006): *Sequential Search and its Application to Vehicle-Routing Problems*. *Computers & Operations Research* 33(8): 2405–2429.
- [19] Irnich S, Schneider M, Vigo D (2014): *Four Variants of the Vehicle Routing Problem*. In: Toth P, Vigo D (eds): *Vehicle Routing: Problems, Methods, and Applications*. Society for Industrial and Applied Mathematics, Philadelphia. 241–271.
- [20] Kindervater G, Savelsbergh M (1997): *Vehicle Routing: Handling Edge Exchanges*. In: Aarts E, Lenstra J (eds): *Local Search in Combinatorial Optimization*. Wiley, Chichester. 337–360.
- [21] Li H, Lim A (2003): *A Metaheuristic for the Pickup and Delivery Problem with Time Windows*. *International Journal on Artificial Intelligence Tools* 12: 173–186.
- [22] Lin S, Yu V, Lu C (2011): *A Simulated Annealing Heuristic for the Truck and Trailer Routing Problem with Time Windows*. *Expert Systems with Applications* 38(12): 15244–15252.
- [23] Masson R, Lehuédé F, Péton O (2013): *An Adaptive Large Neighborhood Search for the Pickup and Delivery Problem with Transfers*. *Transportation Science* 47(3): 344–355.
- [24] Masson R, Lehuédé F, Péton O (2013): *Efficient Feasibility Testing for Request Insertion in the Pickup and Delivery Problem with Transfers*. *Operations Research Letters* 41(3): 211–215.
- [25] Parragh S, Cordeau JF (2017): *Branch-and-Price and Adaptive Large Neighborhood Search for the Truck and Trailer Routing Problem with Time Windows*. *Computers & Operations Research* 83: 28–44.
- [26] Parragh S, Doerner K, Hartl R (2008): *A Survey on Pickup and Delivery Models Part I: Transportation between Customers and Depot*. *Journal für Betriebswirtschaft* 58(1): 21–51.
- [27] Parragh S, Doerner K, Hartl R (2008): *A Survey on Pickup and Delivery Models Part II: Transportation between Pickup and Delivery Locations*. *Journal für Betriebswirtschaft* 58(2): 81–117.
- [28] Parragh S, Doerner K, Hartl R (2010): *Variable Neighborhood Search for the Dial-a-Ride Problem*. *Computers & Operations Research* 77(6): 58–71.

- [29] Parragh S, Schmid V (2013): *Hybrid Column Generation and Large Neighborhood Search for the Dial-A-Ride Problem*. *Computers & Operations Research* 40(1): 490–497.
- [30] Pisinger D, Ropke S (2010): *Large Neighborhood Search*. In: Gendreau M, Potvin JY (eds): *Handbook of Metaheuristics, Second Edition*. International Series in Operations Research & Management Science 146. Springer, Boston. 399–419.
- [31] Prins C (2004): *A Simple and Effective Evolutionary Algorithm for the Vehicle Routing Problem*. *Computers & Operations Research* 31(12): 1985–2002.
- [32] Prodhon C, Prins C (2014): *A Survey of Recent Research on Location-Routing Problems*. *European Journal of Operational Research* 238(1): 1–17.
- [33] Ropke S, Cordeau JF (2009): *Branch and Cut and Price for the Pickup and Delivery Problem with Time Windows*. *Transportation Science* 43(3): 267–286.
- [34] Ropke S, Cordeau JF, Laporte G (2007): *Models and Branch-and-Cut Algorithms for Pickup-and-Delivery Problems with Time Windows*. *Networks* 49(4): 258–272.
- [35] Ropke S, Pisinger D (2006): *An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows*. *Transportation Science* 40(4): 455–472.
- [36] Rothenbächer AK, Drexl M, Irnich S (2018): *Branch-and-Price-and-Cut for the Truck-and-Trailer Routing Problem with Time Windows*. *Transportation Science* 52(5): 1174–1190.
- [37] Savelsbergh M (1985): *Local Search in Routing Problems with Time Windows*. *Annals of Operations Research* 4(1): 285–305.
- [38] Savelsbergh M (1990): *An Efficient Implementation of Local Search Algorithms for Constrained Routing Problems*. *European Journal of Operational Research* 47(1): 75–85.
- [39] Savelsbergh M (1992): *The Vehicle Routing Problem with Time Windows: Minimizing Route Duration*. *ORSA Journal on Computing* 4(2): 146–154.
- [40] Shaw P (1997): *A New Local Search Algorithm Providing High Quality Solutions to Vehicle Routing Problems*. Technical Report, Department of Computer Science, University of Strathclyde.
- [41] Solomon M (1987): *Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints*. *Operations Research* 35: 254–265.
- [42] Tilk C, Bianchessi N, Drexl M, Irnich S, Meisel F (2018): *Branch-and-Price-and-Cut for the Active-Passive Vehicle-Routing Problem*. *Transportation Science* 52(2): 300–319.
- [43] Vidal T, Crainic TG, Gendreau M, Prins C (2014): *A Unified Solution Framework for Multi-Attribute Vehicle Routing Problems*. *European Journal of Operational Research* 234(3): 658–673.
- [44] Villegas J, Prins C, Prodhon C, Medaglia A, Velasco N (2011): *A GRASP with Evolutionary Path Relinking for the Truck and Trailer Routing Problem*. *Computers & Operations Research* 38(9): 1319–1334.
- [45] Villegas J, Prins C, Prodhon C, Medaglia A, Velasco N (2013): *A Matheuristic for the Truck and Trailer Routing Problem*. *European Journal of Operational Research* 230(2): 231–244.
- [46] Xue Z, Zhang C, Lin W, Miao L, Yang P (2014): *A Tabu Search Heuristic for the Local Container Drayage Problem under a New Operation Mode*. *Transportation Research Part E* 62: 136–150.

Appendix

The subsequent Table 3 specifies the parameter settings of the ALNS used for the computational experiments. The following Tables 4–9 present the detailed computational results for each of the benchmark instances with these settings. Table 2 was compiled based on these data. Euclidean distances were computed with full double precision, 10,000 iterations were performed, and the objective function values were rounded to three digits.

Table 3: ALNS parameter settings

| Parameter | Value |
|--|--------------|
| Value for computing start temperature | 5 |
| Cooling rate | 0.99975 |
| Maximum number of iterations between update of performance statistics | 100 |
| Score 1 | 33 |
| Score 2 | 9 |
| Score 3 | 13 |
| Score update factor | 0.1 |
| Absolute parameter for determining minimal number of tasks to be removed per iteration | 30 |
| Absolute parameter for determining maximal number of tasks to be removed per iteration | 60 |
| Relative parameter for determining minimal number of tasks to be removed per iteration | 0.1 |
| Relative parameter for determining maximal number of tasks to be removed per iteration | 0.4 |
| Randomization degree of worst removal heuristics | 3 |
| Randomization degree of Shaw removal heuristics | 6 |
| Randomization degree of arc frequency history removal heuristic | 6 |
| Randomization degree of zero split removal heuristic | 6 |
| Randomization degree of subroute removal heuristic | 6 |
| Distance weight parameter of Shaw removal heuristics | 9 |
| Time weight parameter of Shaw removal heuristics | 3 |
| Load weight parameter of Shaw removal heuristics | 2 |
| Number of solutions to be considered for arc frequency history removal | 50 |

Table 4: Detailed computational results size class 100

| Instance | Objective function value | Iteration where best solution was found | No. routes | No. LTC routes | No. subroutes | No. PTLs used | Running time ALNS with constant-time test (secs.) | Ratio running time linear / constant |
|----------|--------------------------|---|------------|----------------|---------------|---------------|---|--------------------------------------|
| lc101 | 970.473 | 6,326 | 11 | 11 | 10 | 9 | 42 | 10.7 |
| lc102 | 944.132 | 6,396 | 10 | 10 | 10 | 9 | 44 | 10.4 |
| lc103 | 1,015.599 | 8,318 | 11 | 11 | 10 | 8 | 42 | 10.6 |
| lc104 | 951.527 | 3,546 | 10 | 10 | 10 | 9 | 46 | 11.6 |
| lc105 | 986.038 | 5,441 | 10 | 10 | 10 | 7 | 43 | 11.1 |
| lc106 | 972.096 | 5,194 | 11 | 11 | 10 | 8 | 44 | 10.8 |
| lc107 | 1,013.618 | 1,850 | 11 | 11 | 11 | 9 | 45 | 10.7 |
| lc108 | 998.764 | 9,249 | 11 | 11 | 11 | 10 | 46 | 10.7 |
| lc109 | 959.115 | 3,228 | 10 | 10 | 10 | 6 | 47 | 11.3 |
| lc201 | 691.886 | 3,004 | 4 | 4 | 4 | 4 | 71 | 37.7 |
| lc202 | 716.538 | 7,761 | 4 | 4 | 4 | 4 | 79 | 35.5 |
| lc203 | 725.034 | 282 | 4 | 4 | 4 | 3 | 80 | 36.1 |
| lc204 | 659.023 | 8,699 | 4 | 4 | 4 | 4 | 90 | 36.1 |
| lc205 | 661.215 | 4,978 | 4 | 4 | 3 | 3 | 71 | 35.7 |
| lc206 | 664.417 | 1,897 | 4 | 4 | 4 | 2 | 76 | 38.1 |
| lc207 | 679.853 | 1,363 | 4 | 4 | 3 | 3 | 77 | 37.5 |
| lc208 | 676.502 | 3,667 | 4 | 4 | 3 | 3 | 79 | 38.8 |
| lr101 | 1,138.993 | 7,753 | 11 | 11 | 11 | 7 | 43 | 9.6 |
| lr102 | 1,171.374 | 4,794 | 11 | 11 | 11 | 6 | 45 | 10.0 |
| lr103 | 1,223.400 | 4,819 | 11 | 11 | 10 | 7 | 40 | 9.1 |
| lr104 | 1,090.987 | 8,552 | 11 | 11 | 10 | 6 | 41 | 9.6 |
| lr105 | 1,136.728 | 9,054 | 11 | 11 | 10 | 6 | 43 | 9.6 |
| lr106 | 1,152.816 | 9,850 | 11 | 11 | 11 | 9 | 40 | 9.4 |
| lr107 | 1,216.950 | 9,236 | 11 | 11 | 11 | 8 | 40 | 9.2 |
| lr108 | 1,162.082 | 6,254 | 11 | 11 | 10 | 6 | 39 | 8.8 |
| lr109 | 1,127.926 | 9,309 | 11 | 11 | 12 | 6 | 43 | 9.9 |
| lr110 | 1,109.671 | 9,538 | 11 | 11 | 12 | 6 | 40 | 9.6 |
| lr111 | 1,207.751 | 9,313 | 11 | 11 | 12 | 9 | 45 | 9.5 |
| lr112 | 1,162.763 | 8,856 | 11 | 11 | 11 | 7 | 42 | 9.7 |
| lr201 | 1,041.385 | 7,809 | 4 | 4 | 4 | 3 | 75 | 33.2 |
| lr202 | 1,071.902 | 7,699 | 3 | 3 | 3 | 3 | 91 | 40.6 |
| lr203 | 969.654 | 7,084 | 3 | 3 | 5 | 5 | 107 | 45.6 |
| lr204 | 846.491 | 4,767 | 2 | 2 | 2 | 2 | 142 | 57.6 |
| lr205 | 992.878 | 7,503 | 3 | 3 | 4 | 3 | 98 | 43.1 |
| lr206 | 975.402 | 9,916 | 3 | 3 | 5 | 5 | 103 | 44.3 |
| lr207 | 884.184 | 3,222 | 2 | 2 | 2 | 1 | 139 | 57.1 |
| lr208 | 734.643 | 414 | 2 | 2 | 2 | 1 | 156 | 57.1 |
| lr209 | 928.528 | 2,202 | 3 | 3 | 4 | 4 | 104 | 44.4 |
| lr210 | 938.639 | 5,383 | 3 | 3 | 3 | 2 | 109 | 48.5 |
| lr211 | 833.859 | 9,747 | 3 | 3 | 3 | 2 | 104 | 45.4 |
| lrc101 | 1,395.622 | 8,464 | 12 | 12 | 12 | 8 | 42 | 8.9 |
| lrc102 | 1,505.317 | 9,916 | 12 | 12 | 12 | 11 | 42 | 8.7 |
| lrc103 | 1,257.967 | 8,890 | 11 | 11 | 11 | 6 | 42 | 9.2 |
| lrc104 | 1,197.082 | 6,581 | 11 | 11 | 11 | 8 | 43 | 9.6 |
| lrc105 | 1,502.650 | 9,346 | 12 | 12 | 14 | 10 | 43 | 8.9 |
| lrc106 | 1,440.623 | 4,938 | 12 | 12 | 12 | 9 | 41 | 8.8 |
| lrc107 | 1,315.167 | 8,371 | 11 | 11 | 12 | 8 | 41 | 9.0 |
| lrc108 | 1,284.733 | 7,992 | 11 | 11 | 11 | 7 | 39 | 9.2 |
| lrc201 | 1,311.424 | 9,792 | 4 | 4 | 5 | 4 | 74 | 32.1 |
| lrc202 | 1,192.597 | 5,102 | 3 | 3 | 3 | 2 | 82 | 36.5 |
| lrc203 | 1,020.702 | 9,999 | 3 | 3 | 3 | 3 | 97 | 40.2 |
| lrc204 | 819.319 | 3,083 | 3 | 3 | 3 | 3 | 108 | 41.0 |
| lrc205 | 1,220.710 | 5,258 | 4 | 4 | 4 | 2 | 76 | 34.6 |
| lrc206 | 1,175.625 | 2,474 | 3 | 3 | 4 | 4 | 93 | 39.9 |
| lrc207 | 1,108.881 | 2,517 | 3 | 3 | 3 | 3 | 94 | 41.3 |
| lrc208 | 885.852 | 4,444 | 3 | 3 | 3 | 3 | 110 | 42.7 |

Table 5: Detailed computational results size class 200

| Instance | Objective function value | Iteration where best solution was found | No. routes | No. LTC routes | No. subroutes | No. PTLs used | Running time ALNS with constant-time test (secs.) | Ratio running time linear / constant |
|-----------|--------------------------|---|------------|----------------|---------------|---------------|---|--------------------------------------|
| lc1_2_1 | 3,093.290 | 6,607 | 21 | 21 | 21 | 14 | 148 | 11.1 |
| lc1_2_2 | 3,611.993 | 9,872 | 21 | 21 | 22 | 15 | 152 | 11.3 |
| lc1_2_3 | 3,436.951 | 9,878 | 20 | 20 | 21 | 13 | 155 | 11.3 |
| lc1_2_4 | 3,210.840 | 8,006 | 20 | 20 | 21 | 12 | 169 | 12.2 |
| lc1_2_5 | 3,141.552 | 9,966 | 21 | 21 | 22 | 15 | 156 | 11.3 |
| lc1_2_6 | 3,169.339 | 9,874 | 21 | 21 | 21 | 16 | 156 | 11.5 |
| lc1_2_7 | 3,145.988 | 6,885 | 20 | 20 | 20 | 12 | 158 | 11.6 |
| lc1_2_8 | 3,103.838 | 5,116 | 21 | 21 | 21 | 14 | 157 | 11.7 |
| lc1_2_9 | 3,449.571 | 9,980 | 21 | 21 | 21 | 14 | 162 | 11.7 |
| lc1_2_10 | 3,399.748 | 9,111 | 21 | 21 | 22 | 13 | 161 | 11.8 |
| lc2_2_1 | 2,175.912 | 5,701 | 8 | 8 | 9 | 8 | 240 | 36.2 |
| lc2_2_2 | 2,170.040 | 5,847 | 7 | 7 | 7 | 6 | 273 | 37.8 |
| lc2_2_3 | 2,013.353 | 2,762 | 7 | 7 | 7 | 6 | 300 | 38.2 |
| lc2_2_4 | 1,890.106 | 9,606 | 6 | 6 | 6 | 5 | 350 | 39.2 |
| lc2_2_5 | 1,988.849 | 9,398 | 6 | 6 | 6 | 6 | 251 | 38.9 |
| lc2_2_6 | 2,078.159 | 5,568 | 6 | 6 | 6 | 4 | 259 | 38.0 |
| lc2_2_7 | 1,997.970 | 4,334 | 6 | 6 | 7 | 6 | 266 | 38.5 |
| lc2_2_8 | 2,051.189 | 5,528 | 7 | 7 | 7 | 6 | 285 | 40.7 |
| lc2_2_9 | 2,030.965 | 3,779 | 7 | 7 | 7 | 6 | 284 | 39.2 |
| lc2_2_10 | 1,990.755 | 4,238 | 6 | 6 | 6 | 5 | 293 | 40.1 |
| lr1_2_1 | 3,264.972 | 7,485 | 10 | 10 | 10 | 8 | 175 | 18.8 |
| lr1_2_2 | 3,216.481 | 7,291 | 10 | 10 | 12 | 9 | 187 | 20.3 |
| lr1_2_3 | 2,951.496 | 8,292 | 10 | 10 | 14 | 11 | 195 | 21.9 |
| lr1_2_4 | 2,474.494 | 4,992 | 9 | 9 | 11 | 7 | 229 | 25.8 |
| lr1_2_5 | 3,256.285 | 9,645 | 11 | 11 | 11 | 7 | 185 | 19.1 |
| lr1_2_6 | 3,068.128 | 4,497 | 11 | 11 | 16 | 11 | 203 | 22.1 |
| lr1_2_7 | 2,677.863 | 8,742 | 10 | 10 | 11 | 8 | 198 | 22.3 |
| lr1_2_8 | 2,475.545 | 9,575 | 8 | 8 | 8 | 7 | 219 | 26.2 |
| lr1_2_9 | 3,091.304 | 8,916 | 10 | 10 | 11 | 9 | 186 | 20.4 |
| lr1_2_10 | 2,757.828 | 7,195 | 10 | 10 | 15 | 11 | 198 | 21.9 |
| lr2_2_1 | 3,479.459 | 8,296 | 5 | 5 | 5 | 3 | 375 | 69.4 |
| lr2_2_2 | 3,295.378 | 9,884 | 4 | 4 | 8 | 8 | 524 | 83.2 |
| lr2_2_3 | 3,052.134 | 8,236 | 4 | 4 | 15 | 13 | 617 | 84.6 |
| lr2_2_4 | 2,325.207 | 7,696 | 4 | 4 | 8 | 8 | 840 | 96.6 |
| lr2_2_5 | 3,311.055 | 8,910 | 4 | 4 | 8 | 7 | 500 | 87.7 |
| lr2_2_6 | 3,279.246 | 9,949 | 4 | 4 | 10 | 8 | 570 | 85.3 |
| lr2_2_7 | 2,712.065 | 9,178 | 3 | 3 | 6 | 6 | 714 | 100.9 |
| lr2_2_8 | 1,962.325 | 294 | 4 | 4 | 6 | 5 | 987 | 120.6 |
| lr2_2_9 | 3,253.515 | 7,558 | 4 | 4 | 7 | 6 | 522 | 89.4 |
| lr2_2_10 | 2,756.037 | 7,993 | 4 | 4 | 7 | 5 | 517 | 85.0 |
| lrc1_2_1 | 2,892.339 | 6,809 | 12 | 12 | 13 | 13 | 175 | 17.8 |
| lrc1_2_2 | 2,794.063 | 6,133 | 11 | 11 | 13 | 12 | 178 | 19.8 |
| lrc1_2_3 | 2,663.717 | 8,898 | 9 | 9 | 13 | 11 | 203 | 22.8 |
| lrc1_2_4 | 2,442.196 | 9,414 | 9 | 9 | 11 | 10 | 230 | 26.8 |
| lrc1_2_5 | 3,102.716 | 9,166 | 11 | 11 | 11 | 9 | 180 | 18.9 |
| lrc1_2_6 | 2,812.237 | 8,063 | 12 | 12 | 15 | 12 | 176 | 18.9 |
| lrc1_2_7 | 2,826.656 | 9,688 | 11 | 11 | 11 | 7 | 187 | 20.3 |
| lrc1_2_8 | 2,635.937 | 9,878 | 10 | 10 | 10 | 7 | 184 | 20.2 |
| lrc1_2_9 | 2,593.870 | 9,457 | 10 | 10 | 13 | 10 | 185 | 20.7 |
| lrc1_2_10 | 2,553.282 | 9,247 | 10 | 10 | 12 | 9 | 206 | 22.1 |
| lrc2_2_1 | 2,861.418 | 8,416 | 6 | 6 | 12 | 9 | 314 | 53.3 |
| lrc2_2_2 | 2,577.954 | 6,528 | 6 | 6 | 10 | 9 | 364 | 54.5 |
| lrc2_2_3 | 2,393.727 | 212 | 5 | 5 | 7 | 6 | 500 | 72.4 |
| lrc2_2_4 | 2,241.657 | 2,418 | 4 | 4 | 7 | 7 | 657 | 82.3 |
| lrc2_2_5 | 2,859.009 | 8,398 | 5 | 5 | 18 | 17 | 466 | 72.1 |
| lrc2_2_6 | 2,668.582 | 9,234 | 5 | 5 | 12 | 9 | 439 | 66.1 |
| lrc2_2_7 | 2,544.252 | 9,759 | 5 | 5 | 9 | 9 | 434 | 61.1 |
| lrc2_2_8 | 2,419.297 | 9,696 | 4 | 4 | 8 | 8 | 526 | 73.9 |
| lrc2_2_9 | 2,272.824 | 6,722 | 4 | 4 | 5 | 5 | 527 | 75.0 |
| lrc2_2_10 | 2,181.054 | 7,304 | 4 | 4 | 11 | 11 | 593 | 75.8 |

Table 6: Detailed computational results size class 400

| Instance | Objective function value | Iteration where best solution was found | No. routes | No. LTC routes | No. subroutes | No. PTLs used | Running time ALNS with constant-time test (secs.) | Ratio running time linear / constant |
|-----------|--------------------------|---|------------|----------------|---------------|---------------|---|--------------------------------------|
| lc1_4_1 | 8,157.242 | 9,894 | 40 | 40 | 42 | 29 | 468 | 10.9 |
| lc1_4_2 | 8,326.408 | 9,606 | 39 | 39 | 39 | 24 | 490 | 11.4 |
| lc1_4_3 | 8,181.272 | 9,487 | 38 | 38 | 40 | 24 | 511 | 11.9 |
| lc1_4_4 | 7,998.630 | 9,611 | 36 | 36 | 35 | 23 | 521 | 12.3 |
| lc1_4_5 | 8,452.623 | 8,748 | 42 | 42 | 42 | 28 | 472 | 11.0 |
| lc1_4_6 | 7,965.205 | 7,750 | 40 | 40 | 40 | 25 | 482 | 11.2 |
| lc1_4_7 | 8,101.512 | 9,001 | 40 | 40 | 42 | 28 | 491 | 11.3 |
| lc1_4_8 | 8,065.265 | 9,911 | 39 | 39 | 39 | 26 | 483 | 11.1 |
| lc1_4_9 | 8,606.076 | 9,197 | 38 | 38 | 39 | 27 | 506 | 11.4 |
| lc1_4_10 | 8,063.120 | 9,998 | 37 | 37 | 39 | 19 | 516 | 11.5 |
| lc2_4_1 | 4,634.553 | 8,424 | 13 | 13 | 13 | 9 | 735 | 33.9 |
| lc2_4_2 | 4,718.921 | 9,331 | 13 | 13 | 14 | 9 | 830 | 34.9 |
| lc2_4_3 | 4,732.656 | 5,581 | 13 | 13 | 15 | 13 | 943 | 35.3 |
| lc2_4_4 | 4,832.596 | 5,247 | 13 | 13 | 15 | 12 | 1,048 | 35.8 |
| lc2_4_5 | 4,793.259 | 9,977 | 14 | 14 | 14 | 13 | 760 | 34.8 |
| lc2_4_6 | 4,330.294 | 8,316 | 12 | 12 | 13 | 11 | 810 | 36.2 |
| lc2_4_7 | 4,455.024 | 9,622 | 13 | 13 | 14 | 14 | 830 | 36.6 |
| lc2_4_8 | 4,542.666 | 8,399 | 13 | 13 | 13 | 11 | 843 | 38.0 |
| lc2_4_9 | 4,796.374 | 9,561 | 13 | 13 | 15 | 12 | 863 | 36.5 |
| lc2_4_10 | 4,542.475 | 9,835 | 13 | 13 | 15 | 13 | 880 | 37.8 |
| lr1_4_1 | 7,600.421 | 9,901 | 21 | 21 | 24 | 20 | 538 | 18.9 |
| lr1_4_2 | 7,122.111 | 8,661 | 20 | 20 | 20 | 19 | 589 | 21.4 |
| lr1_4_3 | 6,293.120 | 9,987 | 17 | 17 | 19 | 18 | 650 | 24.5 |
| lr1_4_4 | 5,453.027 | 9,352 | 15 | 15 | 17 | 13 | 756 | 28.9 |
| lr1_4_5 | 7,197.859 | 6,366 | 20 | 20 | 26 | 20 | 544 | 22.8 |
| lr1_4_6 | 6,924.309 | 9,903 | 18 | 18 | 20 | 16 | 614 | 22.3 |
| lr1_4_7 | 6,248.194 | 6,972 | 17 | 17 | 23 | 19 | 660 | 26.0 |
| lr1_4_8 | 5,421.693 | 7,130 | 15 | 15 | 18 | 15 | 763 | 30.5 |
| lr1_4_9 | 6,931.926 | 9,867 | 19 | 19 | 20 | 17 | 574 | 20.8 |
| lr1_4_10 | 6,615.693 | 6,418 | 18 | 18 | 23 | 20 | 616 | 22.9 |
| lr2_4_1 | 8,661.925 | 9,136 | 9 | 9 | 15 | 12 | 1,198 | 76.2 |
| lr2_4_2 | 7,568.071 | 9,942 | 8 | 8 | 19 | 16 | 1,528 | 83.5 |
| lr2_4_3 | 6,753.730 | 4,193 | 8 | 8 | 25 | 22 | 1,981 | 97.3 |
| lr2_4_4 | 5,311.441 | 9,961 | 6 | 6 | 18 | 16 | 2,868 | 122.0 |
| lr2_4_5 | 7,686.022 | 8,490 | 8 | 8 | 17 | 16 | 1,400 | 93.2 |
| lr2_4_6 | 6,735.141 | 4,596 | 8 | 8 | 20 | 17 | 1,678 | 93.4 |
| lr2_4_7 | 6,054.436 | 8,945 | 6 | 6 | 15 | 13 | 2,238 | 114.8 |
| lr2_4_8 | 5,226.712 | 8,953 | 5 | 5 | 13 | 12 | 3,432 | 142.5 |
| lr2_4_9 | 7,196.682 | 9,888 | 8 | 8 | 24 | 22 | 1,508 | 94.0 |
| lr2_4_10 | 6,732.745 | 9,932 | 8 | 8 | 24 | 22 | 1,599 | 93.7 |
| lrc1_4_1 | 7,147.617 | 9,910 | 24 | 24 | 24 | 21 | 493 | 17.9 |
| lrc1_4_2 | 6,410.331 | 6,771 | 21 | 21 | 24 | 19 | 546 | 20.6 |
| lrc1_4_3 | 6,093.626 | 8,966 | 19 | 19 | 24 | 19 | 580 | 23.4 |
| lrc1_4_4 | 5,340.786 | 9,989 | 15 | 15 | 17 | 14 | 681 | 29.1 |
| lrc1_4_5 | 7,131.683 | 9,877 | 23 | 23 | 27 | 20 | 512 | 17.8 |
| lrc1_4_6 | 6,542.183 | 9,589 | 22 | 22 | 24 | 16 | 515 | 19.3 |
| lrc1_4_7 | 6,635.200 | 6,959 | 21 | 21 | 25 | 18 | 547 | 19.6 |
| lrc1_4_8 | 6,390.169 | 9,938 | 22 | 22 | 23 | 21 | 550 | 19.7 |
| lrc1_4_9 | 6,440.317 | 7,864 | 20 | 20 | 21 | 16 | 550 | 20.7 |
| lrc1_4_10 | 5,877.081 | 6,906 | 19 | 19 | 20 | 15 | 565 | 22.2 |
| lrc2_4_1 | 6,755.846 | 9,929 | 12 | 12 | 27 | 23 | 940 | 59.0 |
| lrc2_4_2 | 6,372.873 | 5,437 | 10 | 10 | 22 | 20 | 1,184 | 69.9 |
| lrc2_4_3 | 5,277.173 | 8,829 | 8 | 8 | 13 | 13 | 1,489 | 81.4 |
| lrc2_4_4 | 4,656.752 | 9,893 | 7 | 7 | 24 | 22 | 2,434 | 108.1 |
| lrc2_4_5 | 6,432.971 | 9,520 | 10 | 10 | 20 | 19 | 1,079 | 65.4 |
| lrc2_4_6 | 6,318.373 | 3,927 | 10 | 10 | 27 | 21 | 1,094 | 67.4 |
| lrc2_4_7 | 5,902.167 | 9,607 | 9 | 9 | 21 | 17 | 1,230 | 73.5 |
| lrc2_4_8 | 5,560.296 | 9,767 | 8 | 8 | 21 | 18 | 1,424 | 81.6 |
| lrc2_4_9 | 5,289.767 | 9,968 | 8 | 8 | 15 | 15 | 1,467 | 85.1 |
| lrc2_4_10 | 5,264.808 | 9,125 | 8 | 8 | 25 | 22 | 1,598 | 88.1 |

Table 7: Detailed computational results size class 600

| Instance | Objective function value | Iteration where best solution was found | No. routes | No. LTC routes | No. subroutes | No. PTLs used | Running time ALNS with constant-time test (secs.) |
|-----------|--------------------------|---|------------|----------------|---------------|---------------|---|
| lc1_6_1 | 16,466.091 | 9,762 | 62 | 62 | 62 | 43 | 778 |
| lc1_6_2 | 16,521.563 | 9,763 | 61 | 61 | 61 | 37 | 799 |
| lc1_6_3 | 16,629.909 | 8,754 | 58 | 58 | 58 | 38 | 826 |
| lc1_6_4 | 16,045.254 | 9,958 | 57 | 57 | 55 | 34 | 857 |
| lc1_6_5 | 17,075.448 | 9,886 | 64 | 64 | 61 | 45 | 789 |
| lc1_6_6 | 17,092.862 | 9,858 | 64 | 64 | 65 | 44 | 820 |
| lc1_6_7 | 16,287.753 | 9,772 | 61 | 61 | 59 | 36 | 818 |
| lc1_6_8 | 17,413.197 | 9,934 | 62 | 62 | 63 | 43 | 815 |
| lc1_6_9 | 18,047.010 | 9,281 | 62 | 62 | 63 | 35 | 834 |
| lc1_6_10 | 17,327.674 | 8,262 | 59 | 59 | 61 | 41 | 868 |
| lc2_6_1 | 10,207.809 | 9,964 | 22 | 22 | 25 | 22 | 1,092 |
| lc2_6_2 | 9,563.355 | 8,330 | 19 | 19 | 21 | 17 | 1,204 |
| lc2_6_3 | 9,454.697 | 9,235 | 20 | 20 | 26 | 22 | 1,352 |
| lc2_6_4 | 8,949.587 | 9,736 | 20 | 20 | 20 | 19 | 1,505 |
| lc2_6_5 | 9,761.939 | 9,942 | 20 | 20 | 23 | 22 | 1,137 |
| lc2_6_6 | 9,809.889 | 9,927 | 22 | 22 | 32 | 28 | 1,189 |
| lc2_6_7 | 9,437.012 | 9,493 | 21 | 21 | 25 | 23 | 1,216 |
| lc2_6_8 | 9,172.899 | 8,737 | 21 | 21 | 24 | 20 | 1,263 |
| lc2_6_9 | 9,478.359 | 9,545 | 21 | 21 | 24 | 23 | 1,263 |
| lc2_6_10 | 8,853.279 | 9,338 | 20 | 20 | 23 | 21 | 1,367 |
| lr1_6_1 | 16,971.485 | 9,266 | 31 | 31 | 47 | 37 | 1,010 |
| lr1_6_2 | 17,411.229 | 5,466 | 27 | 25 | 34 | 30 | 1,074 |
| lr1_6_3 | 14,309.551 | 9,961 | 24 | 24 | 31 | 28 | 1,229 |
| lr1_6_4 | 12,432.560 | 8,843 | 18 | 17 | 23 | 21 | 1,406 |
| lr1_6_5 | 16,143.579 | 7,255 | 28 | 28 | 37 | 34 | 1,016 |
| lr1_6_6 | 16,917.078 | 9,557 | 25 | 23 | 34 | 30 | 1,119 |
| lr1_6_7 | 13,302.893 | 9,981 | 23 | 23 | 28 | 26 | 1,215 |
| lr1_6_8 | 11,141.020 | 9,976 | 19 | 19 | 33 | 31 | 1,452 |
| lr1_6_9 | 15,254.064 | 9,497 | 28 | 28 | 36 | 30 | 1,021 |
| lr1_6_10 | 16,886.003 | 3,047 | 26 | 25 | 35 | 34 | 1,088 |
| lr2_6_1 | 18,358.684 | 8,917 | 12 | 12 | 42 | 34 | 2,216 |
| lr2_6_2 | 15,947.769 | 8,049 | 12 | 12 | 45 | 37 | 2,843 |
| lr2_6_3 | 14,063.910 | 7,062 | 10 | 10 | 41 | 37 | 3,651 |
| lr2_6_4 | 11,140.270 | 9,007 | 7 | 7 | 28 | 25 | 5,599 |
| lr2_6_5 | 17,091.142 | 7,637 | 11 | 11 | 60 | 48 | 2,596 |
| lr2_6_6 | 15,925.030 | 9,745 | 10 | 10 | 43 | 32 | 3,230 |
| lr2_6_7 | 13,519.317 | 9,566 | 9 | 9 | 45 | 38 | 4,105 |
| lr2_6_8 | 10,320.749 | 7,862 | 7 | 7 | 29 | 26 | 5,803 |
| lr2_6_9 | 15,666.296 | 9,915 | 10 | 10 | 37 | 29 | 2,819 |
| lr2_6_10 | 15,595.795 | 8,820 | 11 | 11 | 54 | 45 | 2,975 |
| lrc1_6_1 | 13,855.237 | 9,714 | 32 | 32 | 35 | 25 | 876 |
| lrc1_6_2 | 12,973.273 | 9,991 | 29 | 29 | 35 | 28 | 969 |
| lrc1_6_3 | 12,659.233 | 9,834 | 25 | 25 | 34 | 30 | 1,117 |
| lrc1_6_4 | 9,902.372 | 9,652 | 18 | 18 | 25 | 22 | 1,391 |
| lrc1_6_5 | 13,645.705 | 7,960 | 32 | 32 | 36 | 31 | 898 |
| lrc1_6_6 | 13,554.014 | 8,787 | 30 | 30 | 32 | 24 | 901 |
| lrc1_6_7 | 12,950.530 | 8,703 | 29 | 29 | 35 | 27 | 933 |
| lrc1_6_8 | 13,145.138 | 8,813 | 28 | 28 | 30 | 26 | 931 |
| lrc1_6_9 | 13,227.627 | 9,645 | 28 | 28 | 29 | 24 | 951 |
| lrc1_6_10 | 12,424.974 | 9,805 | 26 | 26 | 32 | 23 | 977 |
| lrc2_6_1 | 14,628.817 | 9,995 | 17 | 17 | 61 | 53 | 1,742 |
| lrc2_6_2 | 13,179.837 | 9,591 | 14 | 14 | 59 | 50 | 2,131 |
| lrc2_6_3 | 11,013.262 | 9,799 | 11 | 11 | 32 | 27 | 2,705 |
| lrc2_6_4 | 9,414.274 | 9,639 | 8 | 8 | 29 | 25 | 4,247 |
| lrc2_6_5 | 13,483.235 | 9,753 | 15 | 15 | 39 | 32 | 1,802 |
| lrc2_6_6 | 13,981.647 | 9,152 | 13 | 13 | 47 | 39 | 1,865 |
| lrc2_6_7 | 12,920.265 | 8,311 | 11 | 11 | 34 | 28 | 2,220 |
| lrc2_6_8 | 12,120.200 | 9,915 | 12 | 12 | 43 | 40 | 2,291 |
| lrc2_6_9 | 13,734.873 | 5,731 | 11 | 11 | 50 | 41 | 2,321 |
| lrc2_6_10 | 12,396.543 | 9,886 | 10 | 10 | 44 | 35 | 2,618 |

Table 8: Detailed computational results size class 800

| Instance | Objective function value | Iteration where best solution was found | No. routes | No. LTC routes | No. subroutes | No. PTLs used | Running time ALNS with constant-time test (secs.) |
|-----------------|---------------------------------|--|-------------------|-----------------------|----------------------|----------------------|--|
| lc181 | 29,779.459 | 9,751 | 84 | 84 | 84 | 58 | 1,081 |
| lc182 | 34,001.982 | 7,969 | 81 | 79 | 76 | 56 | 1,129 |
| lc183 | 31,269.812 | 8,397 | 78 | 78 | 79 | 53 | 1,148 |
| lc184 | 31,909.582 | 8,508 | 73 | 70 | 70 | 47 | 1,202 |
| lc185 | 32,090.767 | 9,405 | 87 | 87 | 84 | 55 | 1,136 |
| lc186 | 30,526.773 | 9,015 | 84 | 84 | 83 | 57 | 1,152 |
| lc187 | 30,846.858 | 8,924 | 82 | 82 | 80 | 51 | 1,123 |
| lc188 | 31,051.107 | 9,965 | 82 | 82 | 83 | 54 | 1,141 |
| lc189 | 31,594.426 | 9,834 | 79 | 79 | 75 | 50 | 1,148 |
| lc1810 | 31,669.092 | 9,775 | 79 | 79 | 79 | 52 | 1,147 |
| lc281 | 15,159.593 | 9,597 | 28 | 28 | 32 | 25 | 1,484 |
| lc282 | 16,926.027 | 9,467 | 30 | 30 | 48 | 39 | 1,628 |
| lc283 | 19,443.635 | 8,930 | 28 | 24 | 33 | 32 | 1,727 |
| lc284 | 17,848.417 | 9,571 | 27 | 25 | 32 | 26 | 1,922 |
| lc285 | 14,943.285 | 9,821 | 27 | 27 | 31 | 23 | 1,631 |
| lc286 | 15,451.011 | 9,651 | 29 | 29 | 43 | 34 | 1,766 |
| lc287 | 15,270.417 | 9,942 | 27 | 26 | 32 | 26 | 1,835 |
| lc288 | 14,743.008 | 9,902 | 28 | 28 | 31 | 28 | 1,823 |
| lc289 | 14,756.891 | 9,916 | 28 | 28 | 32 | 27 | 1,804 |
| lc2810 | 15,605.859 | 7,271 | 29 | 29 | 43 | 36 | 1,923 |
| lr181 | 41,949.098 | 9,001 | 38 | 27 | 48 | 42 | 1,431 |
| lr182 | 32,628.581 | 9,008 | 37 | 32 | 48 | 45 | 1,513 |
| lr183 | 30,878.496 | 9,914 | 29 | 23 | 39 | 35 | 1,686 |
| lr184 | 21,989.100 | 9,897 | 22 | 19 | 34 | 34 | 2,082 |
| lr185 | 32,183.407 | 9,916 | 39 | 36 | 57 | 48 | 1,406 |
| lr186 | 33,190.601 | 9,998 | 35 | 28 | 36 | 30 | 1,515 |
| lr187 | 25,346.191 | 9,803 | 29 | 27 | 35 | 33 | 1,684 |
| lr188 | 20,062.252 | 7,632 | 23 | 21 | 28 | 26 | 2,083 |
| lr189 | 34,384.747 | 8,366 | 37 | 31 | 48 | 43 | 1,406 |
| lr1810 | 31,676.044 | 8,330 | 34 | 27 | 41 | 35 | 1,471 |
| lr281 | 33,711.230 | 7,592 | 17 | 16 | 81 | 71 | 2,816 |
| lr282 | 28,988.585 | 9,681 | 13 | 13 | 59 | 55 | 3,589 |
| lr283 | 23,193.891 | 9,942 | 12 | 11 | 51 | 43 | 4,739 |
| lr284 | 18,490.427 | 9,455 | 8 | 8 | 34 | 31 | 7,834 |
| lr285 | 29,125.350 | 9,666 | 15 | 14 | 70 | 58 | 3,105 |
| lr286 | 24,148.337 | 9,663 | 14 | 14 | 58 | 47 | 3,967 |
| lr287 | 22,686.727 | 9,918 | 12 | 12 | 62 | 56 | 5,224 |
| lr288 | 17,010.234 | 9,518 | 7 | 7 | 30 | 26 | 7,905 |
| lr289 | 27,858.035 | 9,842 | 13 | 13 | 66 | 54 | 3,235 |
| lr2810 | 25,980.059 | 9,423 | 14 | 14 | 70 | 54 | 3,395 |
| lrc181 | 33,874.714 | 8,637 | 47 | 40 | 48 | 40 | 1,116 |
| lrc182 | 29,331.441 | 9,941 | 44 | 39 | 43 | 35 | 1,230 |
| lrc183 | 24,651.073 | 8,960 | 36 | 33 | 42 | 35 | 1,411 |
| lrc184 | 19,563.444 | 9,296 | 26 | 24 | 35 | 32 | 1,743 |
| lrc185 | 32,054.570 | 9,468 | 45 | 39 | 43 | 38 | 1,179 |
| lrc186 | 30,576.310 | 9,699 | 44 | 38 | 42 | 38 | 1,225 |
| lrc187 | 29,794.101 | 9,970 | 43 | 39 | 50 | 43 | 1,208 |
| lrc188 | 26,477.220 | 9,989 | 39 | 36 | 45 | 37 | 1,268 |
| lrc189 | 22,577.839 | 9,984 | 38 | 38 | 46 | 37 | 1,224 |
| lrc1810 | 24,871.446 | 9,263 | 37 | 34 | 39 | 35 | 1,258 |
| lrc281 | 25,832.234 | 6,887 | 21 | 20 | 75 | 59 | 2,040 |
| lrc282 | 24,480.590 | 9,191 | 17 | 14 | 51 | 50 | 2,551 |
| lrc283 | 19,833.313 | 9,929 | 16 | 15 | 53 | 44 | 3,133 |
| lrc284 | 14,549.916 | 9,218 | 12 | 12 | 35 | 32 | 4,923 |
| lrc285 | 23,944.816 | 7,576 | 19 | 19 | 103 | 79 | 2,408 |
| lrc286 | 24,865.120 | 8,616 | 17 | 15 | 87 | 76 | 2,401 |
| lrc287 | 22,520.066 | 6,938 | 16 | 16 | 76 | 64 | 2,543 |
| lrc288 | 21,990.194 | 8,369 | 16 | 15 | 84 | 71 | 2,848 |
| lrc289 | 19,734.446 | 9,866 | 15 | 15 | 77 | 68 | 2,864 |
| lrc2810 | 19,399.705 | 9,992 | 13 | 13 | 39 | 39 | 3,223 |

Table 9: Detailed computational results size class 1,000

| Instance | Objective function value | Iteration where best solution was found | No. routes | No. LTC routes | No. subroutes | No. PTLs used | Running time ALNS with constant-time test (secs.) |
|----------|--------------------------|---|------------|----------------|---------------|---------------|---|
| lc1101 | 65,073.718 | 9,695 | 108 | 94 | 89 | 64 | 1,310 |
| lc1102 | 67,649.754 | 7,571 | 103 | 90 | 93 | 70 | 1,369 |
| lc1103 | 55,315.955 | 9,803 | 95 | 90 | 93 | 69 | 1,516 |
| lc1104 | 53,303.065 | 7,549 | 92 | 88 | 88 | 63 | 1,521 |
| lc1105 | 61,452.807 | 9,950 | 105 | 95 | 91 | 65 | 1,426 |
| lc1106 | 59,887.447 | 9,518 | 110 | 103 | 96 | 68 | 1,465 |
| lc1107 | 55,160.075 | 9,995 | 105 | 102 | 98 | 72 | 1,494 |
| lc1108 | 63,357.039 | 9,999 | 105 | 96 | 94 | 69 | 1,512 |
| lc1109 | 61,419.643 | 5,994 | 108 | 103 | 108 | 78 | 1,557 |
| lc11010 | 54,890.505 | 9,554 | 97 | 93 | 90 | 62 | 1,597 |
| lc2101 | 22,527.689 | 9,988 | 35 | 34 | 40 | 33 | 1,956 |
| lc2102 | 23,792.805 | 7,876 | 36 | 35 | 51 | 46 | 2,169 |
| lc2103 | 23,031.139 | 9,819 | 34 | 34 | 48 | 42 | 2,321 |
| lc2104 | 25,118.021 | 9,545 | 34 | 32 | 46 | 39 | 2,581 |
| lc2105 | 24,710.473 | 9,475 | 37 | 35 | 41 | 37 | 2,073 |
| lc2106 | 23,694.035 | 9,704 | 36 | 35 | 46 | 40 | 1,995 |
| lc2107 | 25,279.518 | 9,221 | 38 | 36 | 58 | 52 | 2,179 |
| lc2108 | 25,028.226 | 7,300 | 37 | 35 | 49 | 42 | 2,237 |
| lc2109 | 25,348.545 | 8,748 | 37 | 36 | 55 | 50 | 2,259 |
| lc21010 | 21,934.962 | 9,304 | 36 | 36 | 44 | 35 | 2,185 |
| lr1101 | 59,691.009 | 9,975 | 50 | 35 | 61 | 51 | 1,822 |
| lr1102 | 48,538.500 | 9,997 | 45 | 39 | 64 | 53 | 1,909 |
| lr1103 | 44,318.348 | 9,671 | 37 | 29 | 53 | 48 | 2,215 |
| lr1104 | 32,194.887 | 9,130 | 27 | 22 | 37 | 34 | 2,739 |
| lr1105 | 56,018.817 | 9,783 | 49 | 40 | 68 | 55 | 1,837 |
| lr1106 | 45,459.665 | 9,881 | 42 | 35 | 49 | 43 | 2,026 |
| lr1107 | 41,921.754 | 9,202 | 35 | 26 | 48 | 43 | 2,305 |
| lr1108 | 35,527.347 | 9,499 | 28 | 23 | 48 | 46 | 2,800 |
| lr1109 | 48,613.159 | 9,857 | 46 | 39 | 53 | 50 | 1,888 |
| lr11010 | 47,691.898 | 8,048 | 41 | 32 | 59 | 52 | 1,968 |
| lr2101 | 51,690.254 | 9,868 | 21 | 17 | 99 | 83 | 3,275 |
| lr2102 | 45,995.484 | 9,789 | 18 | 16 | 70 | 64 | 4,190 |
| lr2103 | 36,607.173 | 9,896 | 16 | 13 | 43 | 42 | 5,416 |
| lr2104 | 27,180.136 | 9,254 | 12 | 12 | 59 | 54 | 9,069 |
| lr2105 | 46,941.618 | 9,595 | 18 | 15 | 77 | 67 | 3,775 |
| lr2106 | 41,616.661 | 8,730 | 17 | 15 | 70 | 61 | 4,741 |
| lr2107 | 34,561.985 | 9,174 | 14 | 12 | 64 | 59 | 6,095 |
| lr2108 | 26,083.772 | 9,832 | 10 | 10 | 50 | 46 | 10,024 |
| lr2109 | 43,679.631 | 8,723 | 18 | 15 | 86 | 72 | 4,052 |
| lr21010 | 41,890.994 | 9,910 | 16 | 13 | 75 | 72 | 4,336 |
| lrc1101 | 56,643.406 | 9,370 | 59 | 44 | 57 | 42 | 1,698 |
| lrc1102 | 48,314.071 | 9,827 | 56 | 48 | 71 | 60 | 1,814 |
| lrc1103 | 39,926.935 | 9,798 | 46 | 39 | 51 | 45 | 2,051 |
| lrc1104 | 32,640.504 | 7,727 | 33 | 29 | 50 | 45 | 2,665 |
| lrc1105 | 54,919.086 | 9,631 | 58 | 47 | 58 | 44 | 1,717 |
| lrc1106 | 55,289.112 | 9,903 | 55 | 39 | 55 | 50 | 1,708 |
| lrc1107 | 46,741.471 | 9,827 | 50 | 39 | 51 | 42 | 1,758 |
| lrc1108 | 50,108.697 | 9,452 | 51 | 39 | 61 | 52 | 1,790 |
| lrc1109 | 46,641.948 | 9,651 | 50 | 39 | 52 | 45 | 1,803 |
| lrc11010 | 39,637.235 | 9,998 | 47 | 42 | 59 | 51 | 1,871 |
| lrc2101 | 39,684.964 | 9,595 | 24 | 20 | 101 | 86 | 2,937 |
| lrc2102 | 37,322.263 | 9,847 | 23 | 20 | 102 | 81 | 3,435 |
| lrc2103 | 30,658.094 | 9,759 | 19 | 18 | 80 | 69 | 4,325 |
| lrc2104 | 25,217.808 | 9,934 | 14 | 13 | 56 | 50 | 6,626 |
| lrc2105 | 38,071.655 | 9,897 | 21 | 16 | 88 | 74 | 3,442 |
| lrc2106 | 33,867.241 | 9,521 | 21 | 21 | 118 | 97 | 3,379 |
| lrc2107 | 33,795.104 | 9,081 | 18 | 16 | 83 | 73 | 3,720 |
| lrc21010 | 33,794.739 | 8,358 | 18 | 16 | 85 | 71 | 4,416 |