Combining heuristic and exact methods to solve the Vehicle Routing problem with pickups, deliveries and time windows

Penny L. Holborn, Jonathan M. Thompson, and Rhyd Lewis

School of Mathematics, Cardiff University, Cardiff, Wales, UK
{holbornpl}@Cardiff.ac.uk

Abstract. The vehicle routing problem with pickups, deliveries and time windows (PDPTW) is an important member in the class of vehicle routing problems. In this paper a general heuristic to construct an initial feasible solution is proposed and compared with other construction methods. New route reconstruction heuristics are then shown to improve on this. These reconstruction heuristics look to reorder individual routes and recombine multiple routes to decrease the number of vehicles used in the solution. A tabu search scheme where the attribute to be recorded has been specifically adapted to the PDPTW is proposed. A new method based on branch and bound optimisation attempts to optimise the final ordering of requests in routes to further improve the solutions. Results are analysed for a standard set of benchmark instances and are shown to be competitive with the state of the art.

Key words: Vehicle Routing, pickup and delivery and tabu search

1 Introduction

The Vehicle Routing Problem (VRP) plays a central role in distribution management. It can be described as the problem of designing a set of routes that start at a depot and visit a set of geographically scattered customer locations, subject to a variety of side constraints. The VRP is known to be *NP*-hard due to it being an extension of the well known Travelling Salesman Problem (TSP), which is itself *NP*-hard. A helpful survey paper on the VRP is that of Laporte & Osman [1].

The Pickup and Delivery Problem with Time Windows (PDPTW) was first formulated by Savelsbergh and Sol [2]. The constraints are as follows: (1) each vehicle must start at the depot and return to the depot before the end of its operating interval; (2) a request's pickup must be scheduled before its corresponding delivery; (3) loads present within a vehicle at any one time must not exceed the maximum capacity of that vehicle; and (4) requests' pickup and delivery time windows must be adhered to. Note that a vehicle may wait at a location if some waiting time is expected at the vehicle's next destination and the problem is one where all requests are known in advance and no uncertainty exists.

Early research surrounding the PDPTW was concerned with the transportation of people instead of goods and is sometimes known as the dial a ride problem (DARP) [3–5]. The first metaheuristic proposed to solve the PDPTW was the reactive tabu search approach of Nanry and Barnes [6]. A two phase method proposed by Lau and Liang [7] developed this, where a construction heuristic was followed by tabu search. Another approach is that of Li and Lim [8] who have applied a tabu-embedded simulating annealing algorithm to solve the problem. They also produced benchmark instances for the PDPTW which are generated from Solomon's 56 benchmark instances [9]. These have since been used as the main basis for comparison of algorithms to solve the problem. Alternatively a two-stage hybrid algorithm has been presented by Bent [10] where the first stage uses a simple simulated annealing algorithm to decrease the number of routes, while the second stage uses large neighbourhood search to decrease the total travel cost. An adaptive large neighbourhood search heuristic has also been proposed by Ropke [11]. Another approach to this problem is by Pankratz [12], who use a grouping genetic algorithm (GGA) and this is extended to a multi-strategy grouping genetic algorithm (MSGGA) by Ding et al. [13]. In addition Dergis and Döhmer [14] show that the approach of indirect local search with greedy decoding gives results which are competitive with both [8] and [12]. Metaheuristics that apply learning mechanisms have been proposed by Lim et al. [15], specifically a squeaky wheel optimisation and more recently ant colony System was applied by Carabetti et al. [16].

In the design of our algorithm we first examine methods previously discussed in the literature, such as initial construction heuristics, neighbourhood search operators and tabu search. We will examine reconstruction heuristics previously applied to the TSP and VRP and look to adapt and evolve these to the PDPTW. Finally we augment our algorithm with a branch and bound method in order to improve the results.

The rest of the paper is organised as follows. The problem is formulated in Section 2. Section 3 provides details on the operators used in our algorithm including the construction of initial feasible solutions, route reconstructions and the branch and bound method. Section 4 provides information on the tabu search heuristic and our algorithmic framework. Finally Section 5 gives computational results and Section 6 provides a conclusion and directions for future research.

2 Problem Formulation

To define the PDPTW, let $V = \{v_0, v_1, \dots v_n\}$ be a set of geographically dispersed locations where v_0 denotes the depot and n is even. The set $N = V \setminus \{v_0\}$ defines the set of pickup and delivery requests and is partitioned into two subsets of equal size. The subset N^+ denotes the set of pickup locations and N^- the set of delivery locations. Therefore, $N^+ \cup N^- = N$, $N^+ \cap N^- = \emptyset$ and $|N^+| = |N^-| = \frac{n}{2}$ = number of pickup and delivery requests. In this problem each location $v_i \in V$ has an associated demand q_i, $(q_0 = 0)$, a service time s_i, $(s_0 = 0)$ and a service time window $[e_i, l_i]$, $(e_0 = l_0 = 0)$, where e_i, is the earliest

time that service at location i can begin and l_i, the latest time that service at location i can begin. With regards to the demand, $q_i > 0$ for $v_i \in N^+$ and $q_i < 0$ for $v_i \in N^-$. For each pair of nodes (v_i, v_j) $(0 \leq i \neq j \leq n)$ a non-negative distance d_{ij} is known, $d_{ij} = d_{ji}$, where distance is equal to time. If a vehicle reaches node v_i before time e_i, it needs to wait until e_i before the service can take place. Let A_i be the arrival time, D_i be the departure time and W_i the waiting time at location i. Then $D_i = \max\{A_i, e_i\} + s_i$. If $A_i < e_i$, then the vehicle has to wait at location i and $W_i = e_i - A_i$. Let M be the number of vehicles, C be the maximum length of operating interval and Q the maximum capacity of each vehicle.

To formulate the PDPTW, two variables are introduced:

$$x_{ij}^k = \begin{cases} 1, & \text{if vehicle } k \text{ goes from node } i \text{ to node } j \\ 0, & \text{otherwise.} \end{cases}$$

$$y_j = \text{load of the vehicle at node } j, \text{ after service at } j$$

and a constant:

$$z_{ij} = \begin{cases} 1, \text{if node } i \text{ and node } j \text{ are the corresponding pickup and} \\ \text{delivery nodes of a single request} \\ 0, \text{otherwise.} \end{cases}$$

The constraints are as follows:

$$\sum_{k=1}^{M} \sum_{j=1}^{n} x_{ij}^k = 1, \forall i \in V \tag{1}$$

$$\sum_{i=1}^{n} x_{i0}^k = 1, k \in [M] \tag{2}$$

$$\sum_{j=1}^{n} x_{0j}^k = 1, k \in [M] \tag{3}$$

$$\sum_{i=1}^{n} x_{ih}^k - \sum_{j=1}^{n} x_{hj}^k = 0, \forall h \in V, k \in [M] \tag{4}$$

$$z_{ij} = 1 \Rightarrow \sum_{l=1}^{n} x_{li}^k - \sum_{p=1}^{n} x_{pj}^k = 0, \forall i, j \in V, k \in [M] \tag{5}$$

$$y_j \leq Q, \forall j \in V \tag{6}$$

$$x_{ij}^k = 1 \Rightarrow y_i + q_i = y_j, \forall i, j \in V, k \in [M] \tag{7}$$

$$x_{ij}^k = 1 \Rightarrow D_i + d_{i,j} \leq A_j \Rightarrow A_j \leq D_j \Rightarrow D_i \leq D_j,$$
$$D_0 = 0, \forall i, j \in V, k \in [M] \tag{8}$$

$$z_{ij} = 1 \Rightarrow A_i \leq A_j, \forall i, j \in V \tag{9}$$

$$A_0 \leq C \tag{10}$$

In the above, constraint 1 ensures that each location is visited exactly once, while constraints 2 and 3 ensure that each vehicle departs from and arrives at the depot. Constraint 4 ensures that if a vehicle arrives at a location then it must also depart from that location. Constraint 5 ensures that the pickup and delivery of a request is carried out by exactly one vehicle. Constraints 6 and 7 together form the capacity constraints. Finally, the time window and precedence constraints are ensured by 2 and 9 and the constraint on the maximum operating interval is ensured by 10. The objective function is:

$$\text{Minimise} \sum_{k=1}^{M} \sum_{i,j \in N: i \neq j} d_{ij} x_{ij}^{k} \tag{11}$$

3 Algorithm operators

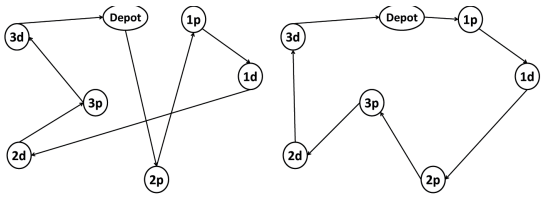
3.1 Construction methods

To construct an initial feasible solution a combination of random and greedy heuristics are applied. The algorithm builds a feasible solution by inserting, at each iteration, a random un-routed request into a current partial route or into a new route using a greedy method. All feasible insertions of both the pickup and delivery request are examined. The insertion which provides the minimal increase in cost to the solution is accepted. This includes the option of inserting the request into a new route. A similar method is also used in [12] and [14], where it is shown that adding an element of randomness generates varied initial solutions which are beneficial when applying neighbourhood operators as a larger search space is examined.

Preliminary results have shown that this method outperforms the simple greedy heuristic of Nanry and Barnes [6], which at each iteration inserts the request from *all* remaining requests that involves the lowest additional cost to the objective function. It also outperforms the method used by Li and Lim [8], which first initialises a route with a request using criteria based on the maximum increase to the objective function with routes then being completed using a greedy method.

3.2 Route Reconstruction heuristics

To attempt to improve on the initial solutions constructed we first examine 2 neighbourhood operators of Li and Lim [8]. The first of these is a *shift* operator. This denotes a reassignment of a request from one route to another. Secondly an *exchange* operator swaps a request from one route with a request of another. In both cases infeasible exchanges are forbidden and the operators attempt to insert a request into a route without making any change to the current ordering of that route. Naturally a higher proportion of neighbourhood moves will be



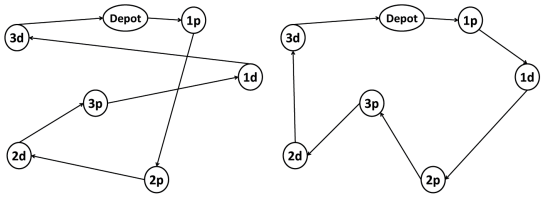
(a) Before reconstruction (b) After reconstruction

Fig. 1: Example of applying the single move within a route reconstruction

seen to retain feasibility if the existing ordering in a route can also be changed, though of course this will also bring additional overheads. To achieve this we suggest three different reconstruction heuristics.

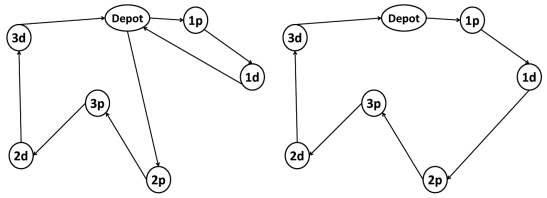
The **single move within a route** heuristic randomly selects a request and removes its pickup and delivery location from its route. It then attempts to insert the pickup and delivery locations in all other feasible positions within that route. If one exists, the insertion position which amounts to the largest reduction in distance is accepted. An example is shown in Figure 1. This method is based on that of *Or*-opt exchanges, see Or [17] but is adapted to the PDPTW.

The **single route reconstruction** attempts to reorder an entire route by first removing all requests from that route and re-inserting them based on three different methods. These are as follows: (1) by allocating at each iteration the location at which the next service can begin first, (this is the maximum of the time the vehicle can arrive at a location and the opening of the time window at that location); (2) by allocating the first pickup location to the route at random and each of the remaining pickup or delivery locations greedily; and (3) by allocating the first pickup whose location is the maximum distance from the depot first and then each of the remaining pickup or delivery locations greedily. Each location (pickup or delivery) is inserted separately. An example of this is shown in Figure 2. The single route reconstruction also attempts to reform a route whilst inserting a request from another. It attempts to find a feasible solution that includes the insertion of this request whilst also minimsing the overall total distanced travelled over all routes.



(a) Before reconstruction (b) After reconstruction

Fig. 2: Example of applying the single route reconstruction to a route



(a) Before reconstruction (b) After reconstruction

Fig. 3: Example of applying the multiple route reconstruction to two routes

Finally the **multiple route reconstruction** attempts to form multiple routes simultaneously. This is carried out for two routes with the aim of reducing to one or three routes with the aim of reducing to two. All requests are removed from the routes and the first route is initialised with the pickup whose location is the maximum distance from the depot. For the case of the second route, if one is used, the pickup which is maximum distance from the first is chosen. The routes are then reconstructed simultaneously using a greedy heuristic. For the second case this is only applied on a combination of routes, if at least one of the routes is an outlier with regards to the number of requests present. An example of this is shown in Figure 3.

3.3 Branch and bound method

To further improve our algorithm a method based on the large neighbourhood search (LNS) of [10] for the PDPTW is incorporated. The main idea is to take a part of a solution (in this case a single route or subset of that route) and find the optimal solution for this sub-problem via a branch and bound routine.

The process starts with a set of currently adjacent locations. According to the constraints of the problem, partly constructed solutions can be discarded: (a) if the delivery location of a request is inserted before the corresponding pickup; (b) if there remains a location still to be inserted that can no longer be feasibly serviced within its time window; (c) if a location cannot be feasibly serviced within its time window when placed after another location; (d) if the current total distance travelled exceeds the minimum recorded so far; and (e) if the minimum distance still to travel plus the current distance exceeds the minimum found. The limit of the initial bound is the total distance travelled of the route before the locations are removed. Branches are searched in the order of location where service can begin first. The search terminates once a complete exploration has taken place and the best solution is returned.

As this method is an exact approach it can be computationally expensive. Our results suggest it can be applied to routes with up to 14 locations. In cases where there are more than this, our approach is to apply branch and bound to successive overlapping sub-sections. This ensures locations located closely to one another are optimised in the same sub-section. In cases where $n > 14$ locations, the route is split into $2\left\lceil \frac{n}{14} \right\rceil - 1$ sub-sections. For example, if a route consists of 28 locations it is split into 3 sub-sections containing locations 1-14, 7-21 and 14-28 respectively.

4 Overall Algorithm

To further improve the algorithm a tabu search heuristic is to be added to the *shift* operator defined in Section 3.2. Within the the *shift* operator the request to be reassigned is selected at random and all feasible insertion positions of both the pickup and delivery are examined. If one is found, the insertion which provides the largest reduction in total travel distance is accepted. It is found that adding the *exchange* operator, both increased computational times and did not provide an improvement to the results when used in conjunction with the reconstruction heuristics.

From the literature, a tabu length and cycle length proportional to the number of requests to be serviced generally yields the most positive results, see [6] who present a reactive tabu search approach to solve the PDPTW. Our tabu search heuristic follows the general guidelines provided in [18] and a tabu tenure equal to the number of requests, $|N|$, and a cycle length equal to the number of locations to be visited, $|2N|$ are found to be most promising. The stopping condition is based on achieving a given number of iterations without improvement to the objective function or there being no more feasible moves.

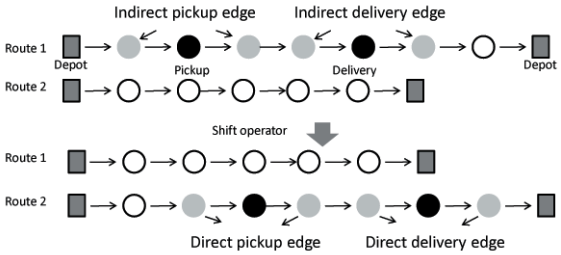
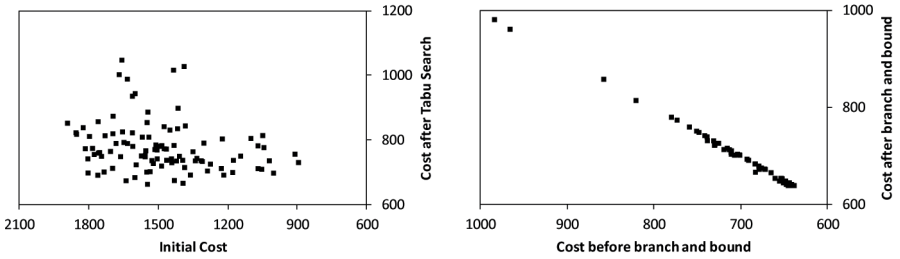


Fig. 4: Example of the tabu attributes added to the tabu list during a move by the Shift operator

The attribute to be stored within the tabu list follows the approach of [19] where edges removed and inserted within a solution are recorded for the VRP with simultaneous pick-up and delivery. In our case this is adapted to the PDPTW where the edges inserted into a solution are classed as the direct edges. These are the locations either side of the new insertions i.e. the locations before and after the insertion of the pickup location and the delivery location of a request. The edges removed from the solution are the indirect edges, i.e. the the locations before and after the pickup and delivery location that has been removed. The attribute to be recorded in the tabu list consists of both the inserted and removed pickup and delivery edges. If the arrangement of locations in a route after the insertion results in both the pickup and delivery edge being either directly or indirectly tabu then the move is disallowed. An example of the attribute to be stored in this tabu list is shown in Figure 4.

Preliminary results suggest that applying the branch and bound method as a final phase to the algorithm and not within the improvement phase yields

promising results. The branch and bound method optimises both routes and sub-sections of routes therefore it limits the number of successful tabu moves and reconstructions within the improvement phase as routes and sub-sections of routes are at a local minima. Applying the tabu search heuristic and the branch and bound method are computationally expensive. Investigations are performed to determine if each method may be performed on a subset of the most promising solutions and still achieve competitive results. Figure 5 contains a plot of the cost after the initial construction phase compared to the cost after applying the tabu search heuristic, and a plot of the the cost after the tabu search phase compared to the cost after applying the branch and bound method. Both for 100 random trials with the instance lc204. There is no correlation between achieving a lower cost for an initial solution and achieving a lower cost for the final solution after the tabu search heuristic. Therefore it will not be possible to select a proportion of initial solutions with a certain initial cost to apply the tabu search heuristic to achieve the most promising results. However there is a direct correlation between achieving a lower cost after the tabu search heuristic and achieving a lower cost after the branch and bound method. Therefore it seems reasonable to select a small subset of of low cost solutions after application of the tabu search heuristic which can then be improved via our branch and bound method.



(a) Tabu Search heuristic (b) Branch and bound method

Fig. 5: Scatter plot showing the before and after costs of each heuristic for 100 random trials with instance lc204

Our overall algorithm consists of first constructing an initial feasible solution which is passed to the improvement phase consisting of the tabu search and reconstruction heuristics outlined in Section 3.2 and above. These methods are carried out until no further improvement can be made. This process is repeated for a given number of iterations and the best 10% of solutions are passed to the branch and bound procedure outlined in Section 3.3. This method differs from others in the literature as a single run involves multiple restarts and only a portion of the best found solutions are passed to the final improvement phase. Preliminary results suggest that applying 300 iterations of the initial construction and improvement phase before passing 30 of the best found solutions to the branch and bound method yields run times that improve on [12] and are similar to that of [13]. This comparison is made using the average run time provided and multiplying by the 30 trials that were required to achieve their best found

solutions. Only computational times for 10 runs of the algorithm of [14] are provided in the literature and are a significant improvement on the others stated, however applying this figure to the case of 30 runs, these would also be comparable to our algorithm. The computational times of [8] will not be compared as their description does not indicate whether these are average values. Caution is needed when making a direct comparison between these computational times due to differences in computer specification.

5 Experimental results

For our experiments we used instances derived by [8], which are based on Solomon's 56 VRPTW 100-customer instances [9]. Each has 100-106 nodes, (i.e. 50-53 requests) and they are organised into 6 classes; lc1 and lc2 are clustered instances; lrc1 and lrc2 are those where requests are partially clustered and partially random; and lr1 and lr2 have randomly distributed requests. Instances ending in a 1 have a short scheduling horizon and those with a 2 have a longer scheduling horizon. Table 1 compares the results of our algorithm with those of Li and Lim [8], Pankratz [12], Dergis & Döhmer [14] and Ding et al. [13].

Caution is needed when making a direct comparisons with these results as our objective function is to minimise the total travel distance which is comparable to that of [12]. Li and Lim [8] however use a prioritised objective function with the order being: (1) minimise number of vehicles; (2) minimise total travel distance; (3) minimise total schedule duration; and (4) minimise total waiting time. The objective of Ding et al. [13] is similar to this although it does not include minimising the total schedule duration. The objective of [14] is to minimise the number of vehicles followed by minimising the total travel distance. For the approach of [8] the overall number of independent runs per instance is not reported and the average solution quality is not discussed. The best results of [12] and [13] are reported after 30 runs of their algorithm and for [14] best results are reported after 10 runs. Our algorithms are implemented using C++ and executed on a PC under Windows XP with a 3.10GHz processor.

Considering the results in Table 1 our algorithm achieves the best known solutions for 51 of the problem instances and with a total travel distance of 57662.02 are competitive with the state of the art. For [8] 40 of the best known solutions are achieved with a total travel distance of 58184.91, [14] achieve 42 with a total travel distance of 57678.4 and [13] achieve 51 with a total travel distance of 57652.05. The minimal total travel distance of 57638.48 is achieved by [12], however with only 42 of the best known solutions found.

The initial improvement phase of our algorithm achieves an average total travel distance for the 56 instances of 61162.92. This is a significant 40% improvement from 101883.30 for the construction phase alone. This is further reduced by the branch and bound method to 58302.08, a further decrease of 4.7%. For the branch and bound method the average decrease in cost for the instances with a longer scheduling horizon is 6.8 % with a 10.0% decrease in the lr2 instances. For the instances with a longer scheduling horizon, the number of vehicles is

Table 1: Summary of results

	Holborn et al.					Li & Lim [8]	Pankratz [12]	Dergis & Döhmer[14]	Ding et al.[13]
Instance	**TD**[1]	**NV**[2]	**TDI** [3]	**TDB**[4]	**CT**[5]	**TD**[1]	**TD**[1]	**TD**[1]	**TD**[1]
lc101	828.94	10	828.94	828.94	92	828.94	828.94	828.94	828.94
lc102	828.94	10	828.94	828.94	367	828.94	828.94	828.94	828.94
lc103	827.86	10	830.58	827.87	679	827.86	827.86	827.86	827.86
lc104	818.60	9	828.17	818.60	1135	861.95	818.60	860.01	860.01
lc105	828.94	10	828.94	828.94	131	828.94	828.94	828.94	828.94
lc106	828.94	10	829.76	828.94	102	828.94	828.94	828.94	828.94
lc107	828.94	10	828.94	828.94	99	828.94	828.94	828.94	828.94
lc108	826.44	10	827.05	826.44	232	826.44	826.44	826.44	826.44
lc109	827.82	10	837.66	827.82	714	827.82	827.82	827.82	827.82
Total lc1	**7445.41**	**89**	**7468.96**	**7445.41**	**3550**	**7488.77**	**7445.42**	**7486.83**	**7486.83**
lr101	1650.80	20	1658.69	1650.80	44	1650.78	1650.80	1650.80	1650.80
lr102	1487.57	19	1515.57	1487.57	459	1487.57	1487.57	1487.57	1487.57
lr103	1292.68	14	1322.47	1292.68	391	1292.68	1292.68	1292.68	1292.68
lr104	1013.39	12	1095.19	1026.02	867	1013.39	1013.99	1013.99	1013.39
lr105	1377.11	16	1383.94	1377.11	142	1377.11	1377.11	1377.11	1377.11
lr106	1252.62	13	1261.18	1252.62	215	1252.62	1252.62	1252.62	1252.62
lr107	1111.31	12	1146.91	1111.31	475	1111.31	1111.31	1111.31	1111.31
lr108	968.97	11	989.88	968.97	438	968.97	968.97	968.97	968.97
lr109	1208.96	15	1271.37	1208.96	317	1239.96	1208.96	1208.96	1208.96
lr110	1165.83	14	1206.88	1166.90	696	1159.35	1165.83	1159.35	1159.35
lr111	1108.90	14	1144.68	1108.90	640	1108.90	1108.90	1108.90	1108.90
lr112	1003.77	12	1059.20	1021.80	929	1003.77	1003.77	1003.77	1003.77
Total lr1	**14641.91**	**172**	**15055.96**	**14673.64**	**5614**	**14666.41**	**14642.51**	**14636.03**	**14635.43**
lrc101	1703.21	14	1731.58	1703.21	98	1708.80	1703.21	1708.80	1708.80
lrc102	1558.07	12	1612.43	1558.07	300	1563.55	1558.07	1558.07	1558.07
lrc103	1258.74	11	1294.23	1258.74	417	1258.74	1258.74	1258.74	1258.74
lrc104	1128.40	10	1141.07	1128.40	481	1128.40	1128.40	1128.40	1128.40
lrc105	1637.62	13	1670.72	1640.15	290	1637.62	1637.62	1637.62	1637.62
lrc106	1424.73	11	1510.12	1439.66	242	1425.53	1424.73	1424.73	1424.73
lrc107	1230.14	11	1265.43	1230.14	357	1230.15	1230.14	1230.14	1230.15
lrc108	1147.43	10	1203.62	1147.43	369	1147.97	1147.43	1147.96	1147.43
Total lrc1	**11088.34**	**92**	**11429.20**	**11105.80**	**2555**	**11100.76**	**11088.34**	**11094.46**	**11093.94**
lc201	591.56	3	591.56	591.56	284	591.56	591.56	591.56	591.56
lc202	591.56	3	598.88	591.56	1416	591.56	591.56	591.56	591.56
lc203	591.17	3	625.56	591.17	1776	585.56	591.17	591.17	591.17
lc204	590.60	3	664.14	635.01	2835	591.17	590.60	590.60	590.60
lc205	588.88	3	603.78	588.88	1127	588.88	588.88	588.88	588.88
lc206	588.49	3	618.24	602.06	2316	588.49	588.49	588.49	588.49
lc207	588.29	3	598.08	588.29	1125	588.29	588.29	588.29	588.29
lc208	588.32	3	606.66	588.32	1237	588.32	588.32	588.32	588.32
Total lc2	**4718.87**	**24**	**4906.89**	**4776.84**	**12116**	**4713.83**	**4718.87**	**4718.87**	**4718.87**
lr201	1253.23	4	1297.60	1254.29	1756	1263.84	1253.23	1253.23	1253.23
lr202	1197.67	3	1355.43	1261.68	2602	1197.67	1197.67	1197.67	1197.67
lr203	949.40	3	1128.34	970.32	3809	949.40	952.29	949.40	949.40
lr204	849.05	2	1042.35	904.49	6153	849.05	849.05	849.05	849.05
lr205	1054.02	3	1168.50	1073.66	3327	1054.02	1054.02	1054.02	1054.02
lr206	931.63	3	1083.69	935.75	3800	931.63	931.63	931.63	931.63
lr207	905.45	2	1094.37	972.03	5471	903.06	903.60	903.06	903.05
lr208	734.85	2	882.29	752.46	7266	734.85	736.00	734.85	734.85
lr209	930.59	3	1049.56	947.17	3657	937.05	932.43	930.59	930.59
lr210	964.22	3	1108.68	988.93	3650	964.22	964.22	964.22	964.22
lr211	884.29	3	978.92	905.68	4976	927.80	888.15	896.76	884.29
Total lr2	**10654.39**	**31**	**12189.72**	**10966.46**	**46467**	**10712.59**	**10662.29**	**10664.48**	**10652.00**
lrc201	1406.94	4	1491.96	1436.85	1705	1468.96	1407.21	1406.94	1406.94
lrc202	1374.27	3	1508.18	1403.33	2600	1374.27	1385.25	1374.27	1374.27
lrc203	1089.07	3	1231.43	1127.13	3502	1089.07	1093.89	1089.07	1089.07
lrc204	818.66	3	920.82	826.70	4415	827.78	818.66	818.66	818.66
lrc205	1302.20	4	1423.97	1326.12	1989	1302.20	1302.20	1302.20	1302.20
lrc206	1159.03	3	1253.95	1185.97	3000	1162.91	1159.03	1159.03	1159.03
lrc207	1062.05	3	1241.62	1091.31	3311	1424.60	1062.05	1062.05	1062.05
lrc208	900.89	3	1040.26	936.53	4274	852.76	852.76	865.51	852.76
Total lrc2	**9113.12**	**26**	**10112.19**	**9333.94**	**24797**	**9502.55**	**9081.05**	**9077.73**	**9064.98**
Total	**57662.02**	**434**	**61162.92**	**58302.08**	**95099**	**58184.91**	**57638.48**	**57678.40**	**57652.05**

[1] Total travel distance of best found solution

[2] Number of vehicles required for best found solution

[3] Average total distance travelled for 300 runs of the initial improvement phase

[4] Average total distance travelled for 30 runs of the branch and bound method using the best found solutions from the improvement phase

[5] Total computational time in seconds for one run of the algorithm including 300 runs of the initial improvement phase and 30 runs of the branch and bound method on the best found solutions

dramatically reduced compared to the instances with a shorter scheduling horizon, resulting in significantly more requests allocated to each vehicle. Therefore the problem now becomes one of finding the best ordering of requests to a route rather than the allocation of requests to routes. In the case of the lr2 instances this becomes increasingly difficult as locations are randomly dispersed, hence the success of a method which specifically focuses on optimising large portions of locations in routes such as our branch and bound method. Due to this our algorithm performs consistently well across the varying instance types whereas [8] and [12] struggle with the instances of a longer scheduling horizon, in particular lr2 and lrc2. The average coefficient of variation across each of the 6 classes of instances ranges from 1% to 7% for the results after the initial improvement phase and is reduced to less than 2% for all results after the branch and bound method.

For the instances lc104 and lrc101, a solution has been found by both [14] and [13] (and by [8] for the case of lrc101), that uses one less vehicle by increasing the total travel distance in the solution. This is the best found solution when the objective is to first minimise the number of vehicles, however these are the only two cases which do not share an identical best found solution. This shows the robustness of our algorithm to changes in the objective function as it also achieved the two solutions stated above but they were disregarded due to the increase in distance. Finally it should be noted that for [8] and [12] Euclidean distances calculated directly from the instances were rounded to 2 decimal places and this could account for some small discrepancies when comparing the total distance travelled.

6 Conclusions

We have shown that the methods applied in this paper generate results which are competitive with the state of the art results found in the literature. Our results obtain the best known solutions in 51 out of a possible 56 instances with the algorithm appearing to perform consistently well over all types of instance. One of the main advantages of our approach is the speed of individual constructions. In this case it has allowed us to produce large samples of solutions in times that are consistent with other approaches. However reducing the number of runs of the initial improvement phase still achieves promising results. This advantage can be exploited when applying these methods to the dynamic PDPTW (DPDPTW) where our algorithm will be repeatedly restarted over a rolling horizon framework to incorporate the arrival of new requests and decisions will need to be made in real time. The DPDPTW has received much less interest in the literature, hence this will be the area for future research. For a recent survey on dynamic pickup and delivery problems see [20].

References

1. Laporte, G., Osman, I.: Routing problems: A bibliography. Annals of Operations Research **61** (1995) 227–262

2. Savelsbergh, M.W.P., Sol, M.: The general pickup and delivery problem. Transportation Science **29**(1) (1995) 17–29
3. Psaraftis, H.: An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows. Transportation Science **17**(3) (1983) 351–357
4. Jaw, J., Odoni, A., Psaraftis, H., Wilson, N.: A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. Transportation Research Part B: Methodological **20** (1986) 243–257
5. Cordeau, J.F., Laporte, G.: The dial-a-ride problem: models and algorithms. Annals of Operations Research **153**(1) (2007) 29–46
6. Nanry, W., Barnes, J.: Solving the pickup and delivery problem with time windows using reactive tabu search. Transportation Research Part B: Methodological **34**(2) (2000) 107–121
7. Lau, H., Liang, Z.: Pickup and delivery with time windows: algorithms and test case generation. In: The 13th IEEE International Conference on Tools with Artificial Intelligence, ICTAI-2001, Dallas, USA. (2001) 333–340
8. Li, H., Lim, A.: A metaheuristic for the pickup and delivery problem with time windows. In: The 13th IEEE International Conference on Tools with Artificial Intelligence, ICTAI-2001, Dallas, USA. (2001) 160–167
9. Solomon, M.M.: Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research **35**(2) (1987) 254–265
10. Bent, R., Van Hentenryck, P.: A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. Computers & Operations Research **33**(4) (2006) 875–893
11. Ropke, R., Pisinger, D.: An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science **40**(4) (2006) 455–472
12. Pankratz, G.: A grouping genetic algorithm for the pickup and delivery problem with time windows. OR Spectrum **27** (2005) 21–41
13. Ding, G., Li, L., Ju, Y.: Multi-strategy grouping genetic algorithm for the pickup and delivery problem with time windows. In: GEC '09: Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation, New York, NY, USA, ACM (2009) 97–104
14. Dergis, U., Dohmer, T.: Indirect search for the vehicle routing problem with pickup and delivery and time windows. OR Spectrum **30** (2008) 149–165
15. Lim, H., Lim, A., Rodrigues, B.: Solving the pickup and delivery problem with time windows using "squeaky wheel" optimization with local search, AMCIS 2002 Proceedings (2002) 2335–2344
16. Carabetti, E., de Souza, S., Fraga, M.: An application of the ant colony system metaheuristic to the vehicle routing problem with pickup and delivery and time windows, Eleventh Brazilian Symposium on Neural Networks, 2010 (2010) 176–181
17. Or, I.: Traveling salesman-type combinatorial problems and their relation the logistics of regional blood banking. PhD thesis, Northwestern University, Evanston, IL. (1976)
18. Glover, F.: Tabu search part 1. ORSA Journal on Computing **1**(3) (1989) 190–206
19. Montané, F.A.T., Galvão, R.D.: A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. Computers and Operations Research **33**(3) (2006) 595–619
20. Berbeglia, G., Cordeau, J.F., Laporte, G.: Dynamic pickup and delivery problems. European Journal of Operational Research **202**(1) (2010) 8–15