Large neighbourhood search with adaptive guided ejection search for the pickup and delivery problem with time windows

Timothy Curtois[1], Dario Landa-Silva[1], Yi Qu[1,2], Wasakorn Laesanklang[1,3,4]

[1]ASAP Research Group, School of Computer Science, The University of Nottingham, Nottingham, UK
[2]Newcastle Business School, Northumbria University, Newcastle-upon-Tyne, UK
[3]Centre of Excellence in Mathematics, CHE, Bangkok, Thailand
[4]Department of Mathematics, Faculty of Science, Mahidol University, Bangkok, Thailand
{tim.curtois@nottingham.ac.uk, dario.landasilva@nottingham.ac.uk, yi.qu@northumbria.ac.uk, wasakorn.lae@mahidol.ac.th}

Abstract

An effective and fast hybrid metaheuristic is proposed for solving the pickup and delivery problem with time windows. The proposed approach combines local search, large neighbourhood search and guided ejection search in a novel way to exploit the benefits of each method. The local search component uses a novel neighbourhood operator. A streamlined implementation of large neighbourhood search is used to achieve an effective balance between intensification and diversification. The adaptive ejection chain component perturbs the solution and uses increased or decreased computation time according to the progress of the search. While the local search and large neighbourhood search focus on minimising travel distance, the adaptive ejection chain seeks to reduce the number of routes. The proposed algorithm design results in an effective and fast solution method that finds a large number of new best known solutions on a well-known benchmark data set. Experiments are also performed to analyse the benefits of the components and heuristics and their combined use in order to achieve a better understanding of how to better tackle the subject problem.

1 Introduction

The pickup and delivery problem (PDP) is a vehicle routing problem in which customers are paired together and a pair must be serviced by the same vehicle [23]. In other words, a load must be collected from one location and delivered to another location by a single vehicle. Clearly there are also ordering or precedence constraints to ensure that the collection site is visited before the delivery site. If there are time windows during which the customers must be visited then the problem is known as PDPTW (pickup and delivery problem with time windows) [6]. The problem commonly arises in real-world logistics and solution methodologies have significant practical application. As such, a large number of techniques have been developed for PDPTW. These include approaches based on exact methods as well as heuristics. The exact based methods have advantages such as solving to provable optimality or providing bounds. They also tend to perform very well on smaller and medium sized instances. The significant disadvantage with these methods though is that they sometimes perform poorly on larger difficult instances. Heuristic methods on the other hand tend to scale well and are more robust for larger instances but are easily outperformed on smaller instances. It could also be argued that some heuristic methods are easier to develop and maintain and to adapt to new problem requirements. These could be the reasons that the majority of commercial vehicle routing software packages use heuristic based methods [9].

Most of the exact methods proposed for the PDPTW are versions of branch and cut and/or branch and price [3,18]. Branch and price is a branch and bound approach where each node in the branch and bound tree is solved using column generation. For PDPTW, the nodes are linear programming, set

partitioning formulations of the problem and the columns (variables) represent possible routes. Generating the new columns, known as the pricing problem, could be solved by a variety of exact and heuristic methods. Most approaches use a dynamic programming method applied to a shortest path type formulation. Solving the pricing problem efficiently is key to a successful approach because this is where most of the computation time is used. Very efficient pricing problem solving can be achieved though through the use of heuristics and problem structure exploitation. Other significant speed ups and algorithm improvements can often be achieved through other ideas such as branching strategies, stabilisation, column management, approximations and other heuristics.

One of the earliest applications of branch and price to PDPTW is given by Dumas et al [6] although it was clearly limited by the computing hardware available at the time. A branch and price method published later in the decade by Savelsbergh and Sol [22] was already able to solve larger instances of the problem in practical computation times. One of the most recent examples of branch and price applied to PDP is from Venkateshan and Mathur [25] and an example of a column generation based heuristic applied to PDPTW is given by Xu et al [26].

An alternative exact method is branch and cut. Branch and cut differs from branch and price in that a different formulation is used in which all the variables are present at the start rather than being dynamically generated. New constraints are generated at the nodes of the branch and bound tree. These cuts aim to accelerate the discovery of the optimal integer solution. For PDPTW and other vehicle routing problems different families of cuts have been proposed. Examples of branch and cut methods applied to other pickup and delivery problems include [12, 21]. A recent paper by Ropke and Cordeau [19] presents a branch and cut and price method and test it on one of the most commonly used sets of benchmark instances by Li and Lim [11]. The results show that although the smaller instances can be solved to optimality very efficiently, the larger instances are still out of reach for exact methods.

Although there are several examples of exact methods for PDP, the majority of publications are on heuristic methods and in particular metaheuristics. Many of these approaches use a variant of neighbourhood search. Due to the pairing and precedence constraints present in PDP but not present in other variants of vehicle routing problems, there is less choice of neighbourhood operators available for the problem. There are however three operators which were employed in the earlier metaheuristics. The first is to simply move a pickup and delivery pair from one route to another. The second is to swap a pair of pickups and deliveries between two routes. The third is to move the pickup and delivery within a route. Li and Lim [11] and Nanry and Barnes [17] both present metaheuristics built around these local search operators. These simpler metaheuristics were later outperformed though by methods based on Large Neighbourhood Search (LNS). As the name implies, LNS uses much larger neighbourhoods and searches them using heuristic and/or exact methods. The motivation is that larger neighbourhoods should provide better local optima. The disadvantage is that they can be slower to search due to their increased size. One of the first examples of LNS applied to PDP is from Bent and Van Hentenryck [2]. They explore large neighbourhoods using a branch and bound method. Ropke and Pisinger [20] then published an alternative LNS which uses a simpler heuristic method for creating and searching large neighbourhoods. Their heuristic is based on the “disrupt and repair” or “ruin and recreate” heuristics. Customers are iteratively removed from solution routes using heuristics such as similarity measures and then re-inserted using heuristics such as regret assignment or greedy assignment. The parameters for these heuristics are dynamically adapted based on their success rate. Their LNS produced many new best known solutions on the Li and Lim benchmark instances. Another paper that has achieved notable results on these benchmark instances is the Guided Ejection Search of Nagata and Kobayashi [15]. The algorithm only aims to optimise the single objective of

reducing the number of vehicles used but does so very effectively. It is a relatively simple but effective iterative heuristic that randomly adjusts the solution using customer swaps and moves. At each iteration, attempts are made to insert heuristically selected customers from removed routes. They later adapted this method into an evolutionary approach to again further improve their results with respect to the full objective function [16]. There are also several publications detailing metaheuristics applied to specific variants of PDP. For example, PDP with transfers [13], PDP with LIFO loading [5], single vehicle PDP [8, 10] and dynamic PDP [7]. For further reading on PDP there are also several survey and overview papers [1, 3, 18, 23].

From the above review of related literature, it is clear that PDPTW is a well-studied variant of the vehicle routing problem. A variety of exact and heuristic methods have been proposed and this has helped to advance our knowledge into how to tackle this difficult combinatorial optimization problem. Nevertheless, it is also clear that large instances of PDPTW remain a difficult challenge to state-of-the-art techniques. In this paper, an effective and efficient heuristic method is proposed which uses several tailored neighbourhood moves within a streamlined version of adaptive large neighbourhood search [20] also incorporating guided ejection search [15]. The proposed technique is not only competitive with state-of-the-art methods but it produces a number of new best known solutions for the well-known Li and Lim benchmark instances [11]. Moreover, this paper also makes a contribution to increase our understanding of which combination of search mechanisms can result in a highly effective and fast hybrid metaheuristic algorithm capable of solving large instances of the PDPTW. Experimental results also show that each of the components plays an essential role to make the overall algorithm perform very well over a wide range of problem sizes.

In the next section we provide the problem definition for the benchmark PDPTW instances tested on. Section 3 introduces the algorithm and Section 4 details the computational results. Finally we present some conclusions and suggested future research in Section 5.

2 Problem Definition

A solution to the pickup and delivery routing problem with time windows or PDPTW is a set of routes for a fleet of vehicles. Each route is executed by one vehicle and consists of a sequence of pickups and deliveries at customers' locations. Each transportation request is a pickup and delivery pair which must be executed in that order by the same vehicle while satisfying the vehicle capacity and the given time windows. The goal is to minimise the number of routes hence the number of vehicles needed and to minimise the total travelled distance.

Parameters:

M set of customers 1...m

L set of locations 0,1…m where 0 is the depot and 1…m are the customers

P set of pickup customers

D set of delivery customers

The intersection of P and D is the empty set ($P \cap D = \emptyset$) and the union of P and D is M ($P \cup D = M$). Each pickup $p_i \in$ P is associated with a corresponding delivery $d_i \in$ D. Let:

t_{ij} the travel time between locations i and j

d_{ij} the distance between locations i and j

s_i the service duration for location i

e_i the earliest time at which the service at location i can start

l_i the latest time at which the service at location i it must start

A service duration and service window have been included for the depot to make the model tidier but in the test instances the service duration for the depot is zero and the service window for the depot is unbounded. This is done in previously published models also, for example: [15].

Constraints:

A route is a sequence of locations visited by a vehicle. A vehicle must start at a depot, visit at least two customers (corresponding to a pickup and a delivery) and return to the depot. A route of length n is therefore denoted by $\langle v_0, v_1 \dots v_n, v_{n+1} \rangle$ where v_0 and v_{n+1} are the depot and visits $v_1 \dots v_n$ are customers. If a route contains a pickup p_i then it must also contain its corresponding delivery d_i (and vice-versa) and p_i must precede d_i in the sequence. These are the pairing and precedence constraints respectively. Each pickup p_i has a nonnegative demand q_i and the corresponding delivery d_i has the demand $-q_i$.

The current total load c_{v_i} carried by a vehicle v at a visit i where $i \geq 1$ is

$$c_{v_i} = \sum_{j=1}^{i} q_{v_j} \quad (1)$$

All vehicles have an identical capacity Q and at all visits in a route the total carried load must not exceed the vehicle capacity

$$c_{v_i} \leq Q \quad \forall v_i \in \{1 \dots n\} \quad (2)$$

The begin time b_v for each visit's service in the route is calculated as

$$b_{v_0} = 0$$
$$b_{v_i} = \max\{b_{v_{i-1}} + s_{v_{i-1}} + t_{v_{i-1}v_i}, e_{v_i}\} \quad \forall i \in \{1 \dots n\} \quad (3)$$

In a route the services must begin before or at a location's latest service start time

$$b_{v_i} \leq l_{v_i} \quad \forall v_i \in \{0 \dots n\} \quad (4)$$

A solution to this PDPTW described above is set of feasible routes which together service all customers exactly once according to the conditions established in the formulation.

Objectives:

The primary objective is to minimise the number of routes, hence the number of vehicles, in the solution. To compare solutions which have the same number of routes a secondary objective is commonly used. This secondary objective is to minimise the total distance of all routes where distance of a route of length n is

$$\sum_{i=0}^{n} d_{v_i, v_{i+1}} \quad (5)$$

3 Hybrid Large Neighbourhood Search and Guided Ejection Search

As discussed in the introduction, a number of methods have been proposed in the literature to tackle the PDPTW. The algorithmic design proposed here incorporates specialised neighbourhood operators to enhance the effectiveness of the local search, adaptive ejection search to reduce the number of routes and streamlined large neighbourhood search to enhance the efficiency of the search. The motivation behind the proposed algorithm design was to identify the essential mechanisms to reduce the number of routes and the total travelled distance and combine them into a streamlined yet effective and fast method.

The algorithm proposed here is a combination of three separate methods:

1) A local search which uses four tailored neighbourhood operators.
2) A simplified version of the Adaptive Large Neighbourhood Search (ALNS) of Ropke and Pisinger [20]. One of the simplifications is to remove the adaptive feature so this sub-routine will be referred to as LNS only.
3) A version of the Guided Ejection Search (GES) by Nagata and Kobayashi [15].

An outline of the overall algorithm approach is given in *Figure 1* and each of the steps is described in detail in the following subsections. The overall strategy is to perform an effective large neighbourhood search on a solution while exploiting guided ejection chain to reduce the number of routes or perturbing the current solution if reducing the number of routes is not possible. This balance between intensification and diversification results in an effective algorithm as shown by the experimental results presented later in the paper.

```
1. LocalSearch
2.
3. while (time remaining)
4. {
5.        try and reduce number of vehicles in best solution so far using GES
6.
7.        if vehicles not reduced then perturb best solution so far
8.
9.        LocalSearch
10.
11.       LNS
12.}
```

Figure 1 Overall Algorithm Outline

3.1 Local Search

The main purpose of the local search is to construct good-quality initial solutions quickly. To do this it uses four neighbourhood structures and the corresponding operators perform an exhaustive search until no further improvements can be made with respect to all neighbourhoods. The first three neighbourhood moves are used in [11, 17]. The neighbourhood moves are:

M1: Insert an unassigned pickup and delivery (PD) pair into an existing route or create a new route for the PD pair

M2: Un-assign an assigned PD pair and try and insert it into a different route or create a new route for the PD pair

M3: Un-assign a PD pair (*pd1*) from a route (*r1*), un-assign a PD pair (*pd2*) from a route (*r2*) and then try and insert *pd1* into route *r2* and *pd2* into route *r1*

M4: Un-assign a PD pair (*pd1*) from a route (*r1*), un-assign a PD pair (*pd2*) from a route (*r2*) and then try and insert *pd1* into route *r2* and *pd2* into a third route *r3*

Note that in all of the neighbourhoods above, when trying to insert a PD pair into a route the local search tries every possible position for the pickup and for each feasible position for the pickup, also tries every possible feasible position for the drop (position refers to order position in the route). This means that each neighbourhood is explored exhaustively and the best of all neighbour solutions is selected. Hence, this is part of the intensification mechanism in the proposed approach. We are not aware of the move M4 being used for PDPTW previously. The rationale behind this move is to have another mechanism for transferring PD pairs between routes. M3 does this while maintaining the same number of PD pairs in each of the two routes involved. In M4 the transfer ends up with two routes having a different number of PD pairs after the move.

A single neighbour operator is applied exhaustively until no more improving moves can be made using that operator. The next operator is then similarly applied exhaustively until no more improving moves are available, and then the next operator and so on. The order the operators are applied is M1 to M4 as in the list above. When operator M4 has been exhausted then the local search returns to operator M1. This process is repeated until there are no available improvements using any of the operators. For the smaller neighbourhoods defined by operators M1 and M2, when testing a possible insertion, a best improvement strategy is used, meaning that the insertion is tested on all available routes and the best improvement move is used. For the larger operators M3 and M4, a first found improvement strategy is used, meaning that as soon as an improving move is found then it is accepted. The local search uses the hierarchical objective because it is possible to reduce the number of routes in the solution using operators M2 and M4.

The intensified local search described above can be completed quickly but the solutions can often still be significantly improved with respect to the objectives of minimising the number of routes and minimising total distance. The next step in the algorithm is to focus on minimising the number of routes used.

3.2 Guided Ejection Search

Guided Ejection Search was originally proposed by Nagata and Braysy [14] for the vehicle routing problem with time windows (VRPTW). Nagata and Kobayashi then developed a version for PDPTW [15]. It only focuses on the objective of minimising the number of routes and their analysis showed that it was very effective on this single objective. An overview of the procedure is given in *Figure 2*.

```
1.  randomly select a route and un-assign all PD-pairs from it
2.
3.  for a fixed number of iterations
4.  {
5.          select an un-assigned PD-pair and try and insert it into an existing route
6.
7.          if PD-pair inserted
8.          {
9.                  if no more PD-pairs to assign
10.                         go to 1.
11.         }
12.         else
13.         {
14.                 try and insert the PD-pair by un-assigning one or more other pairs (the
                    ejection is heuristically selected by trying to avoid un-assigning
                    PD-pairs that were difficult to insert before)
15.
16.                 perturb the solution by randomly moving or swapping PD-pairs between
                    routes
17.         }
```

```
18. }
```

Figure 2 GES Outline

The method starts by randomly selecting a route and un-assigning all the PD-pairs in it. It then proceeds to try and re-insert the un-assigned pairs over the remaining routes. When it cannot insert a pair, it un-assigns (ejects) another pair(s) to allow it to insert it. It then perturbs the partial solution and tries again to insert an un-assigned pair. This is repeated until either there are no un-assigned pairs, in which case a route has been successfully removed, or a maximum number of iterations has been reached. If a route is removed then the procedure is repeated by selecting another route and un-assigning the pairs within it and then trying to insert them again over the remaining routes and so on.

The next pair selected for insertion is selected from an unassigned pairs list on a last in first out (LIFO) basis. LIFO was also used in [15], possibly because it improves the efficacy of the ejection heuristic which will be described later. When trying to insert the pair each route is tested in a random order and every possible position for the pair in the route is tested. If a feasible position for the pair is found then it is inserted. If more than one feasible position is found then the position for insertion is selected randomly. As with the local search, testing each possible position means trying each possible position for the pickup and for each possible position for the pickup also trying each possible position for the drop.

If the pair cannot be inserted then an attempt is made to insert it by ejecting one or two pairs from another route. First an attempt is made to insert it by ejecting a single pair and if this fails then every set of two pairs is tested to see if their ejection would allow the insertion. A maximum of two pairs was used for increased speed. If more than one set of pairs can be ejected to allow the insertion of the pair, then the set to eject is selected heuristically. Every time an attempt is made to insert a pair, a counter for that pair is increased by one. The heuristic for choosing which pair to eject is the pair with the lowest sum of the counter values (i.e. the set that has been previously attempted to be inserted the least number of times). The motivation behind the heuristic is that if a pair was previously difficult to insert (i.e. the counter value is high) then try not to eject it because it may be difficult to insert again.

The perturbation procedure at line 16 of Figure 2 not only creates the possibility of later being able to insert pairs but it also reduces the risks of cycling. The perturbation randomly selects one of two possible move operators (each with 0.5 probability) and then executes the move on the current partial solution. The first move (*PairMove*) randomly selects a route and a PD-pair within it, then randomly selects a second route and attempts to move the pair to a feasible position in the second route. If there is more than one feasible position in the second route then one is randomly selected. The second move (*SwapMove*) randomly selects two routes and a pair within each route. It then un-assigns the pairs and attempts to insert them into feasible positions in the opposite route. This time it selects the best possible positions (according to the secondary objective function – minimise total distance) rather than a random position. The perturbation finishes when 10 moves have been executed.

The implementation in the present work is similar to the original version by Nagata and Braysy except for two changes. The first difference is at line 14. The original algorithm examines all sets of pairs for ejection up to a fixed size. The larger the fixed size, the more sets there are to examine and the longer the algorithm takes. The approach in this paper only examines sets of length one first. That is, it tries ejecting a single pair first and then if this fails in allowing the insertion, then it tries ejecting two pairs. Again the two pairs are selected by minimising the sum of their previous insertion attempt counters.

The second main difference is the stopping condition. Instead of finishing after a certain number of iterations or a fixed time limit, the number of iterations is extended based on the progress of reducing

the number of un-assigned pairs. Every time a new partial solution with a new smallest number of un-assigned pairs is found then a counter is reset to zero. The counter is increased by one each time an attempt is made to perturb the solution by doing either *PairMove* or *SwapMove*. The procedure terminates if the counter reaches a predefined value (one million in our implementation). The motivation behind this heuristic is to terminate quickly if the progress suggests that the route will not be removed but to provide more time when the number of un-assigned pairs is being reduced but more slowly. This modified guided ejection chain mechanism maintains the intensification ability of the original approach but it also incorporates an adaptive ability to push the intensification or not according to the current solution.

3.3 Large Neighbourhood Search

After the modified GES, the local search is applied again followed by a large neighbourhood search. An overview of the LNS is shown in *Figure 3*.

```
1.  for a minimum number of iterations and maximum time limit
2.  {
3.          select a removal heuristic
4.
5.          un-assign heuristically selected PD-pairs in the solution using the removal
            heuristic
6.
7.          select an assignment heuristic
8.
9.          re-assign un-assigned PD-pairs using the assignment heuristic
10.
11.         accept or reject the new solution as the current solution using LAHC
12. }
```

Figure 3 LNS Outline

The LNS can be described as a “disrupt and repair” heuristic. It repeatedly un-assigns some PD-pairs from a solution and then attempts to heuristically re-assign them but creating an improved solution. The method is based on the ALNS of Ropke and Pisinger but with several changes. One of the main changes was to replace a simulated annealing + noise acceptance criterion with late acceptance hill climbing (LAHC) [4]. The main reason for this was to have a streamlined version by simplifying parameter setting because LAHC has only one parameter to set. LAHC is very similar to SA in that it accepts non-improving solutions but it replaces the probability-based acceptance criterion by a time-based deterministic one. At the start of the algorithm LAHC may accept many non-improving solutions and so provide more search diversification whereas at the end the search intensifies as less and less non-improving solutions are accepted. LAHC is described by *Figure 4*. In the figure, the initial solution is the solution created by the GES phase followed by the local search and the candidate solutions are the solutions generated by the removal and re-assignment heuristics. The *LHC_LEN* parameter was set as 2000 in all the experiments. A small amount of testing was performed in selecting this parameter but these initial tests suggested that this parameter did not have a large impact on the overall performance of the entire algorithm. It is possible that some additional performance gains could be achieved by tuning this parameter or more advanced sensitivity analysis (or even dynamically adapting it).

```
    Input parameters:
     Input solution s
     The length of the costs array LHC_LEN
1.  Calculate initial cost function C(s)
2.
3.  Create a new array (costs) of length LHC_LEN
4.
5.  FOR x{0..LHC_LEN-1} SET costs[x] := C(s)
6.
7.  SET iter := 0
```

```
8.
9.  UNTIL stopping condition
10.
11.    Construct a candidate solution s* from s
12.
13.    SET x := iter mod LHC_LEN
14.
15.    IF C(s*) ≤ costs[x] or C(s*) ≤ C(s)
16.       then accept the candidate (SET s := s*)
17.    ELSE
18.       reject the candidate (SET s := s)
19.
20.    SET costs[x] := C(s)
21.
22.    SET iter := iter+1
23.
24. END UNTIL
```

Figure 4 LAHC Outline

In the LNS, two removal heuristics are used: Shaw removal [24] and random removal [20, 24]. At each iteration one of the removal heuristics is randomly selected and applied. The Shaw removal heuristic aims to select a set of pairs PD that are similar. The idea is that if the pairs are similar then there is more possibility of re-arranging them in a new and possibly better way. If the pairs are all very different then they will probably be replaced exactly where they were originally assigned. The pair characteristics that are used to measure their similarity are: distance from each other, arrival times and demand. The formula for calculating the similarity is the same as given in [20]. The second heuristic is to simply randomly select a set of pairs. The probability of selecting the Shaw heuristic is set at 0.6, else the random selection heuristic is used. This creates a slight bias towards using the intelligent Shaw heuristic over the un-intelligent random heuristic. The number of pairs to remove by each heuristic is a number randomly selected from the range 4-80. These values were selected based on the results and guidance from [20].

To re-assign the pairs the regret assignment heuristic only is used [20]. The regret heuristic tries to improve upon greedy assignment by incorporating look-ahead. It does so by not only considering the best possible route for a pair insertion but by also the second, third, fourth… k^{th} best routes. When selecting which pair to insert next it selects pairs that have less possible positions for insertion that are low cost relative to their other possible positions. The motivation is that if that pair is not inserted now there may be regret later if that position is no longer available due to a previous insertion in the route. The parameter *k* is randomly selected from 2,3,4,5,*#R* where *#R* is the number of routes in the current solution.

Although the GES only uses the objective of minimising the number of routes and ignores the objective of minimising distance, the LNS uses the full hierarchical objective. It is possible for the LNS to remove routes if a removal heuristic selects a set of pairs which includes all the pairs for an entire route and then the assignment heuristic re-assigns them over other routes. During the testing we did observe the LNS reducing the number of routes in a solution occasionally but as will be shown, the GES is far more effective for minimising the total number of routes.

The LNS stops when a minimum number of iterations (800 in this paper) without improvement has been done or both of the following are satisfied:

1) There was no improvement in the last 400 iterations
2) And a minimum time limit has been reached, set as twice the time taken to complete the GES phase

This streamlined LNS maintains the diversification and intensification ability but at the same time it excludes the adaptive mechanism which was shown to provide only a few extra percent benefit in performance [20].

3.4 Restarts

After the LNS is completed, the overall algorithm goes to step 3 in Figure 1 to try reducing the number of vehicles again in the best solution found so far, by using GES. After, if the number of vehicles was not reduced then the best solution so far is perturbed using the same perturbation function as in GES. The number of perturbation moves is set as the instance's number of PD-pairs multiplied by 0.2. This is larger than the perturbation used within the GES phase where only ten moves are made. A larger perturbation is performed here to increase the search diversification whereas within the GES phase the perturbation is to try and allow a single PD-pair to be inserted. The algorithm then continues by applying the local search followed by LNS again and so on. The algorithm terminates when a maximum time limit is reached. Hence the heuristic approach proposed in this paper alternates between a random (when no route is removed) and a greedy (when a route is removed) perturbation to the current solution to then perform an effective LNS that balances intensification and diversification. All algorithm parameters are summarised in Table 1.

GES	
Maximum total perturbs	1000000
Perturbation	
Max perturbs within GES	10
Max perturbs during Restarts	Max[20, m * 0.2]
Probability of selecting *PairMove*	0.5
Probability of selecting *SwapMove*	0.5
LNS	
Stopping conditions	Min 800 consecutive iterations without improvement OR (Min 400 without improvement AND Min 2x CPU time used by GES)
LAHC array length (LHC_LEN)	2000
Min PD-Pairs removed by removal heuristic	4
Max PD-Pairs removed by removal heuristic	80
Probability of selecting Shaw heuristic	0.6
Probability of selecting removal heuristic	0.4

Table 1 Parameters summary

4 Results

To test the algorithm the benchmark instances of Li and Lim [11] are used[1]. There are approximately 360 instances categorised into six groups of different sizes ranging from approximately 50 PD-pairs up to 500 PD-pairs. Each group is also subdivided into instances with clustered locations, randomly

[1] Available at http://www.sintef.no/projectweb/top/pdptw/li-lim-benchmark/

distributed locations and randomly clustered locations. Each sub group is then further split by instances with short planning horizons and instances with long planning horizons.

Three sets of experiments were performed. The first was to investigate the benefit of the adaptive heuristic added to the GES. As described earlier, this heuristic terminates the GES phase more quickly when the progress suggests that an extra route will not be eliminated but allows more time when the progress suggests it is getting closer to removing a route. The second set of experiments was to investigate different configurations of the individual components and their combined benefit. The third analysis was to simply compare solutions generated with the current best knowns.

For the first two sets of experiments, five different algorithms were applied to all the test instances. The algorithms tested are the full algorithm and then four other versions, each with different components removed. The aim was to investigate the impact of the individual components or whether there was not any benefit in combining components when given the same computation time, or if combining the algorithms produces a more effective overall algorithm. The configurations were as follows:

1. LS+AGES+LNS (Algo1): The full algorithm as described and using the adaptive heuristic for the GES (labelled **A**daptive GES).
2. LS+GES+LNS (Algo2) : The same as 1 but without the adaptive heuristic in GES.
3. LS+AGES (Algo3) : The same as 1 but without the LNS phase, to see if the LS alone is sufficient at minimising the distance objective.
4. LS+LNS (Algo4) : The same as 1 but without the AGES phase which aims at minimising total routes. LS and LNS are both also able to minimise total routes on their own but this test was to investigate whether they are sufficient on their own if given the extra time not used by the removed AGES phase.
5. AGES+LNS (Algo5) : The same as 1 but with the LS phase removed. Previous papers indicated that LNS is much more effective than LS so there may be no benefit in including the LS phase, and instead just giving more time to the LNS phase.

Note that GES will exit sooner than AGES and so LNS in Algo2 will also have less time than LNS in Algo1 per iteration. However because all algorithms are being run for the same fixed time Algo2 will complete more iterations than Algo1 and so the overall CPU time distributed between the different phases will be similar overall.

On the ‘100’ group of instances, 5 minutes of computation time was allowed. On the ‘200’ and ‘400’ groups, 15 minutes. On the ‘600’ group, 30 minutes and on the ‘800’ and ‘1000’ groups, 60 minutes. These values were chosen based on similar run times in other papers [20] [15]. All runs were performed on an Intel Xeon CPU E5-1620 @ 3.5 GHz utilising a single core per run. 32GB RAM was available (although testing showed the algorithm requires a maximum of 70MB on the largest instances). The code was written in C#.

Table 2 lists the total number of vehicles used and the total distance for all the solutions for each group of instances, for each algorithm. These results are further broken down in Table 3 in which we rank the algorithms by how they performed against each other. For each group of instances, we record the total number of times that each algorithm found the best solution, the second best solution, the third best, fourth best and fifth best out of the five algorithms. Kendall’s non-parametric test is applied to the rankings to determine if the pairwise comparisons between two algorithms is statistically

significant. The mean values and P-values used in the statistical test are given in Table 4. Pairwise comparisons are made to see if the differences are statistically significant at the 0.05 level.

		LS+AGES+LNS		LS+GES+LNS		LS+AGES		LS+LNS		AGES+LNS	
Inst.	t (m)	Veh.	Dist.	Veh.	Dist.	Veh.	Dist.	Veh.	Dist.	Veh.	Dist.
100	5	402	58163.22	**402**	**58162.29**	402	59145.32	404	58175.81	402	58604.47
200	15	601	186158.57	601	185610.99	**600**	**197910.36**	619	176257.69	601	187890.12
400	15	1142	447627.39	1145	441007.53	**1140**	**485337.69**	1183	408446.49	1144	452306.60
600	30	**1643**	**935948.36**	1653	915686.48	1644	998440.66	1710	824991.45	1643	947724.82
800	60	2146	1551495.35	2153	1550252.37	2146	1664710.86	2223	1383687.84	**2136**	**1545456.09**
1000	60	2634	2310830.27	2645	2282149.28	2636	2443110.48	2733	2047589.35	**2605**	**2259241.06**

Table 2 Algorithm comparisons on all instances

Size	Rank	LS+AGES+LNS	LS+GES+LNS	LS+AGES	LS+LNS	AGES+LNS
100	First	53	54	41	52	39
	Second	1	2	0	0	0
	Third	2	0	1	1	1
	Fourth	0	0	4	1	7
	Fifth	0	0	10	2	9
200	First	36	38	17	25	17
	Second	12	10	0	10	4
	Third	7	11	6	5	6
	Fourth	4	1	15	2	22
	Fifth	1	0	22	18	11
400	First	25	28	10	18	14
	Second	14	20	2	5	6
	Third	18	7	5	2	17
	Fourth	3	5	30	0	16
	Fifth	0	0	13	35	7
600	First	27	23	6	9	19
	Second	18	18	6	7	7
	Third	14	11	8	4	15
	Fourth	1	8	33	1	11
	Fifth	0	0	7	39	8
800	First	24	18	8	17	16
	Second	22	19	3	1	12
	Third	12	15	7	3	14
	Fourth	1	7	35	2	8
	Fifth	1	1	7	37	10
1000	First	25	15	4	12	21
	Second	17	21	4	3	11
	Third	14	11	10	2	14
	Fourth	2	8	32	2	7
	Fifth	0	3	8	39	5
All	First	190	176	86	133	126
	Second	84	90	15	26	40

	Third	67	55	37	17	67
	Fourth	11	29	149	8	71
	Fifth	2	4	67	170	50

Table 3 Algorithm rankings

Test	P-value	Mean Ranks				
		Algo1	Algo2	Algo3	Algo4	Algo5
All	<.01	2.25	2.38	3.69	3.65	3.03
100	<.01	2.72	2.68	3.38	2.80	3.41
200	<.01	2.3	2.21	3.84	3.23	3.42
400	<.01	2.26	2.11	3.76	3.78	3.10
600	<.01	2.04	2.31	3.68	4.11	2.87
800	<.01	2.11	2.46	3.69	3.88	2.87
1000	<.01	2.09	2.57	3.76	4.07	2.51

Table 4 Algorithm mean rankings and p-values

For the pairwise comparisons we define A<B as meaning A has lower rank than B but the pairwise comparison is not significant. We define A<<B as meaning A has lower rank than B and the pairwise comparison is statistically significant. The rankings are as follows:

Over all instances we may rank the five algorithms as Algo1 < Algo2 << Algo5 << Algo4 < Algo3.
On the '100' instances the rank result is Algo2 < Algo1 < Algo4 < Algo3 < Algo5.
On the '200' instances the rank result is Algo2 < Algo1 << Algo4 < Algo5 < Algo3.
On the '400' instances the rank result is Algo2 < Algo1 << Algo5 < Algo3 < Algo4.
On the '600' instances the rank result is Algo1 < Algo2 < Algo5 << Algo3 < Algo4. (Algo1 << Algo5).
On the '800' instances the rank result is Algo1 < Algo2 < Algo5 << Algo3 < Algo4.
On the '1000' instances the rank result is Algo1 < Algo2 < Algo5 << Algo3 < Algo4.

Looking at Table 2 adaptive GES produces solutions with less routes than the GES (apart from the 100 and 200 instances where they are the same). When we compare the rankings pairwise GES is better on the smaller instances but AGES is better on the larger instances and over all instances. However the mean rankings are too similar to say the difference is statistically significant.

Investigating the benefit of including the (A)GES phase, it is clear that it is very effective. Algo4 (no GES) is always worse than Algo1 and Algo2 (the full algorithms with AGES or GES) and the pairwise comparisons are statistically significant. Similarly it is clear than the LNS is an important component of the algorithm. When the LNS phase is removed (Algo3), the full algorithms (Algo1 and Algo2) are significantly better. It is clear than giving extra time and more iterations to LS and GES is not as effective as including the LNS albeit with less time for each phase and less iterations. Algo5 is also statistically better than Algo3 showing that using LNS instead of LS is more effective. Finally we can conclude that over all instances, including the LS phase is more effective than not including it (Algo5) but on the largest instances 800, and 1000, the superiority is still visible in the mean rankings but is not large enough to be statistically significant. These results show that combining the three components in this configuration, local search with specialised moves, streamlined large neighbourhood search and adaptive guided ejection search, is more effective than using just two of the components. Using all three components means there is less time available for each method but it still

more effective than just using two of the components even if there is more time available for each individual phase.

Next we compare the results against the best known results in peer-reviewed publications and the current best knowns that have been verified on the SINTEF website but have not been published in peer-reviewed outlets and for which no information is available about computation times and methods used. For comparing against published methods (

	LNS			GES			LS+AGES+LNS		
Instances	Veh.	Dist.	t (s)	Veh.	Dist.	t (s)	Veh.	Dist.	t (s)
100	402	56060	-	-	-		402	58163.22	300
200	606	180419	-	601	-	3000	601	186158.57	900
400	1157	420396	-	1139	-	3000	1142	447627.39	900
600	1664	860898	-	1636	-	3000	1643	935948.36	1800
800	2181	1423063	-	2135	-	3000	2146	1551495.35	3600
1000	2646	2122922	-	2613	-	3000	2634	2310830.27	3600

Table 5Table 2), we use the results of the Adaptive LNS method of [20] and the GES method of [15]. These are the current best known published results. Comparing against these results is not simple though. For the best results of [20] we do not know the computation times. For their best results the authors "*report the best solutions obtained in several experiments with our ALNS heuristic and with various parameter settings*". We do not know how many experiments were run but we have an indication of computation times which are from 66 seconds per run on the smallest instances to 5370 seconds per run on the largest instances. We also know that the heuristic was run at least "*5 or 10 times on each instance*" (not including the different parameter setting testing) and that a 1.5 GHz Pentium IV processor was used. Again, comparing against Nagata and Kobayashi is difficult because their algorithm only minimises the number of vehicles used and we know from our results that using less vehicles often increases the total distance. They used an Opteron 2.6 GHz processor. Due to the unknown computation times, the differences in computing power and the difficulty in comparing summed values for a problem with hierarchical objectives we cannot have strong conclusions. The GES method produces solutions with less vehicles in total but the algorithm uses all its time minimising this objective where as our method only uses a large proportion of its time also minimising distance. The ALNS uses both objectives but produces solutions with more vehicles in total. Comparing total distances is not helpful because often solutions with less vehicles have longer distance. To assist future researchers and facilitate future comparisons we have included in this paper in Table 12 Results for a Single Run of Algo1Table 12 our results for a single run for a fixed run time for a single configuration (Algo1).

	LNS			GES			LS+AGES+LNS		
Instances	Veh.	Dist.	t (s)	Veh.	Dist.	t (s)	Veh.	Dist.	t (s)
100	402	56060	-	-	-		402	58163.22	300
200	606	180419	-	601	-	3000	601	186158.57	900
400	1157	420396	-	1139	-	3000	1142	447627.39	900
600	1664	860898	-	1636	-	3000	1643	935948.36	1800
800	2181	1423063	-	2135	-	3000	2146	1551495.35	3600
1000	2646	2122922	-	2613	-	3000	2634	2310830.27	3600

Table 5 Comparing against other methods

For the next comparison we compare against best knowns from published and unpublished methods. Tables 5-9 list the solutions found after applying the LS+AGES+LNS algorithm on all instances. The algorithm was allowed one hour computation but the best reported may be the best from several tests with different random seeds. Each table lists the solutions for each set of instances grouped by the number of locations. The tables also list the previous best known solutions for each instance, and the date it was found. The information is taken from SINTEF's website which is regularly updated[2]. Solutions in italics are equal to previous best knowns and solutions in bold italics are new best knowns.

The results show that the algorithm was able to find a large number of new best known solutions. On the 100 site instances the algorithm equalled the best knowns on all instances. On the 200 site instances 35 best knowns were equalled and seven new best knowns were found (out of 60). Of the seven new best knowns 3 were improvements in terms of the number of vehicles. For example on the instance *LR2_2_6* the new best known has a solution of three vehicles whereas the previous had four vehicles. This was an impressive result because the previous best known had stood for 15 years. On the 400 site instances there are 19 equal best knowns and 22 new best knowns (out of 60). On the 600 site instances there are six equal best knowns and 33 new best knowns (out of 60). On the 800 site instances there are five equal best knowns and 45 new best knowns (out of 60). On the 1000 site instances there are four equal best knowns and 35 new best knowns (out of 58). For many of the new best knowns the primary objective of reducing the number of vehicles is improved. This is particularly noticeable on the larger instances where the number of vehicles is reduced by more than one vehicle. For example on instance *LRC1_10_5* the previous best known required 76 vehicles whereas the new best known has only 72 vehicles. This demonstrates the benefit of using the AGES within the algorithm specifically for reducing the number of vehicles.

5 Conclusion

This paper proposes an effective and fast hybrid metaheuristic algorithm to tackle the pickup and delivery problem with time windows (PDPTW). The approach performs a large neighbourhood search (LNS) that incorporates mechanisms for intensification and diversification. The approach also incorporates mechanisms to perturb the current solution. Such perturbation can be greedy by removing a full route from the solution through guided ejection search, or random when such removal is not successful. Then, alternating the LNS with guided ejection search and local search has resulted in a relatively simple but demonstrably effective framework. The guided ejection search is specifically designed for minimising the number of routes within solutions. An adaptive heuristic is developed for the guided ejection search phase which provides more time to the heuristic when its progress suggests it is close to removing a route. The local search and large neighbourhood search are more focused on minimising travel distances. The aim is to combine these strengths into an overall robust and successful method. A new search neighbourhood operator was added to the local search method and the LNS was streamlined and simplified without loss of efficacy.

The results show that when any one of the components is removed the results are significantly worse. In other words, two components given more time is not as effective as the three components but with less time for each component. The adaptive heuristic for the GES phase is particularly effective on the larger instances. Including the local search phase benefits the smaller instances and including the LNS

[2] Retrieved from http://www.sintef.no/projectweb/top/pdptw/li-lim-benchmark/ on 25-Feb-2016

phase is better for all instance sizes. When tested on a large and well used benchmark data set, the algorithm is able to find 142 (out of 354 instances) new best known solutions, confirming its efficacy. The many new best-known solutions obtained with the LS+AGES+LNS heuristic algorithm proposed here have been already verified and hence published in the SINTEF's website.

Although a large amount of research, development and testing was required to develop this algorithm, there are still possibilities for further research. The Li and Lim benchmark instances are a very useful resource that have stimulated and enabled innovative research within a competitive and verifiable environment. Although that research forms the basis for many commercial vehicle routing problem solvers [9] it could be argued that more realistic benchmark instances could lead to even more effective methods for real-world problems. Data sets that contain requirements such as driver breaks rules, maximum driving hours, working time constraints, soft time windows etc. could have significant practical benefit. We believe that the algorithm presented could be adapted to handle these requirements but there would undoubtedly be new research required for such new challenges within benchmark data sets.

Appendix

	Best known			LS+AGES+LNS			Best known			LS+AGES+LNS	
Instance	Veh	Distance	Date	Veh	Distance	Instance	Veh	Distance	Date	Veh	Distance
lc101	10	828.94	23-Jun-05	10	828.94	lr201	4	1253.23	28-Feb-03	4	1253.23
lc102	10	828.94	23-Jun-05	10	828.94	lr202	3	1197.67	23-Jun-05	3	1197.67
lc103	9	1035.35	27-Jun-03	9	1035.35	lr203	3	949.40	23-Jun-05	3	949.40
lc104	9	860.01	11-Apr-03	9	860.01	lr204	2	849.05	23-Jun-05	2	849.05
lc105	10	828.94	23-Jun-05	10	828.94	lr205	3	1054.02	23-Jun-05	3	1054.02
lc106	10	828.94	2001	10	828.94	lr206	3	931.63	23-Jun-05	3	931.63
lc107	10	828.94	23-Jun-05	10	828.94	lr207	2	903.06	2001	2	903.06
lc108	10	826.44	2001	10	826.44	lr208	2	734.85	2001	2	734.85
lc109	9	1000.60	27-Jun-03	9	1000.60	lr209	3	930.59	09-Mar-03	3	930.59
lc201	3	591.56	23-Jun-05	3	591.56	lr210	3	964.22	23-Jun-05	3	964.22
lc202	3	591.56	2001	3	591.56	lr211	2	911.52	15-May-03	2	911.52
lc203	3	591.17	11-Mar-03	3	591.17	lrc101	14	1708.80	23-Jun-05	14	1708.80
lc204	3	590.60	08-Mar-03	3	590.60	lrc102	12	1558.07	19-Feb-03	12	1558.07
lc205	3	588.88	2001	3	588.88	lrc103	11	1258.74	23-Jun-05	11	1258.74
lc206	3	588.49	23-Jun-05	3	588.49	lrc104	10	1128.40	23-Jun-05	10	1128.40
lc207	3	588.29	23-Jun-05	3	588.29	lrc105	13	1637.62	23-Jun-05	13	1637.62
lc208	3	588.32	23-Jun-05	3	588.32	lrc106	11	1424.73	28-Feb-03	11	1424.73
lr101	19	1650.80	2001	19	1650.80	lrc107	11	1230.14	18-Feb-03	11	1230.14
lr102	17	1487.57	23-Jun-05	17	1487.57	lrc108	10	1147.43	28-Feb-03	10	1147.43
lr103	13	1292.68	23-Jun-05	13	1292.68	lrc201	4	1406.94	28-Feb-03	4	1406.94
lr104	9	1013.39	2001	9	1013.39	lrc202	3	1374.27	23-Jun-05	3	1374.27
lr105	14	1377.11	23-Jun-05	14	1377.11	lrc203	3	1089.07	23-Jun-05	3	1089.07
lr106	12	1252.62	23-Jun-05	12	1252.62	lrc204	3	818.66	23-Mar-03	3	818.66
lr107	10	1111.31	23-Jun-05	10	1111.31	lrc205	4	1302.20	23-Jun-05	4	1302.20
lr108	9	968.97	23-Jun-05	9	968.97	lrc206	3	1159.03	12-Mar-03	3	1159.03
lr109	11	1208.96	27-Feb-03	11	1208.96	lrc207	3	1062.05	04-Jan-04	3	1062.05
lr110	10	1159.35	23-Jun-05	10	1159.35	lrc208	3	852.76	23-Jun-05	3	852.76
lr111	10	1108.90	23-Jun-05	10	1108.90						
lr112	9	1003.77	23-Jun-05	9	1003.77						

Table 6 Best Solutions for 100 Site Instances

	Best known			LS+AGES+LNS			Best known			LS+AGES+LNS	
Instance	Veh	Distance	Date	Veh	Distance	Instance	Veh	Distance	Date	Veh	Distance
LC1_2_1	20	2704.57	23-Jun-05	20	2704.57	LR2_2_1	5	4073.10	09-Dec-03	5	4073.10
LC1_2_2	19	2764.56	23-Jun-05	19	2764.56	LR2_2_2	4	3796.00	14-Feb-03	4	3796.00
LC1_2_3	17	3128.61	08-Jul-03	17	3128.61	LR2_2_3	4	3098.36	08-Jul-03	4	3098.36
LC1_2_4	17	2693.41	27-Jun-03	17	2693.41	LR2_2_4	3	2486.14	08-Jul-03	3	2491.73
LC1_2_5	20	2702.05	23-Jun-05	20	2702.05	LR2_2_5	4	3438.39	13-Dec-03	4	3438.39
LC1_2_6	20	2701.04	2001	20	2701.04	LR2_2_6	4	3201.54	2001	3	4639.85
LC1_2_7	20	2701.04	23-Jun-05	20	2701.04	LR2_2_7	3	3124.57	19-Nov-15	3	3201.68
LC1_2_8	19	3379.97	19-Nov-15	19	3397.65	LR2_2_8	2	2552.16	14-Apr-15	2	2586.42
LC1_2_9	18	2724.24	23-Jun-05	18	2724.24	LR2_2_9	3	3930.49	25-Feb-05	3	3927.13
LC1_2_10	17	2942.13	19-Nov-15	17	2947.00	LR2_2_10	3	3282.45	19-Nov-15	3	3274.96
LC2_2_1	6	1931.44	2001	6	1931.44	LRC1_2_1	19	3606.06	14-Dec-03	19	3606.06
LC2_2_2	6	1881.40	23-Jun-05	6	1881.40	LRC1_2_2	15	3671.02	19-Nov-15	15	3683.24
LC2_2_3	6	1844.33	15-Apr-03	6	1844.33	LRC1_2_3	13	3161.44	19-Nov-15	13	3154.92
LC2_2_4	6	1767.12	2001	6	1767.12	LRC1_2_4	10	2631.82	08-Jul-03	10	2631.82
LC2_2_5	6	1891.21	23-Jun-05	6	1891.21	LRC1_2_5	16	3715.81	27-Jun-03	16	3715.81
LC2_2_6	6	1857.78	29-Dec-03	6	1857.78	LRC1_2_6	16	3572.16	19-Nov-15	16	3572.16
LC2_2_7	6	1850.13	10-Dec-03	6	1850.13	LRC1_2_7	14	3666.34	13-Apr-15	14	3697.71
LC2_2_8	6	1824.34	2001	6	1824.34	LRC1_2_8	13	3167.23	19-Nov-15	13	3202.16
LC2_2_9	6	1854.21	01-Sep-03	6	1854.21	LRC1_2_9	13	3157.34	19-Nov-15	13	3157.34
LC2_2_10	6	1817.45	23-Jun-05	6	1817.45	LRC1_2_10	12	2928.90	19-Nov-15	12	2948.60
LR1_2_1	20	4819.12	2001	20	4819.12	LRC2_2_1	6	3595.18	14-Apr-15	6	3595.18
LR1_2_2	17	4621.21	08-Jul-03	17	4621.21	LRC2_2_2	5	3184.23	17-Nov-15	5	3342.00
LR1_2_3	14	4656.11	02-Nov-15	14	4402.38	LRC2_2_3	4	2907.35	17-Nov-15	4	2948.56
LR1_2_4	10	3031.20	15-Jul-03	10	3044.69	LRC2_2_4	3	2861.74	26-May-14	3	2908.50
LR1_2_5	16	4760.18	27-Jun-03	16	4760.18	LRC2_2_5	5	2776.93	27-Jun-03	5	2776.93
LR1_2_6	14	4175.16	27-Jun-03	13	4800.94	LRC2_2_6	5	2707.96	21-Dec-03	5	2707.96
LR1_2_7	12	3550.61	08-Jul-03	12	3550.61	LRC2_2_7	4	3018.05	17-Nov-15	4	3057.23
LR1_2_8	9	2766.42	19-Nov-15	9	2814.32	LRC2_2_8	4	2399.89	17-Nov-15	4	2400.19
LR1_2_9	14	4343.86	14-Apr-15	13	5050.75	LRC2_2_9	4	2208.49	08-Jul-03	4	2208.49
LR1_2_10	11	3692.20	19-Nov-15	11	3748.06	LRC2_2_10	3	2442.59	17-Nov-15	3	2664.99

Table 7 Best Solutions for 200 Site Instances

	Best known			LS+AGES+LNS			Best known			LS+AGES+LNS	
Instance	Veh	Distance	Date	Veh	Distance	Instance	Veh	Distance	Date	Veh	Distance
LC1_4_1	40	7152.06	16-Jun-03	40	7152.06	LR2_4_1	8	9726.88	27-Jun-03	8	9726.88
LC1_4_2	38	8007.79	08-Oct-15	38	8007.79	LR2_4_2	7	9440.93	16-Nov-15	7	9473.11
LC1_4_3	33	8162.80	09-Oct-15	32	9252.95	LR2_4_3	6	8116.53	25-Feb-05	5	10658.64
LC1_4_4	30	6451.68*	2001	30	6918.00	LR2_4_4	4	6649.78	08-Jul-03	4	6721.54
LC1_4_5	40	7150.00	19-Apr-03	40	7150.00	LR2_4_5	6	10084.44	11-Jan-16	6	10152.96
LC1_4_6	40	7154.02	2001	40	7154.02	LR2_4_6	5	9044.03	14-Dec-15	5	9145.93
LC1_4_7	40	7149.43	15-Mar-03	40	7149.43	LR2_4_7	5	6729.67	14-Dec-15	5	6964.65
LC1_4_8	39	7111.16	2001	39	7111.16	LR2_4_8	4	5356.37	02-Oct-15	4	5454.27
LC1_4_9	36	7451.20	09-Oct-15	36	7451.20	LR2_4_9	6	7930.55	04-Jan-16	6	7998.74
LC1_4_10	35	7387.13	08-Jul-03	35	7325.01	LR2_4_10	5	7846.99	16-Nov-15	5	8349.58
LC2_4_1	12	4116.33	2001	12	4116.33	LRC1_4_1	36	9127.15	25-Feb-05	36	9124.52
LC2_4_2	12	4144.29	15-May-03	12	4144.29	LRC1_4_2	31	8346.06	08-Jul-03	31	8346.06
LC2_4_3	12	4424.08	27-May-14	12	4418.88	LRC1_4_3	25	7307.09	27-Jun-03	24	7856.72
LC2_4_4	12	3743.95*	2001	12	4038.00	LRC1_4_4	19	5806.20	30-Jun-11	19	5841.95
LC2_4_5	12	4030.63	01-May-03	12	4030.63	LRC1_4_5	32	8867.38	30-Jun-11	32	8872.08
LC2_4_6	12	3900.29	20-Apr-03	12	3900.29	LRC1_4_6	30	8423.70	30-Jun-11	30	8396.08
LC2_4_7	12	3962.51	27-Jun-03	12	3962.51	LRC1_4_7	28	8037.87	30-Jun-11	28	8295.76

LC2_4_8	12	3844.45	2001	12	3844.45	LRC1_4_8	27	7563.09	30-Jun-11	26	8173.63
LC2_4_9	12	4188.93	08-Jul-03	12	4188.93	LRC1_4_9	26	7790.26	30-Jun-11	25	8181.32
LC2_4_10	12	3828.44	27-Jun-03	12	3828.44	LRC1_4_10	24	7065.73	08-Jul-03	23	7222.97
LR1_4_1	40	10639.75	2003	40	10639.75	LRC2_4_1	12	7454.14	20-Oct-15	12	7454.14
LR1_4_2	31	10015.85	25-Feb-05	31	9985.28	LRC2_4_2	10	7424.72	28-Jan-16	10	7605.61
LR1_4_3	22	9458.99	25-Apr-15	22	9291.25	LRC2_4_3	9	5410.19	19-Jan-16	8	6576.48
LR1_4_4	16	6744.33	08-Jul-03	16	6710.99	LRC2_4_4	5	5322.43	08-Jul-03	5	5779.02
LR1_4_5	29	10599.54	25-Feb-05	28	11374.06	LRC2_4_5	11	6120.13	27-Jun-03	10	7462.66
LR1_4_6	24	10326.45	21-Apr-15	24	9891.02	LRC2_4_6	9	6337.08	21-Jan-16	9	6342.74
LR1_4_7	19	8200.37	25-Feb-05	18	8999.97	LRC2_4_7	8	6326.50	20-Jan-16	8	6704.21
LR1_4_8	14	5946.44	08-Jul-03	14	5944.67	LRC2_4_8	7	5814.93	19-Jan-16	7	6070.90
LR1_4_9	24	9886.14	25-Feb-05	24	9862.65	LRC2_4_9	7	5259.20	20-Jan-16	6	6877.02
LR1_4_10	21	8016.62	08-Jul-03	20	8364.66	LRC2_4_10	6	5585.18	22-Jan-16	6	5840.81

Table 8 Best Solutions for 400 Site Instances

	Best known			LS+AGES+LNS			Best known			LS+AGES+LNS	
Instance	Veh	Distance	Date	Veh	Distance	Instance	Veh	Distance	Date	Veh	Distance
LC1_6_1	60	14095.60	23-Jun-05	60	14095.64	LR2_6_1	11	21945.30	14-Jul-03	11	21955.29
LC1_6_2	57	15120.97	10-Apr-15	57	15048.16	LR2_6_2	10	19666.59	25-Feb-05	9	23903.03
LC1_6_3	50	14683.43	14-Jul-03	50	14724.64	LR2_6_3	8	15609.96	14-Jul-03	7	19183.41
LC1_6_4	47	13648.03	14-Jul-03	48	13348.84	LR2_6_4	6	10819.45	14-Jul-03	6	11994.47
LC1_6_5	60	14086.30	23-Jun-05	60	14086.30	LR2_6_5	9	19567.41	14-Jul-03	9	19411.73
LC1_6_6	60	14090.79	14-Jul-03	60	14090.79	LR2_6_6	8	17262.96	14-Jul-03	7	22570.45
LC1_6_7	60	14083.76	13-Jul-03	60	14083.76	LR2_6_7	6	15812.42	14-Jul-03	6	15526.81
LC1_6_8	59	14554.27	13-Jul-03	58	14880.70	LR2_6_8	5	10950.90	14-Jul-03	5	11410.69
LC1_6_9	54	14706.12	14-Jul-03	54	14661.73	LR2_6_9	8	18799.36	14-Jul-03	8	19838.35
LC1_6_10	52	18762.62	21-Apr-15	52	15204.30	LR2_6_10	7	17034.63	14-Jul-03	7	17129.58
LC2_6_1	19	7977.98	14-Jul-03	19	7977.98	LRC1_6_1	52	27262.21	18-Apr-15	52	18312.60
LC2_6_2	18	9914.10	18-Jan-16	18	9940.34	LRC1_6_2	44	16302.54	25-Feb-05	43	17063.21
LC2_6_3	17	8718.22	04-Jan-16	18	7436.50	LRC1_6_3	36	14060.31	14-Jul-03	36	14115.00
LC2_6_4	17	7902.66	04-Jan-16	17	8022.96	LRC1_6_4	25	10950.52	14-Jul-03	25	11006.02
LC2_6_5	19	8047.37	27-Jun-03	19	8047.37	LRC1_6_5	47	16742.55	14-Jul-03	46	17067.17
LC2_6_6	19	8094.11	14-Jul-03	18	8859.78	LRC1_6_6	44	16894.37	14-Jul-03	42	17405.48
LC2_6_7	19	7997.96	15-Jan-16	19	7997.96	LRC1_6_7	39	15394.87	14-Jul-03	38	15609.86
LC2_6_8	18	7579.93	14-Jul-03	18	7580.88	LRC1_6_8	36	15154.79	14-Jul-03	33	15919.78
LC2_6_9	18	8864.29	12-Jan-16	18	9019.78	LRC1_6_9	35	15134.24	14-Jul-03	34	15236.23
LC2_6_10	17	7965.41	14-Jan-16	17	8064.71	LRC1_6_10	31	13925.51	14-Jul-03	29	14607.38
LR1_6_1	59	22838.30	27-Jun-03	59	22824.32	LRC2_6_1	16	14817.72	14-Jul-03	16	14892.18
LR1_6_2	45	20246.18	14-Jul-03	45	20355.13	LRC2_6_2	14	12758.77	14-Jul-03	13	15649.64
LR1_6_3	37	18073.14	14-Jul-03	37	17987.49	LRC2_6_3	10	12812.67	25-Feb-05	10	14845.21
LR1_6_4	28	13269.71	14-Jul-03	28	13191.79	LRC2_6_4	7	10574.87	25-Feb-05	7	11282.95
LR1_6_5	38	22562.81	25-Feb-05	38	22489.30	LRC2_6_5	14	13009.52	14-Jul-03	13	15196.60
LR1_6_6	32	20641.02	14-Jul-03	31	22188.80	LRC2_6_6	13	12642.68	04-Jan-04	12	17149.19
LR1_6_7	25	17162.90	14-Jul-03	24	18531.68	LRC2_6_7	11	11975.28	24-Apr-15	10	16094.11
LR1_6_8	19	11957.59	14-Jul-03	18	12255.29	LRC2_6_8	10	12163.43	14-Jul-03	9	15024.47
LR1_6_9	32	21423.05	14-Jul-03	32	21117.75	LRC2_6_9	9	13768.01	14-Jul-03	9	14560.69
LR1_6_10	27	18723.13	14-Jul-03	26	19028.25	LRC2_6_10	8	12016.94	14-Jul-03	7	15098.15

Table 9 Best Solutions for 600 Site Instances

	Best known			LS+AGES+LNS			Best known			LS+AGES+LNS	
Instance	Veh	Distance	Date	Veh	Distance	Instance	Veh	Distance	Date	Veh	Distance
LC1_8_1	80	25184.38	16-Jun-03	80	25184.38	LR2_8_1	15	33816.90	22-Aug-03	14	46452.00
LC1_8_2	77	33329.26	04-May-15	77	26864.13	LR2_8_2	12	32575.97	22-Aug-03	12	35732.86

LC1_8_3	65	25918.45	22-Aug-03	63	27459.81	LR2_8_3	10	25310.53	22-Aug-03	9	30485.19
LC1_8_4	60	22970.88	22-Aug-03	60	22943.54	LR2_8_4	7	19506.42	25-Feb-05	6	24285.15
LC1_8_5	80	25211.22	12-Jul-03	80	25211.22	LR2_8_5	12	32634.29	22-Aug-03	11	37332.53
LC1_8_6	80	25164.25	12-Jul-03	80	25164.25	LR2_8_6	10	27870.80	22-Aug-03	9	31372.05
LC1_8_7	80	25158.38	13-Jul-03	80	25158.38	LR2_8_7	8	25077.85	25-Feb-05	7	30605.35
LC1_8_8	78	25348.45	22-Aug-03	78	25381.04	LR2_8_8	5	19256.79	22-Aug-03	6	19390.75
LC1_8_9	73	25541.94	22-Aug-03	72	26360.69	LR2_8_9	10	30791.77	22-Aug-03	10	34699.60
LC1_8_10	71	25712.12	22-Aug-03	70	26811.45	LR2_8_10	9	28265.24	22-Aug-03	9	30147.88
LC2_8_1	24	11687.06	19-May-03	24	11687.06	LRC1_8_1	66	35901.74	04-May-15	66	32302.57
LC2_8_2	24	14358.92	22-Aug-03	24	14039.96	LRC1_8_2	56	28843.10	27-Jun-03	56	28042.91
LC2_8_3	24	13198.29	22-Aug-03	24	13265.98	LRC1_8_3	48	29276.29	04-May-15	48	24693.73
LC2_8_4	23	13376.82	22-Aug-03	24	12373.83	LRC1_8_4	34	23099.15	04-May-15	34	18806.89
LC2_8_5	25	12298.90	27-Jun-03	25	12329.80	LRC1_8_5	59	34909.37	04-May-15	58	31457.69
LC2_8_6	24	12702.87	22-Aug-03	24	12835.00	LRC1_8_6	56	29971.97	02-Jun-03	54	29836.27
LC2_8_7	25	11855.86	22-Aug-03	25	11854.44	LRC1_8_7	52	32430.86	10-Jul-15	51	28705.17
LC2_8_8	24	11482.88	22-Aug-03	24	11454.33	LRC1_8_8	47	29814.93	04-May-15	45	27374.52
LC2_8_9	24	11629.61	22-Aug-03	24	11629.41	LRC1_8_9	46	28955.40	10-Jul-15	44	25980.07
LC2_8_10	24	11578.58	22-Aug-03	24	11583.30	LRC1_8_10	42	24271.52	22-Aug-03	40	24582.28
LR1_8_1	80	39315.92	22-Aug-03	80	39292.13	LRC2_8_1	20	23289.40	22-Aug-03	20	23157.34
LR1_8_2	59	34370.37	22-Aug-03	59	34325.92	LRC2_8_2	18	21786.62	22-Aug-03	17	22686.62
LR1_8_3	44	29718.09	22-Aug-03	44	29676.42	LRC2_8_3	15	36127.35	04-May-15	14	21651.20
LR1_8_4	25	21197.65	22-Aug-03	25	21189.75	LRC2_8_4	11	38856.21	10-Jul-15	11	16232.51
LR1_8_5	50	39046.06	22-Aug-03	49	39624.94	LRC2_8_5	17	41062.80	04-May-15	16	24404.69
LR1_8_6	41	40488.25	04-May-15	40	35042.41	LRC2_8_6	16	21088.57	22-Aug-03	15	23854.87
LR1_8_7	31	33431.22	04-May-15	30	28252.49	LRC2_8_7	15	19695.96	22-Aug-03	14	24514.06
LR1_8_8	21	19570.21	22-Aug-03	20	20037.07	LRC2_8_8	13	19009.33	22-Aug-03	12	21508.49
LR1_8_9	42	36126.69	22-Aug-03	40	40077.86	LRC2_8_9	12	19003.68	22-Aug-03	11	21998.18
LR1_8_10	32	30200.86	22-Aug-03	31	32241.06	LRC2_8_10	10	19766.78	22-Aug-03	10	19712.20

Table 10 Best Solutions for 800 Site Instances

	Best known			LS+AGES+LNS			Best known			LS+AGES+LNS	
Instance	Veh	Distance	Date	Veh	Distance	Instance	Veh	Distance	Date	Veh	Distance
LC1_10_1	100	42488.66	23-Apr-03	100	42488.66	LR2_10_1	19	45422.58	25-Feb-05	18	56089.88
LC1_10_2	95	43858.42	10-Apr-15	94	45238.29	LR2_10_2	15	47824.44	25-Feb-05	14	58654.95
LC1_10_3	82	42631.11	25-Feb-05	80	45175.07	LR2_10_3	11	39894.32	25-Feb-05	11	45477.21
LC1_10_4	74	39217.18	23-May-15	74	38376.00	LR2_10_4	8	28314.95	25-Feb-05	8	29113.98
LC1_10_5	100	42477.40	10-Aug-03	100	42477.41	LR2_10_5	14	53209.98	25-Feb-05	14	54053.67
LC1_10_6	101	42838.39	11-Aug-03	101	42838.39	LR2_10_6	12	43792.11	25-Feb-05	11	53051.42
LC1_10_7	100	42854.99	25-Jun-05	100	42854.99	LR2_10_7	9	36728.20	25-Feb-05	9	41524.99
LC1_10_8	98	42951.56	25-Feb-05	98	42965.59	LR2_10_8	7	26278.09	25-Feb-05	7	30358.26
LC1_10_9	92	42391.98	25-Feb-05	91	43426.76	LR2_10_9	13	48447.49	25-Feb-05	13	50126.78
LC1_10_10	90	42435.16	25-Feb-05	88	42873.50	LR2_10_10	11	44155.66	25-Feb-05	11	43889.20
LC2_10_1	30	16879.24	2003	30	16879.24	LRC1_10_1	84	49315.30	27-Jun-03	82	49285.19
LC2_10_2	31	18980.98	25-Feb-05	31	19342.00	LRC1_10_2	72	58291.72	10-Jul-15	72	45289.03
LC2_10_3	30	17772.49	25-Feb-05	30	17943.81	LRC1_10_3	54	43435.85	10-Jul-15	53	36499.96
LC2_10_4	29	18089.98	25-Feb-05	29	18231.53	LRC1_10_4	40	27680.12	09-Apr-15	40	27987.45
LC2_10_5	31	17137.53	25-Feb-05	31	17289.55	LRC1_10_5	76	49816.18	25-Feb-05	72	51733.03
LC2_10_6	31	17198.01	25-Feb-05	31	17204.18	LRC1_10_6	69	44469.08	25-Feb-05	68	44444.34
LC2_10_7	31	19117.67	25-Feb-05	31	19347.93	LRC1_10_7	63	49951.54	10-Jul-15	61	41917.62
LC2_10_8	30	17018.63	25-Feb-05	30	17015.41	LRC1_10_8	58	50171.34	10-Jul-15	56	42640.89
LC2_10_9	31	17565.95	25-Feb-05	30	20057.42	LRC1_10_9	56	46710.65	10-Jul-15	53	40848.27
LC2_10_10	29	17425.55	25-Feb-05	29	17898.25	LRC1_10_10	50	45156.36	10-Jul-15	48	36092.22
LR1_10_1	100	56903.88	25-Feb-05	100	56875.21	LRC2_10_1	22	35059.40	30-Nov-15	22	34960.69
LR1_10_2	80	49652.10	25-Feb-05	80	49627.16	LRC2_10_2	21	30932.74	25-Feb-05	20	33576.15
LR1_10_3	54	42124.44	25-Feb-05	54	42346.86	LRC2_10_3	16	28403.51	25-Feb-05	16	29495.48

LR1_10_4	28	32133.36	25-Feb-05	28	31617.58	LRC2_10_4	12	23083.20	25-Feb-05	11	26239.60
LR1_10_5	61	59135.86	25-Feb-05	59	60616.90	LRC2_10_5	18	33451.16	30-Nov-15	17	37312.43
LR1_10_6	49	57674.00	10-Jul-15	48	51842.12	LRC2_10_6	17	31470.58	30-Nov-15	17	32745.90
LR1_10_7	37	38936.54	25-Feb-05	36	40127.52	LRC2_10_7	17	29499.79	30-Nov-15	16	32537.87
LR1_10_8	26	29452.32	25-Feb-05	26	29099.22		-	-	-		
LR1_10_9	50	52223.15	25-Feb-05	49	53353.22		-	-	-		
LR1_10_10	40	46218.35	25-Feb-05	39	47290.00	LRC2_10_10	12	29380.12	30-Nov-15	11	31334.49

Table 11 Best Solutions for 1000 Site Instances

Instance	Veh.	Dist.	Time (s)	Instance	Veh.	Dist.	Time (s)
lc101	10	828.94	300	lr201	4	1253.23	300
lc102	10	828.94	300	lr202	3	1197.67	300
lc103	9	1038.35	300	lr203	3	952	300
lc104	9	861.95	300	lr204	2	849.05	300
lc105	10	828.94	300	lr205	3	1054.02	300
lc106	10	828.94	300	lr206	3	931.63	300
lc107	10	828.94	300	lr207	2	903.06	300
lc108	10	826.44	300	lr208	2	734.85	300
lc109	9	1000.6	300	lr209	3	930.59	300
lc201	3	591.56	300	lr210	3	964.22	300
lc202	3	591.56	300	lr211	2	911.52	300
lc203	3	591.17	300	lrc101	14	1708.8	300
lc204	3	638.18	300	lrc102	12	1558.07	300
lc205	3	588.88	300	lrc103	11	1258.74	300
lc206	3	588.49	300	lrc104	10	1128.4	300
lc207	3	588.29	300	lrc105	13	1637.62	300
lc208	3	588.32	300	lrc106	11	1424.73	300
lr101	19	1650.8	300	lrc107	11	1230.14	300
lr102	17	1487.57	300	lrc108	10	1147.43	300
lr103	13	1292.68	300	lrc201	4	1455.54	300
lr104	9	1013.39	300	lrc202	3	1374.27	300
lr105	14	1377.11	300	lrc203	3	1089.07	300
lr106	12	1252.62	300	lrc204	3	818.66	300
lr107	10	1111.31	300	lrc205	4	1302.2	300
lr108	9	968.97	300	lrc206	3	1159.03	300
lr109	11	1208.96	300	lrc207	3	1062.05	300
lr110	10	1159.35	300	lrc208	3	852.76	300
lr111	10	1108.9	300				
lr112	9	1003.77	300				
LC1_2_1	20	2704.57	900	LR2_2_1	5	4073.1	900
LC1_2_2	19	2764.56	900	LR2_2_2	4	3796	900
LC1_2_3	17	3127.78	900	LR2_2_3	4	3098.36	900
LC1_2_4	17	2695.35	900	LR2_2_4	3	2511.77	900
LC1_2_5	20	2702.05	900	LR2_2_5	4	3438.39	900
LC1_2_6	20	2701.04	900	LR2_2_6	3	5079.87	900
LC1_2_7	20	2701.04	900	LR2_2_7	3	3226.37	900
LC1_2_8	19	3453.1	900	LR2_2_8	2	2907.04	900
LC1_2_9	18	2724.24	900	LR2_2_9	3	3998.73	900
LC1_2_10	17	3114.95	900	LR2_2_10	3	3372.39	900
LC2_2_1	6	1931.44	900	LRC1_2_1	19	3606.06	900
LC2_2_2	6	1881.4	900	LRC1_2_2	15	3678.89	900
LC2_2_3	6	1844.7	900	LRC1_2_3	13	3194.84	900

LC2_2_4	6	1767.12	900	LRC1_2_4	10	2633.25	900
LC2_2_5	6	1891.21	900	LRC1_2_5	16	3715.81	900
LC2_2_6	6	1857.78	900	LRC1_2_6	16	3713.7	900
LC2_2_7	6	1850.13	900	LRC1_2_7	14	3769.84	900
LC2_2_8	6	1824.34	900	LRC1_2_8	13	3209.72	900
LC2_2_9	6	1854.21	900	LRC1_2_9	13	3250.5	900
LC2_2_10	6	1817.45	900	LRC1_2_10	12	2963.71	900
LR1_2_1	20	4819.12	900	LRC2_2_1	6	3595.18	900
LR1_2_2	17	4621.21	900	LRC2_2_2	5	3253.69	900
LR1_2_3	14	4405.43	900	LRC2_2_3	4	3003.99	900
LR1_2_4	10	3034.17	900	LRC2_2_4	3	3092.38	900
LR1_2_5	16	4773	900	LRC2_2_5	5	2776.93	900
LR1_2_6	13	4800.94	900	LRC2_2_6	5	2707.96	900
LR1_2_7	12	3550.61	900	LRC2_2_7	4	3080.74	900
LR1_2_8	9	2810.26	900	LRC2_2_8	4	2399.99	900
LR1_2_9	14	4372.32	900	LRC2_2_9	4	2208.52	900
LR1_2_10	11	3714.16	900	LRC2_2_10	3	2691.21	900
LC1_4_1	40	7152.06	900	LR2_4_1	8	9842.79	900
LC1_4_2	38	8199.22	900	LR2_4_2	7	10510.74	900
LC1_4_3	33	8512.53	900	LR2_4_3	6	9449.97	900
LC1_4_4	30	7166.68	900	LR2_4_4	4	7447.74	900
LC1_4_5	40	7150	900	LR2_4_5	6	10658.54	900
LC1_4_6	40	7154.02	900	LR2_4_6	5	9663.35	900
LC1_4_7	40	7149.43	900	LR2_4_7	5	7438.04	900
LC1_4_8	39	7111.16	900	LR2_4_8	4	6178.83	900
LC1_4_9	36	8240.48	900	LR2_4_9	6	9741.13	900
LC1_4_10	35	7785.96	900	LR2_4_10	5	8588.43	900
LC2_4_1	12	4116.33	900	LRC1_4_1	36	9148.44	900
LC2_4_2	12	4144.29	900	LRC1_4_2	31	8366.84	900
LC2_4_3	12	4427.69	900	LRC1_4_3	24	7940.56	900
LC2_4_4	12	4155.54	900	LRC1_4_4	19	5851.35	900
LC2_4_5	12	4030.63	900	LRC1_4_5	32	8973.9	900
LC2_4_6	12	3900.29	900	LRC1_4_6	30	8561.18	900
LC2_4_7	12	4290	900	LRC1_4_7	28	8552.88	900
LC2_4_8	12	3844.45	900	LRC1_4_8	26	8341.27	900
LC2_4_9	12	4189.25	900	LRC1_4_9	25	8475.36	900
LC2_4_10	12	3829.34	900	LRC1_4_10	23	7552.75	900
LR1_4_1	40	10639.75	900	LRC2_4_1	12	7587.81	900
LR1_4_2	31	10009.1	900	LRC2_4_2	10	8488.36	900
LR1_4_3	22	9437.76	900	LRC2_4_3	8	6739.24	900
LR1_4_4	16	7101.42	900	LRC2_4_4	5	6086	900
LR1_4_5	29	10621.62	900	LRC2_4_5	10	7835.13	900
LR1_4_6	24	9872.59	900	LRC2_4_6	9	6522.83	900
LR1_4_7	19	8501.8	900	LRC2_4_7	8	6780.16	900
LR1_4_8	14	5953.51	900	LRC2_4_8	7	5969.35	900
LR1_4_9	24	10052.08	900	LRC2_4_9	7	5420.5	900
LR1_4_10	20	8584.19	900	LRC2_4_10	6	7590.79	900
LC1_6_1	60	14095.64	1800	LR2_6_1	11	22004.07	1800
LC1_6_2	57	15062.27	1800	LR2_6_2	9	23804.19	1800
LC1_6_3	50	14684.81	1800	LR2_6_3	7	19376.93	1800
LC1_6_4	48	13346.44	1800	LR2_6_4	6	13085.37	1800
LC1_6_5	60	14086.3	1800	LR2_6_5	9	22058.19	1800

LC1_6_6	60	14090.79	1800	LR2_6_6	7	23302.97	1800
LC1_6_7	60	14083.76	1800	LR2_6_7	6	16431.55	1800
LC1_6_8	58	15205.1	1800	LR2_6_8	8	21904.29	1800
LC1_6_9	54	14859.61	1800	LR2_6_9	8	22699.46	1800
LC1_6_10	52	15811.84	1800	LR2_6_10	7	17914.13	1800
LC2_6_1	19	7977.98	1800	LRC1_6_1	52	18467.73	1800
LC2_6_2	18	10685.95	1800	LRC1_6_2	43	17092.29	1800
LC2_6_3	18	7440.4	1800	LRC1_6_3	36	14110.35	1800
LC2_6_4	17	8068.58	1800	LRC1_6_4	25	11099.15	1800
LC2_6_5	19	8047.37	1800	LRC1_6_5	46	17167.51	1800
LC2_6_6	18	9622.26	1800	LRC1_6_6	42	17576.85	1800
LC2_6_7	19	8010.02	1800	LRC1_6_7	38	15557.42	1800
LC2_6_8	18	7579.93	1800	LRC1_6_8	33	16049.69	1800
LC2_6_9	18	11084.98	1800	LRC1_6_9	34	15476.3	1800
LC2_6_10	18	7493.7	1800	LRC1_6_10	29	15213.25	1800
LR1_6_1	59	22821.65	1800	LRC2_6_1	16	15118.51	1800
LR1_6_2	45	20458.42	1800	LRC2_6_2	13	17675.41	1800
LR1_6_3	37	18157.15	1800	LRC2_6_3	10	13508.6	1800
LR1_6_4	28	13419.2	1800	LRC2_6_4	7	15185.64	1800
LR1_6_5	38	22777.17	1800	LRC2_6_5	13	15438.84	1800
LR1_6_6	31	23145.06	1800	LRC2_6_6	12	15897.94	1800
LR1_6_7	25	17481.34	1800	LRC2_6_7	10	16674.47	1800
LR1_6_8	18	12683.97	1800	LRC2_6_8	9	15397.81	1800
LR1_6_9	32	21878.25	1800	LRC2_6_9	9	14019.72	1800
LR1_6_10	26	19555.51	1800	LRC2_6_10	8	12924.28	1800

LC1_8_1	80	25184.38	3600	LR2_8_1	15	37159.43	3600
LC1_8_2	77	27053.48	3600	LR2_8_2	12	37143.89	3600
LC1_8_3	64	27080.22	3600	LR2_8_3	9	31195.51	3600
LC1_8_4	60	23046.35	3600	LR2_8_4	12	46347.34	3600
LC1_8_5	80	25211.22	3600	LR2_8_5	11	38199.48	3600
LC1_8_6	80	25164.25	3600	LR2_8_6	9	36157.03	3600
LC1_8_7	80	25158.38	3600	LR2_8_7	8	28099.62	3600
LC1_8_8	78	25525.38	3600	LR2_8_8	11	35100.74	3600
LC1_8_9	72	26309.29	3600	LR2_8_9	10	34160.16	3600
LC1_8_10	70	26896.55	3600	LR2_8_10	9	28650.21	3600
LC2_8_1	24	11687.06	3600	LRC1_8_1	66	32343.63	3600
LC2_8_2	24	14432.89	3600	LRC1_8_2	56	28028.22	3600
LC2_8_3	24	13489.52	3600	LRC1_8_3	48	24868.53	3600
LC2_8_4	24	12724.81	3600	LRC1_8_4	34	18651.8	3600
LC2_8_5	25	12329.8	3600	LRC1_8_5	58	31743.18	3600
LC2_8_6	24	12938.93	3600	LRC1_8_6	55	28986.21	3600
LC2_8_7	25	13313.06	3600	LRC1_8_7	51	28684.47	3600
LC2_8_8	24	11481.82	3600	LRC1_8_8	45	27386.97	3600
LC2_8_9	24	11630.33	3600	LRC1_8_9	44	25541.71	3600
LC2_8_10	24	11586.62	3600	LRC1_8_10	40	24775.03	3600
LR1_8_1	80	39314.59	3600	LRC2_8_1	20	23280.89	3600
LR1_8_2	59	34639.58	3600	LRC2_8_2	17	23662.24	3600
LR1_8_3	44	29672.24	3600	LRC2_8_3	15	19558.58	3600
LR1_8_4	25	21351.41	3600	LRC2_8_4	11	21502.55	3600
LR1_8_5	49	39911.37	3600	LRC2_8_5	16	26300.04	3600
LR1_8_6	40	35319.74	3600	LRC2_8_6	15	23003.29	3600
LR1_8_7	30	28180.3	3600	LRC2_8_7	14	23874.42	3600
LR1_8_8	20	20352.08	3600	LRC2_8_8	12	23407.84	3600

LR1_8_9	41	36802.82	3600	LRC2_8_9	11	21294.5	3600
LR1_8_10	31	31612.63	3600	LRC2_8_10	10	22986.75	3600
LC1_10_1	100	42488.66	3600	LR2_10_1	18	58315.37	3600
LC1_10_2	95	43813.15	3600	LR2_10_2	14	62490.96	3600
LC1_10_3	82	43103.5	3600	LR2_10_3	11	47933.54	3600
LC1_10_4	74	38849.97	3600	LR2_10_4	14	42660.35	3600
LC1_10_5	100	42477.41	3600	LR2_10_5	14	54270.3	3600
LC1_10_6	101	42838.39	3600	LR2_10_6	12	51141.9	3600
LC1_10_7	100	42854.99	3600	LR2_10_7	10	64292.78	3600
LC1_10_8	98	43012.3	3600	LR2_10_8	14	64986.32	3600
LC1_10_9	91	43146.18	3600	LR2_10_9	13	52350.2	3600
LC1_10_10	88	44889.54	3600	LR2_10_10	11	48354.75	3600
LC2_10_1	30	16879.24	3600	LRC1_10_1	82	49267.82	3600
LC2_10_2	31	19572.89	3600	LRC1_10_2	72	45408	3600
LC2_10_3	30	18045.44	3600	LRC1_10_3	53	37085.36	3600
LC2_10_4	30	18637.19	3600	LRC1_10_4	40	28722.74	3600
LC2_10_5	31	17289.55	3600	LRC1_10_5	72	52609.93	3600
LC2_10_6	31	17212.05	3600	LRC1_10_6	68	44559.84	3600
LC2_10_7	31	18893.71	3600	LRC1_10_7	61	42347.71	3600
LC2_10_8	30	17310.82	3600	LRC1_10_8	56	42529.31	3600
LC2_10_9	30	19420.8	3600	LRC1_10_9	53	40648.11	3600
LC2_10_10	29	17567.64	3600	LRC1_10_10	48	37535.48	3600
LR1_10_1	100	57060.03	3600	LRC2_10_1	22	35357.39	3600
LR1_10_2	80	49688.13	3600	LRC2_10_2	21	31843.7	3600
LR1_10_3	54	42999.25	3600	LRC2_10_3	16	34504.22	3600
LR1_10_4	29	30784.3	3600	LRC2_10_4	24	40740.78	3600
LR1_10_5	60	58370.14	3600	LRC2_10_5	17	38958.91	3600
LR1_10_6	48	50117.58	3600	LRC2_10_6	17	31476.76	3600
LR1_10_7	36	39091.89	3600	LRC2_10_7	16	32788.83	3600
LR1_10_8	26	29338.14	3600				
LR1_10_9	49	53212.61	3600				
LR1_10_10	40	46634.81	3600	LRC2_10_10	11	32048.61	3600

Table 12 Results for a Single Run of Algo1

References

1. Battarra, M., J. Cordeau, and M. Iori, *Chapter 6: Pickup-and-Delivery Problems for Goods Transportation*, in *Vehicle Routing*. 2014, Society for Industrial and Applied Mathematics. p. 161-191.
2. Bent, R. and P.V. Hentenryck, *A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows.* Computers & Operations Research, 2006. **33**(4): p. 875-893.
3. Berbeglia, G., et al., *Static pickup and delivery problems: a classification scheme and survey.* TOP, 2007. **15**(1): p. 1-31.
4. Burke, E.K. and Y. Bykov, *The Late Acceptance Hill-Climbing Heuristic.* European Journal of Operational Research, Aceepted for publication 2016.
5. Cherkesly, M., G. Desaulniers, and G. Laporte, *A population-based metaheuristic for the pickup and delivery problem with time windows and LIFO loading.* Computers & Operations Research, 2015. **62**: p. 23-35.
6. Dumas, Y., J. Desrosiers, and F. Soumis, *The pickup and delivery problem with time windows.* European Journal of Operational Research, 1991. **54**(1): p. 7-22.

7. Gendreau, M., et al., *Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries.* Transportation Research Part C: Emerging Technologies, 2006. **14**(3): p. 157-174.
8. Gendreau, M., G. Laporte, and D. Vigo, *Heuristics for the traveling salesman problem with pickup and delivery.* Computers & Operations Research, 1999. **26**(7): p. 699-714.
9. Hall, R. and J. Partyka, *Vehicle Routing Software Survey: Higher expectations drive transformation.* ORMS-Today, 2016. **43**(1).
10. Kammarti, R., et al. *A new hybrid evolutionary approach for the pickup and delivery problem with time windows*. in *Systems, Man and Cybernetics, 2004 IEEE International Conference on*. 2004.
11. Li, H. and A. Lim, *A Metaheuristic for the Pickup and Delivery Problem with Time Windows.* International Journal on Artificial Intelligence Tools, 2003. **12**(02): p. 173-186.
12. Lu, Q. and M. Dessouky, *An Exact Algorithm for the Multiple Vehicle Pickup and Delivery Problem.* Transportation Science, 2004. **38**(4): p. 503-514.
13. Masson, R., F. Lehuédé, and O. Péton, *An Adaptive Large Neighborhood Search for the Pickup and Delivery Problem with Transfers.* Transportation Science, 2012. **47**(3): p. 344-355.
14. Nagata, Y. and O. Bräysy, *A powerful route minimization heuristic for the vehicle routing problem with time windows.* Operations Research Letters, 2009. **37**(5): p. 333-338.
15. Nagata, Y. and S. Kobayashi, *Guided Ejection Search for the Pickup and Delivery Problem with Time Windows*, in *Evolutionary Computation in Combinatorial Optimization: 10th European Conference, EvoCOP 2010, Istanbul, Turkey, April 7-9, 2010. Proceedings*, P. Cowling and P. Merz, Editors. 2010, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 202-213.
16. Nagata, Y. and S. Kobayashi, *A Memetic Algorithm for the Pickup and Delivery Problem with Time Windows Using Selective Route Exchange Crossover*, in *Parallel Problem Solving from Nature, PPSN XI: 11th International Conference, Kraków, Poland, September 11-15, 2010, Proceedings, Part I*, R. Schaefer, et al., Editors. 2010, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 536-545.
17. Nanry, W.P. and J. Wesley Barnes, *Solving the pickup and delivery problem with time windows using reactive tabu search.* Transportation Research Part B: Methodological, 2000. **34**(2): p. 107-121.
18. Parragh, S., K. Doerner, and R. Hartl, *A survey on pickup and delivery problems.* Journal für Betriebswirtschaft, 2008. **58**(1): p. 21-51.
19. Ropke, S. and J.-F. Cordeau, *Branch and Cut and Price for the Pickup and Delivery Problem with Time Windows.* Transportation Science, 2009. **43**(3): p. 267-286.
20. Ropke, S. and D. Pisinger, *An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows.* Transportation Science, 2006. **40**(4): p. 455-472.
21. Ruland, K.S. and E.Y. Rodin, *The pickup and delivery problem: Faces and branch-and-cut algorithm.* Computers & Mathematics with Applications, 1997. **33**(12): p. 1-13.
22. Savelsbergh, M. and M. Sol, *DRIVE: Dynamic Routing of Independent Vehicles.* Operations Research, 1998. **46**(4): p. 474-490.
23. Savelsbergh, M.W.P. and M. Sol, *The General Pickup and Delivery Problem.* Transportation Science, 1995. **29**(1): p. 17-29.
24. Shaw, P., *Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems*, in *Principles and Practice of Constraint Programming — CP98: 4th International Conference, CP98 Pisa, Italy, October 26–30, 1998 Proceedings*, M. Maher and J.-F. Puget, Editors. 1998, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 417-431.
25. Venkateshan, P. and K. Mathur, *An efficient column-generation-based algorithm for solving a pickup-and-delivery problem.* Computers & Operations Research, 2011. **38**(12): p. 1647-1655.
26. Xu, H., et al., *Solving a Practical Pickup and Delivery Problem.* Transportation Science, 2003. **37**(3): p. 347-364.