A Metaheuristic for the Pickup and Delivery Problem with Time Windows

Haibing Li and Andrew Lim
Department of Computer Science
National University of Singapore
Singapore, 117543
{lihaibin, alim}@comp.nus.edu.sg

Abstract

In this paper, we propose a metaheuristic to solve the pickup and delivery problem with time windows. Our approach is a tabu-embedded simulated annealing algorithm which restarts a search procedure from the current best solution after several non-improving search iterations. The computational experiments on the six newly-generated different data sets marked our algorithm as the first approach to solve large multiple-vehicle PDPTW problem instances with various distribution properties.

1 Introduction

The class of Vehicle routing problems(VRP) is an intensive research area because of its usefulness to the logistics and transportation industry. Many new side constraints have been added to meet real life needs. A famous extension is the vehicle routing problem with time windows(VRPTW).

Another useful extension that has been proposed to VRPTW is to handle pickup and delivery where goods must be collected at a predetermined customer location before it is delivered to another specified customer location. Therefore, two additional side constraints are introduced. One is known as the *precedence* constraint and the other one is the *coupling* constraint. The two constraints require that any paired pickup and delivery locations must be serviced by the same vehicle and the pickup location must be scheduled before the corresponding delivery location in the route. Practical applications of this extension include the dial-a-ride problem, airline scheduling, bus routing, etc.

Let us define the Pickup and Delivery Problem with Time Windows (PDPTW) formally. Let $G = (V, A)$ be a digraph. $V = P \cup \{v_0\}$ is the node set where $P = \{v_i \in V | i = 1, 2, \ldots, n, n\ is\ even\}$ represents the customers and node v_0 denotes the depot where a fleet of vehicles are housed. In addition, let $P^+ \subset P$ be the set of pickup locations and $P^- \subset P$ be the set of delivery locations. Therefore, $P = P^+ \cup P^-, |P^+| = |P^-| = n/2$. Each node $v_i \in V$ has an associated customer demand $q_i (q_0 = 0)$, a service time $s_i (s_0 = 0)$ and a service-time window $[e_i, l_i]$. $q_i > 0$ for $v_i \in P^+$ and $q_i < 0$ for $v_i \in P^-$. For each pair of nodes $\langle v_i, v_j \rangle (i \neq j, i, j = 0, 1, 2, \ldots, n)$, a non-negative distance d_{ij} and a non-negative travel time t_{ij} are known. Due to the time window constraints, arcs may not exist between some node pairs. Therefore, the arc set can be defined as $A = \{\langle v_i, v_j \rangle | v_i, v_j \in V, v_i \neq v_j, t_{0i} + s_i + t_{ij} \leq l_j\}$. If a vehicle reaches a customer v_i before e_i, it needs to wait until e_i in order to service the customer. The schedule duration of a route is the sum of waiting time, service time and travel time. Depending on different contexts, the problem consists of minimizing several objectives subject to a variety of constraints. For transportation of goods, the objective involves minimizing the number of vehicles, travel costs and schedule duration. However, for dial-a-ride problems, it's preferable to minimize the inconvenience caused by pickups or deliveries performed earlier or later than desired time. In this paper, the objective of PDPTW is to service all customers without violating vehicle capacity, time window, precedence and coupling constraints with a minimum number of vehicles and, for the same number of routes with the minimum total travel distance, followed by the minimum total schedule duration and the minimum total waiting time.

Due to difficulty of PDPTW, most previous works focused on the single vehicle dial-a-ride problem with time windows(1-PDPTW) with slightly different objectives. For the objective to minimize the total customer inconvenience, Psarafits [5] [6] developed a dynamic programming algorithm with a $O(n^2 3^n)$ time complexity which can solve problems with only 10 or fewer requests. Sexton and Bodin [9] [10] solved the same problem by breaking it into a coordinating routing master problem formulated as an integer program, and a scheduling subproblem for a fixed route, which was formulated as linear program. By using a heuristic version of Benders' decomposition, the routing master problem and the scheduling subproblem were solved indi-

vidually. Results of real problems with sizes from 7 to 20 were reported. Sexton and Choi [11] used a similar approach to minimize a linear combination of total vehicle operating time and total customer penalty due to the violation of the time windows for the single vehicle pickup and delivery problem with soft time windows. For minimizing the schedule duration, Van der Bruggen *et al.* [12] developed a two-phase heuristic algorithm based on arc-exchange procedures and an alternative algorithm based on simulated annealing. Their approaches produced high quality solutions on real-life problems in reasonable computational time. Finally, for minimizing the total travel costs, a forward dynamic programming approach was developed by Desrosiers *et al.* [2]. The efficiency of the algorithm was improved by eliminating states that were incompatible with vehicle capacity, precedence and time window constraints. The multiple-vehicle pickup and delivery problem with time windows has received little attention. The only optimal algorithm developed by Dumas *et al.* [3] employed a column generation scheme with a shortest path subproblem with capacity, time window, precedence and coupling constraints. The algorithm can solve 1-PDPTW problems up to 55 paired requests and multiple-vehicle PDPTW with small number of paired requests per vehicle. Recently, Willian *et al.* [13] proposed a reactive tabu search approach to minimize travel cost by using a penalty objective function in terms of travel time, penalty for violation of overload and time window constraints. The approach was tested on instances with sizes of 25, 50 and 100 customers. These test cases were constructed from Solomon's $C1$ VRPTW benchmark instances (Solomon [7]) which were solved optimally.

In this paper, we propose a tabu-embedded simulated annealing approach to solve the general multiple-vehicle PDPTW. In addition, test cases are generated from all of the Solomon's benchmark instances for VRPTW.

The paper is organized as follows. Section 2 describes the local search structures. Section 3 presents the metaheuristic. The time complexity of the algorithms is analyzed in Section 4. Section 5 reports computational results for 56 test cases we generated from Solomon's benchmark instances for VRPTW. Finally, we present our conclusion in Section 6.

2 Local Search Structures

2.1 Notations

N, M: total number of customers, total number of vehicles respectively.

$MSNI$: max number of iterations without improvement in annealing procedure.

$Cost(S)$: objective function of solution *S* of local search.

$SACost(S)$: objective function of solution *S* in tabu-embedded METROPOLIS_PROCEDURE in the simulated annealing algorithm.

$Dist(S), ST(S), WT(S)$: total travel distance, total schedule time and total waiting time of solution *S* respectively.

$N_{PDS}(S), N_{PDE}(S), N_{PDR}(S)$: neighborhood of solution *S* obtained by PD-Shift operator, PD-Exchange operator and PD-Rearrange operator respectively.

S_b: local optimal solution.

K: the maximum number of children in a *K*-ary tree.

ϕ: the depth of a *K*-ary tree.

T_0, T: initial and global annealing temperature respectively.

δ: cooling ratio of *T*.

$\alpha, \beta, \gamma, \lambda$: penalty factor for *M*, *Dist(S)*, *ScheTime(S)*, *Dist(S)* respectively.

2.2 Route Construction Heuristics

Insertion Heuristic by Solomon [7] is modified to construct the initial solution. During the insertion procedure, all PDPTW constraints must be satisfied. The procedure first initializes a route with a pickup-delivery(PD) paired customers, using criteria of maximal combined farthest distances to depot, minimal combined latest bound of time windows and minimal combined period of time windows. Then for each of the unrouted PD-pairs, their best feasible insertion positions in the current partially-constructed route are computed. The PD-pair which causes minimal increment in travel cost is selected and inserted into the route. Thus after all the PD-pairs are inserted, a feasible solution is obtained.

2.3 Neighborhood Structures

In our method, three PD-pair swapping operators are used to define local search neighborhoods. These operators are PD-Shift operator, PD-Exchange operator and PD-Rearrange operator. The first two operators are used for generating neighborhoods for metaheuristic guided local search procedure, while the PD-Rearrange operator is only used for post-optimization purpose.

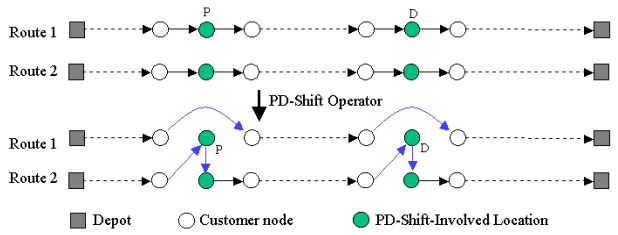

Figure 1. PD-Shift Operator

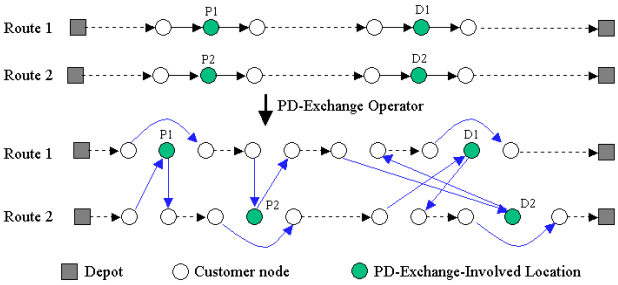

Figure 2. PD-Exchange Operator

2.3.1 PD-Shift Operator

The PD-Shift operator moves PD-pairs from one route to another, subject to all the constraints imposed on PDPTW. For each pair of selected routes, say, *Route1* and *Route2*, the PD-Shift operator is used twice, for shifting PD-pairs from *Route1* to *Route2* and from *Route2* to *Route1*. We have denoted the neighborhood of solution *S* obtained by the PD-Shift operator as $N_{PDS}(S)$. In Figure 1, locations *P* and *D* are originally a PD-pair in *Route1*. The PD-Shift operator first removes the PD-pair from *Route1* and then inserts them into a feasible position-pair in *Route2*. Infeasible shifts are forbidden with regards to the PDPTW constraints.

2.3.2 PD-Exchange Operator

The PD-Exchange operator swaps PD-pairs of two routes, subject to all the constraints imposed on PDPTW. We have denoted the neighborhood of solution *S* obtained by the PD-Exchange operator as $N_{PDE}(S)$. In Figure 2, locations *P1* and *D1* are originally a PD-pair in *Route1*, while locations *P2* and *D2* are originally a PD-pair in *Route2*. The PD-Exchange operator first removes the two PD-pairs from *Route1* and *Route2* respectively, then insert *P1-D1* into a feasible position-pair in *Route2*, while insert *P2-D2* into a feasible position-pair in *Route1*. Again, infeasible exchanges are forbidden with regards to the PDPTW constraints.

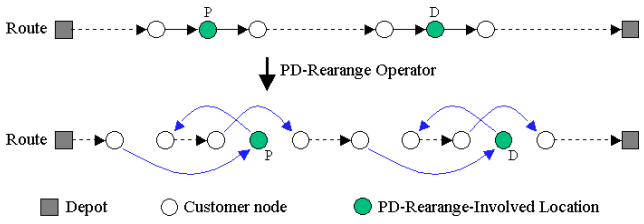

Figure 3. PD-Rearrange Operator

2.3.3 PD-Rearrange Operator

Within the same route, the PD-Rearrange operator rearranges PD-pairs to the best positions that maximally reduce the value of the objective function. We have denoted the neighborhood of solution *S* obtained by the PD-Rearrange operator as $N_{PDR}(S)$. In Figure 3, locations *P* and *D* are a PD-pair in *Route*. The PD-Rearrange operator removes the PD-pair from *Route*, then inserts them into a new feasible position-pair within the same route. Infeasible rearrangements are also forbidden with regards to the PDPTW constraints.

2.3.4 Objectives and Cost Function in Local Search

In our research, we prefer the priority order of PDPTW's objectives as follows: i) to minimize the number of vehicles; ii) to minimize the total travel cost; iii) to minimize the total schedule duration; iv) to minimize the drivers' total waiting time to start service. To reflect this order of priority, we adopt the following cost function:

$$Cost(S) = \alpha M + \beta Dist(S) + \gamma ST(S) + \lambda WT(S) \quad (1)$$

The penalty weight factors are set to be $\alpha \gg \beta \gg \gamma \gg \lambda$. Under this setting, even if the initial solutions obtained by the construction heuristic contains large number of vehicles, the number of vehicles will be reduced during the search procedure.

2.3.5 Descent Local Search

As an extension of local search, descent local search(DSL) starts from an initial solution. All the solutions in the neighborhoods are checked. The best solution with the minimum objective cost assessed by (1) will act as the initial solution for repeating this DSL procedure. This is repeated until no improvement is found.

INPUT: a solution *S*;
OUTPUT: a local optimal solution S_b
DLS(Solution S)

1. $S_b \leftarrow S$

2. Select the best solution $S_b^{'}$ with minimum objective cost within a defined neighborhood of S_b

3. **IF** $Cost(S_b^{'}) < Cost(S_b)$ **THEN**

3.1 $S_b \leftarrow S_b^{'}$; **GOTO** Step 2

4. **ELSE RETURN** S_b

3 Our Metaheuristic

Metaheuristics are strategies that guide local search procedures. In this section, we present a metaheuristic that guides the DSL procedure.

3.1 Tabu Structure

We define the following eigenvalue structure to represent a solution:

```
Struct Eigenvalue{
  Number of vehicles (NV);
  Total travel cost (TC);
  Total schedule duration (SD);
  Total waiting time (WT);
}
```

NV, TC, SD and WT will be used as table fields in later sections.

Since the probability that two different solutions having the same eigenvalue is very small, it's reasonable to regard two solutions as the same if they share the same eigenvalue. We record the eigenvalues of the solutions into a tabu set which differs from previous works that use edge-moves in their tabu structure.

3.2 K-Restarts Strategy

In our algorithm, we use a simulated annealing algorithm which differs from its traditional implementation. Instead of performing the simulated annealing procedure on the probabilistically accepted solution repeatedly until the procedure terminates, we compel the simulated annealing procedure to restart from the current best solution after several simulated annealing iterations without any improvement (the constant $MSNI$ can be set to 3 or 4 to produce good results). The algorithm terminates after K restarts without improvement. This acts like a rightist K-ary tree.

In addition to this K-restarts strategy, the visited solutions are recorded into a tabu set to avoid cycling. The hybrid metaheuristic is given below:

INPUT: a solution *S*;
OUTPUT: a local optimal solution S_b
TABU_EMBEDDED_SA(Solution S)

1. Obtain a solution S_b from **DSL** within $N_{PDS}(S) \cup N_{PDE}(S)$

2. $S_b \leftarrow$ Perform a **DSL** within $N_{PDR}(S)$

3. $NoImpr \leftarrow 0$

4. $S \leftarrow S_b$

5. **WHILE** $NoImpr < MSNI$ **DO**

5.1 Use **METROPOLIS_PROC(S)** to obtain a feasible solution S' which was not recorded in tabu set, from the neighborhood $N_{PDS}(S) \cup N_{PDE}(S)$, record the two routes on which PD-Shift/PD-Exchange operator performed.

5.2 Reduce number of vehicles by moving PD-pairs out from one route and insert them into other routes based on criteria of minimal increment in objective cost.

5.3 Use Solomon's Insertion Procedure to respectively re-order the two recorded routes of S' in step 5.1.

5.4 $S_b^{'} \leftarrow$ Perform a **DSL** within $N_{PDS}(S') \cup N_{PDE}(S')$

5.5 $S_b^{'} \leftarrow$ Perform a **DSL** within $N_{PDR}(S_b^{'})$

5.6 **IF** $Cost(S_b^{'}) < Cost(S_b)$ **THEN**

5.6.1 $S_b \leftarrow S_b^{'}$; $NoImpr \leftarrow 0$

5.7 **ELSE** $NoImpr \leftarrow NoImpr + 1$

5.8 $S \leftarrow S_b^{'}$

6. **RETURN** S_b

METROPOLIS_PROC(Solution S)

1. **WHILE NOT** yet accept an neighboring solution of *S* **DO**

1.1 $S' \leftarrow$ get a random neighbor of *S* which is not in tabu set, within $N_{PDS}(S') \cup N_{PDE}(S')$

1.2 $\Delta \leftarrow SACost(S') - SACost(S)$

1.3 **IF** $\Delta \leq 0$ **THEN** $prob \leftarrow 1$

1.4 **ELSE** $prob \leftarrow e^{-\frac{\Delta}{T}}$

1.5 **IF** $random(0,1) \leq prob$ **THEN**

1.5.1 Accept S'

1.5.2 Update global annealing temperature: $T \leftarrow \delta T$

1.5.3 Record S' into tabu set

2. **RETURN** S'

In METROPOLIS_PROC, the cost function SACost(S) is defined as:

$$SACost(S) = Dist(S) + \varphi \times WT(S) \qquad (2)$$

Here, $\varphi = 0.01 \times Dist(S)$. In procedure TABU_EMBEDDED_SA, from the viewpoint of tabu search, the METROPOLIS_PROC in step 5.1 acts as an annealing-probabilistic diversification strategy to escape from local optima. In addition, Step 5.3 and S tep 5.5 act as intensification strategies. In fact, the best-based K-restarts strategy in the following overall algorithm itself acts as an intensification strategy.

INPUT: problem data;
OUTPUT: a best solution found S_b found
OVERALL_ALGORITHMS

1. Input problem data

2. $S_b \leftarrow$ the better solution produced from Modified Solomon's Insertion Algorithm

3. Configure parameters: K, T_0, T, δ and $MSNI$

4. Set tabu set to be empty

5. $gNoImpro \leftarrow 0$

6. **WHILE** $gNoImpro < K$ **DO**

6.1 $S \leftarrow S_b$

6.2 $S_b^{'} \leftarrow TABU\_EMBEDDED\_SA(S)$

6.3 **IF** $Cost(S_b^{'}) < Cost(S_b)$ **THEN**

6.3.1 $S_b \leftarrow S_b^{'}$; $gNoImpro \leftarrow 0$

6.4 **ELSE** $gNoImpro \leftarrow gNoImpro + 1$

7. **OUTPUT** S_b

4 Analysis of the Algorithms

In the above overall algorithm, *K* determines the stopping condition of the overall algorithm. The total number of iterations of local search guided by the tabu-embedded simulated annealing procedure will be $O(\phi K)$, where ϕ is the depth of the rightist *K*-ary tree due to K restarts.

In each iteration, the time complexity of the overall algorithm is mainly determined by the time complexity of local searches in the neighborhoods that we defined. Let $T(m,n)$ be the overall time complexity of local searches in the neighborhoods. The ti me complexity of the whole algorithm is $O(\phi K) \times T(m,n)$.

On average, each vehicle will serve $\lceil \frac{n}{m} \rceil$ customers, that is, $\lceil \frac{n}{2m} \rceil$ PD- pairs. For the PD-Shift Operator, once a pickup location *P* in *Route 1* is selected for swapping, it takes $O(\lceil \frac{n}{m} \rceil)$ time to locate its delivery counterpart *D* within the same rou te. Similarly, once a insertion position in *Route 2* is determined for *P*, it also take $O(\lceil \frac{n}{m} \rceil)$ time to determine another insertion position in *Route 2* for *D*. In addition, there are $O(m^2)$ route-pair combinations. Therefore, the time complexity for local search in $N_{PDS}(S)$ is $O(\frac{n^4}{m^4} \times m^2) = O(\frac{n^4}{m^2})$

Repeating similar analysis for $N_{PDE}(S)$, the time complexity of a local search in $N_{PDE}(S)$ defined by PD- Exchange operator is $O(\frac{n^6}{m^6} \times m^2) = O(\frac{n^6}{m^4})$.

The rearrange operator is only used for post-optimization in the procedure of TABU_EMBEDDED_SA. In addition, it is only applied to the affected routes which the PD-Shift operator or PD-Exchange operator were applied. As the a bove analysis, time complexity of a local search in $N_{PDR}(S)$ is $O(\frac{n^4}{m^4} \times 2) = O(\frac{n^4}{m^4})$. Therefore, $T(m,n) = max[O(\frac{n^4}{m^2}), O(\frac{n^6}{m^4}), O(\frac{n^4}{m^4})]$, since *m* will never be larger than *n*, we get $T(m,n) = O(\frac{n^6}{m^4})$. It follows that the time complexity of the whole algorithm is $O(\phi K \frac{n^6}{m^4})$. As long as K is kept to a small number, our running time is reasonable.

5 Experimental Results

5.1 Benchmark Problem Instances

We generated from Solomon's 56 benchmark problem instances (See Solomon [7]), by randomly pairing up the customer locations within routes in solutions obtained by our heuristic approach for the vehicle routing problem with time windows (See Li *et al.* [4]). This is different from the way that William *et al.* [13] just paired up customers appearing in the routes of the optimal solutions one by one. In addition, we were not restricted to generating data sets using only optimal solutions. One reason is that most of the Solomon's instances have not yet been optimally solved. Another reason is that the PD-pairs may be much more randomly distributed in real life problems, so they may not necessarily be paired up within the same route of VRPTW solutions. Corresponding to Solomon's classification of $C1$, $C2$, $R1$, $R2$, $RC1$ and $RC2$, our data sets were also generated in six classes: $LC1$, $LC2$, $LR1$, $LR2$, $LRC1$ and $LRC2$. All problems have 100 real customers with several additional dummy nodes for coupling purpose if necessary, a central depot, capacity constraints, time window constraints, precedence constraints together with coupling constraints. The customers in LC problems are clustered. In LR problems the customers are randomly distributed. In LRC problems, customers are partially clustered and partially randomly distributed. The $LC1$, $LR1$ and $LRC1$ problems have short scheduling horizon, while $LC2$, $LR2$ and $LRC2$ have longer scheduling horizon.

Prob.	*NV*	*TC*	*SD*	*CT*	WV	WSD
NC101	10	828.9	9828.9	35	10	9827.3
NC102	10	828.9	9828.9	130	10	9827.3
NC103	10	831.9	10065.1	156	10	9829.9
NC104	9	869.5	10166.7	1539	10	9834.7
NC105	10	828.9	9828.9	25	10	9827.3
NC106	10	828.9	9828.9	31	10	9827.3
NC107	10	827.8	9921.8	22	10	9826.1
NC108	10	827.8	9904.8	39	10	9826.1
NC109	10	827.8	9831.8	85	10	9827.3

Table 1. Results Obtained for $NC1$ Instances

5.2 Computational Results

Our experimental environment is the Linux Kernel 2.2.14-5.0 smp on i686. The algorithms were coded in C++ with Standard Template Library. The values of the parameters used are $T_0 = 50, \delta = 0.95, K = \lceil \frac{20M}{3} \rceil, MSNI = 4$.

5.2.1 Results for 9 $NC1$ problem instances

Table 1 shows the results obtained by our algorithms and by reactive tabu search method of William et al. [13] on the 9 $NC1$ problems. Although our objective functions are different, a rough comparison table is given. Here, CT means CPU time in seconds, while WV and WSD represent NV and SD obtained by William *et al.* [13]. William *et al.* [13] obtained 7 optimal solutions out of the 9 instances except $NC103$ and $NC104$, by their reactive tabu search heuristic. These are very good solutions. As far as we know, Chiang *et al.* [14] also solved VRPTW using the same approach, but they did not reach optimal solutions for any $C1$ instances. In addition, no heuristic has ever reported optimal solutions for $C1$ test cases. For $NC104$, our solution has a smaller number of vehicles. By comparison, our schedule durations are slightly larger than those of William *et al.* [13].

5.2.2 Results for the newly-generated 56 problem instances

It's true that William et al. [13] is the first to solve 100-customer instances, however, their experiments were confined within only 9 instances with clustered distribution property. Because of this reason, we conduct more experiences on 56 newly-generated instances with various distribution properties. Table 2 shows the best results obtained by our algorithms. The table lists all the 4 objective values in order of their priorities togther with computational times.

Prob.	*NV*	*TC*	*SD*	*WT*	*CT*
LC101	10	828.94	9828.94	0	33
LC102	10	828.94	9828.94	0	71
LC103	10	827.86	10058.03	230.17	191
LC104	9	861.95	10006.90	144.95	1254
LC105	10	828.94	9828.94	0	47
LC106	10	828.94	9828.94	0	43
LC107	10	828.94	9828.94	0	54
LC108	10	826.44	9826.44	0	82
LC109	10	827.82	9831.78	3.96	255
LC201	3	591.56	9591.56	0	27
LC202	3	591.56	9591.56	0	94
LC203	3	585.56	9521.66	26.09	145
LC204	3	591.17	9591.17	0	746
LC205	3	588.88	9588.88	0	190
LC206	3	588.49	9588.49	0	88
LC207	3	588.29	9660.40	72.12	102
LC208	3	588.32	9744.23	155.91	178
LR101	19	1650.78	3599.45	948.65	87
LR102	17	1487.57	3202.46	714.89	1168
LR103	13	1292.68	2729.15	436.48	169
LR104	9	1013.39	2050.85	37.46	459
LR105	14	1377.11	2631.56	254.45	69
LR106	12	1252.62	2412.11	159.49	87
LR107	10	1111.31	2220.27	108.96	287
LR108	9	968.97	2029.93	60.96	415
LR109	11	1239.96	2311.37	71.42	348
LR110	10	1159.35	2222.99	63.64	547
LR111	10	1108.90	2189.53	80.63	179
LR112	9	1003.77	2013.17	9.41	638
LR201	4	1263.84	3495.81	1231.97	193
LR202	3	1197.67	2749.39	551.73	885
LR203	3	949.40	2762.80	813.40	1950
LR204	2	849.05	1988.57	139.52	2655
LR205	3	1054.02	2550.13	496.11	585
LR206	3	931.63	2502.46	570.84	747
LR207	2	903.056	1936.44	33.38	1594
LR208	2	734.85	1858.36	123.51	3572
LR209	3	937.05	2432.61	459.56	2773
LR210	3	964.22	2782.06	817.83	1482
LR211	2	927.80	1954.57	26.77	4204
LRC101	14	1708.80	2956.32	247.52	119
LRC102	13	1563.55	2764.14	200.59	152
LRC103	11	1258.74	2443.68	184.95	175
LRC104	10	1128.40	2238.17	109.77	202
LRC105	13	1637.62	2830.16	192.54	179
LRC106	11	1425.53	2475.01	49.48	459
LRC107	11	1230.15	2344.94	114.80	154
LRC108	10	1147.97	2245.30	97.34	650
LRC201	4	1468.96	3358.41	889.45	266
LRC202	3	1374.27	2657.14	282.87	987
LRC203	3	1089.07	2700.30	611.23	1605
LRC204	3	827.78	2537.83	710.05	3634
LRC205	4	1302.20	3464.61	1162.41	639
LRC206	3	1162.91	2444.87	281.96	445
LRC207	3	1424.60	2461.42	36.82	607
LRC208	3	852.76	2271.19	418.44	4106

Table 2. Solutions for the new instances

Data Sets	*MNV*	*MTC*	*MSD*	*MWT*
NC1	9.89	833.40	9911.77	78.37
LC1	9.89	832.08	9874.20	42.12
LC2	3.00	589.23	9609.74	31.77
LR1	11.92	1222.20	2467.74	245.54
LR2	2.73	977.14	2455.75	478.60
LRC1	11.63	1387.59	2537.22	149.62
LRC2	3.25	1187.82	2736.97	549.15

Table 3. Statistical summary for the data sets

5.2.3 Statistical summary for the data sets

The statistical computational results are shown in Table 3. Here, MNV, MTC, MSD and MWT represents the mean values of NV, TC, SD and WT respectively.

5.2.4 Route details for several selected instances

Table 4 and Table 5 show the route details for several selected instances. The node indices larger than 100 indicate that these are dummy nodes for pairing purposes. NC means number of customers in the route of a vehicle.

6 Conclusion

In this paper, we proposed a metaheuristic with annealing like restart strategy to guide the local search in three neighborhoods that we defined to solve the general multiple-vehicle PDPTW. This is combined with an effective metaheuristic based on a K restarts annealing procedure with tabu-list to avoid cycling in the search process. In addition, we generated 56 100-customer problem instances from Solomon's benchmark instances for the vehicle routing problem with time windows. The experimental results on the six different data sets marked our algorithms as the first efficient approach to solve practical sized multiple-vehicle PDPTW problem instances with various distribution properties. Furthermore, our algorithm can be easily adapted to solve further generalizations of PDPTW.

References

[1] J.Desrosiers, Y.Dumas, M.M.Solomon, and F.Sormis, "Time Constrained Routing and Scheduling," in *Handbooks in Operations Research and Management Science: Network Routing*, M.O.Ball, T.L.Magnanti, C.L.Monma and G.L.Nemhauser Elsevier(eds), 35–139, Elsevier, Amsterdam, 1995.

[2] Y.Dumas, J.Desrosiers, F.Soumis, "A dynamic programming solution of the large-scale single vehicle

Prob.	*NC*	*Details of routes*
NC104	12	13-17-18-19-15-16-28-23-26-22-20-24
	12	25-27-44-46-45-48-51-50-52-49-47-105
	10	81-78-76-71-70-73-77-79-80-63
	10	67-65-62-74-72-61-64-68-66-69
	12	5-3-7-8-11-9-6-4-2-1-75-104
	14	98-96-95-94-92-93-97-100-99-101-14-12-10-103
	12	57-55-54-53-56-58-60-59-41-40-43-42
	14	32-33-31-35-37-38-39-36-34-102-29-30-21-106
	10	90-87-86-83-82-84-85-88-89-91
		$TD = 869.53, NV = 9$
LR106	8	65-71-51-9-35-34-3-77
	6	63-64-11-90-10-31
	10	48-47-36-19-49-46-102-82-7-52
	8	62-88-30-20-66-32-70-1
	8	83-45-18-8-84-17-5-60
	10	21-72-39-23-67-55-54-4-25-101
	10	59-37-14-44-38-86-43-100-98-93
	10	50-33-29-79-81-78-68-24-80-12
	8	96-85-91-16-61-99-6-89
	10	28-26-73-41-22-75-56-74-2-58
	10	94-92-42-15-57-87-97-104-95-13
	6	27-103-69-76-40-53
		$TD = 1252.62, NV = 12$
LR204	50	27-52-18-7-82-48-19-11-62-88-31-69-76-12-67-39-53-28-79-81-9-51-20-66-65-71-35-34-78-29-24-55-4-72-74-73-21-2-13-97-37-98-100-91-85-93-59-96-6-89
	50	94-95-92-42-57-22-75-56-23-41-15-43-14-44-38-86-16-61-99-87-60-83-8-84-5-17-45-46-47-36-49-64-63-90-32-10-30-70-1-50-33-3-77-68-80-54-25-26-40-58
		$TD = 849.05, NV = 2$
LR211	50	95-2-21-72-75-23-67-39-12-76-29-79-33-81-9-65-71-51-30-90-63-64-49-36-47-48-46-8-45-84-5-6-94-96-59-37-97-13-58-26-74-56-4-25-55-54-24-80-68-77
	50	28-27-52-69-31-88-62-11-19-7-82-18-83-99-85-61-16-86-38-14-44-98-92-87-42-43-15-57-41-22-73-40-53-3-78-34-35-66-20-32-10-50-1-70-60-17-91-100-93-89
		$TD = 927.80, NV = 2$

Table 4. Details of selected solutions

Prob.	*NC*	*Details of routes*
LRC104	12	14-47-15-11-9-87-59-74-102-86 -57-52
	12	66-106-20-49-19-23-21-48-18-25 -24-22
	8	85-63-89-76-84-51-64-83
	10	69-98-53-12-17-16-13-108-10-82
	12	42-43-44-38-37-35-36-40-39-41- 72-54
	12	61-70-1-3-5-45-101-8-46-4-100- 68
	8	90-65-99-97-75-58-103-77
	12	80-91-56-95-62-67-94-105-93-71 -96-81
	12	92-50-33-32-107-30-28-26-27-29 -31-34
	10	88-60-78-73-79-7-6-2-55-104 $TD = 1128.40, NV = 10$
LRC108	10	65-83-57-24-22-49-21-23-25-77
	10	41-42-44-43-40-38-36-35-37-39
	10	69-88-61-81-94-71-72-54-96-68
	12	92-95-102-67-62-50-63-85-84-56 -91-80
	10	64-51-76-89-18-48-19-20-101-66
	10	82-99-52-86-87-59-97-75-58-74
	10	12-14-47-17-16-15-13-9-11-10
	10	90-53-60-7-104-6-2-70-100-55
	12	98-78-73-79-8-46-5-3-1-103- 45-4
	10	33-32-30-28-26-27-29-31-34-93 $TD = 1147.97, NV = 10$
LRC202	38	92-95-85-63-33-28-26-27-29-31- 30-62-67-71-72-38-40-41-81-90- 91-84-49-20-83-66-56-50-34-32- 89-24-21-25-77-75-58-102
	32	65-82-12-14-47-15-11-69-64-19- 23-48-18-76-51-22-86-87-9-57- 52-10-97-59-74-13-17-7-4-60- 100-70
	32	98-45-5-3-1-42-36-37-39-44-61- 88-53-78-79-8-6-46-2-55-68-101 -43-35-54-96-93-94-80 $TD = 1374.27, NV = 3$
LRC205	22	92-95-33-28-27-29-31-30-63-76- 85-67-84-51-49-22-20-24-74-13- 17-60
	28	65-83-64-19-23-21-18-57-86-52- 99-9-87-59-75-97-102-10-55-68- 43-35-37-54-96-93-91-80
	30	2-45-5-42-39-36-72-71-62-94-61 -44-40-38-41-81-90-53-82-66-56 -50-34-32-26-89-48-25-77-58
	22	69-98-11-15-16-47-14-12-88-78- 73-79-7-6-8-46-4-3-1-70-100-101 $TD = 1302.20, NV = 4$

Table 5. (Continued) Details of selected solutions

dial-a-ride problem with time windows", *American Journal of Mathematical and Management Science* 16, 301–325 (1986).

[3] Y.Dumas, J.Desrosiers and F.Soumis, "The pick-up and delivery problem with time windows", *European Journal of Operational Research* 54, 7–22 (1991).

[4] H.Li, A.Lim and J.Huang, "Local Search with Annealing-like Restarts to solve the VRPTW", Research Paper, Department of Computer Science, National University of Singapore, (2001).

[5] H.Psarafis, "A dyanmic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem", *Transportation Science* 14, 130–154 (1980).

[6] H.Psarafis, "An exact algorithm for the single vehicle many-to-many immediate request dial-a-ride problem", *Transportation Science* 17 (4), 351–361 (1983).

[7] M.M.Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constrains", *Operations Research* 35, 254–264 (1987).

[8] M.W.P.Savelsbergh, M.Sol, "The general pickup and delivery problem", *Transportation Science* 29 (1), 107–121 (1995).

[9] T.R.Sexton, D.B.Lawrence, "Optimizing single vehicle many-to-many dial-a-ride problem with desired delivery time: I Scheduling", *Transportation Science* 19, 378–410 (1985).

[10] T.R.Sexton, D.B.Lawrence, "Optimizing single vehicle many-to-many dial-a-ride problem with desired delivery time: II Routing", *Transportation Science* 19, 411–435 (1985).

[11] T.R.Sexton, Y.-Y.Choi, "Pickup and delivery partial loads with "soft" time windows", *American Journal of Mathematical and Management Science* 6, 369–398 (1986).

[12] L.J.J.Van der Bruggen, J.K.Lenstra, P.C.Schuur, "Variable-depth search for the single vehicle pickup and delivery problem with time windows", *Transportation Science* 27, 298–311 (1993).

[13] P.N.William, J.W.Barnes, "Solving the pickup and delivery problem with time windows using tabu search", *Transportation Research Part B* 34, 107–121 (2000).

[14] W.-C.Chiang and R.Russel, "A reactive tabu search metaheuristic for the vehicle routing problem with time windows", *INFORMS Journal on Computing* 9, 417–430 (1997).