A New Two-phase Hybrid Metaheuristic for Vehicle Routing Problem with Time Windows

Mingyao QI[a], Ying ZHANG[b], Jinjin ZHANG[c], Lixin MIAO[d]

[a,b,c,d]*Graduate School at Shenzhen, Tsinghua University, Shenzhen, 518055, China*
[a]*E-mail: qimy@sz.tsinghua.edu.cn*
[b]*E-mail: zhang.ying111@alum.sz.tsinghua.edu.cn*

Abstract: This paper proposes a new two-phase hybrid metaheuristic for vehicle routing problem with time windows (VRPTW). The first phase is to minimize the number of routes by means of variable neighborhood search algorithm, while the second phase is mainly aimed at minimizing the total travel distance using tabu search algorithm. Three neighborhood search operators (*All-exchange*, *All-2-opt*, *All-crossexchange*) and two local serarch operators (*All-relocate* and ejection chain) are designed. To further lower the number of vehicles, the sum-of-squares route sizes is maximized in the first phase. A comparative test is implemented by our algorithm on the basis of 56 benchmark problems proposed by Solomon (1987), the mean number of vehicles and running time of this algorithm is very competitive comparing to previous metaheuristics, showing that the new two-phase hybrid metaheuristic is effective and fast.

Keywords: Vehicle Routing Problem, Time Windows, Metaheuristic, Variable Neighborhood Search, Tabu Search

1. INTRODUCTION

The Vehicle Routing Problem (VRP) is a well-known combinatorial optimization problem which focuses on the optimal arrangement or schedule of a fleet of vehicles while serving scattered customers. When customer has his own time period and only during this period the service can be accepted, the problem is called Vehicle Routing Problem with Time Windows (VRPTW). First introduced in 1959 (Dantzig *et al.*, 1959), it is still a very important problem that is broadly studied now because of its widespread use and hardness. Even the simplest form of VRP is Strongly NP-Complete. Thus, heuristic algorithms are always introduced to get a good solution.

Basically, three kinds of VRP algorithms can be found from previous studies, exact algorithms, classic heuristics, and metaheuristics (Laporte, 2007). Though exact algorithms are often developed to obtain global optimal solutions, the most sophisticated exact algorithms for the VRP can only solve instances up to about 100 customers, and with a varied success rate (Baldacci *et al.*, 2008). This explains to a large extent why most of the research effort has been directed to heuristics and metaheuristics. For a complete survey of VRPTW, see Braysy and Gendreau (2005a, 2005b).

Construction heuristics is a kind of algorithm that adds each customer to existing routes step by step according to greedy strategies. Although it computes very quickly, the quality of the solutions is relatively poor. Improvement heuristics (local search) starts its search procedure from an initial feasible solution that obtained by a construction heuristics. By

searching the neighborhoods of current solution iteratively, the best solution is updated until no improvements can be gained. The methods of generating neighborhood solution are based on swap or reinsert strategies, such as exchange, crossexchange, 2-OPT, 3-OPT, Or-OPT, etc. Since the construction heuristics always seeks better neighborhood, it will easily fall into local optima. The metaheuristics overcome this shortage, it can guide the search procedure to escape from local optima and jump to another solution space. Therefore, it can gain better solutions.

Over the past decade, a few new metaheuristics for VRPTW were published which can improve the computational results of some benchmark problems. Although it is very hard to further improve the results of this classical problem, we still try in this paper to present a new algorithm which contains a few computation tricks, the results look promising.

2. LITERATURE REVIEW

As a classical combinatorial optimization problem, VRP was studied extensively in a great quantity of literatures. For recent review papers, refer to Laporte (2007), Bräysy and Gendreau (2005a, 2005b). Before 2005, Tabu search algorithm was thought as one of the most efficient algorithm to solve VRP. Tabu search was first introduced by Glover (1986). Contrary to classical descent methods, it can accept poorer solution from one iteration to another to escape from local optima. Renaud and Laporte *et al.* (1996) describes a tabu search algorithm for the multi-depot vehicle routing problem with capacity and route length restrictions. The algorithm is tested on a set of 23 benchmark instances. It is shown to outperform existing heuristics at that time. Badeau and Guretin *et al.* (1997) proposed a parallel tabu search algorithm for VRPTW which can reduce calculation time. Chiang and Russell (1997) proposed a reactive tabu search algorithm which can run out of local optima by changing the tabu list length dynamically. It renewed several best solutions. For review of tabu search algorithms of VRPW before 2002, refer to Bräysy and Gendreau (2002). More recently, Fu and Eglese (2005) proposed a new tabu search algorithm for another version of the vehicle routing problem (VRP)-the open vehicle routing problem.

Genetic algorithm and simulated annealing are also effective metaheuristics to solve VRPTW, however, here we just review some new papers that using Variable neighborhood search (VNS). VNS was first proposed by Mladenovic (1997) and Hansen (1997). As a local search based metaheuristic, VNS behaves excellent in solving NP hard problems. During the past few years, the VNS approach has been applied to variants of the VRPs efficiently. Bräysy (2003) presented a modification of VNS for solving VRPTW, named Reactive Variable Neighborhood Search (RVNS). Polacek *et al.* (2004) proposed a VNS heuristic to tackle the basic capacitated vehicle routing problem (CVRP) and achieved new best solutions in 71 cases. Goel and Gruhn (2006) proposed iterative improvement approaches based on the idea of changing the neighborhood structure during the search to solve the General Vehicle Routing Problem (GVRP). Kytojoki *et al.* (2007) presented an efficient VNS heuristic, which is specifically aimed at solving very large scale real-life vehicle routing problems. Polacek *et al.* (2004) proposed a VNS heuristic to solve Multi-Depot VRPTW and compared it with a tabu search algorithm. Hemmelmayr *et al.* (2009) also proposed a new heuristic based on VNS for the Periodic Vehicle Routing Problem (PVRP) without time windows, and the computational results testified the excellent competitiveness of their approach. Qi and Li *et al.* (2010) proposed a VNS for large scale real-time time-dependent vehicle routing problem with time windows.

Since the objective of VRPTW have two fold: minimizing the number of vehicles as well as the total travel distance, and the former is set as the primary criteria, a few papers adopted a two stage strategy to solve the problem. In the first stage, the minimum number of vehicles is calculated, and the total travel distance is optimized in the second phase.

Recently, several scholars obtained good results using two-phase hybrid metaheuristics. Gehring and Homberger (2002, 2005) proposed a parallel two-phase metaheuristic and a two-phase hybrid metaheuristic respectively for VRPTW. The first phase is aimed at minimizing the number of vehicles by means of evolution strategies and the second phase is to minimize the total travel distance using tabu search algorithm. Russell, Hentenryck (2006) proposed a two-phase hybrid metaheuristic for VRP with pick-up and delivery. It adopted simulated annealing in the first phase and large neighborhood search in the second phase. The above two-phase metaheuristics all renewed some best solutions of enchmark problems (Solomon 1987), showing that it's an effective way.

In this paper we also adopt two-phase strategy and design a hybrid metaheuristic for VRPTW. VNS is used in the first phase to decrease the number of routes, while tabu search is used in the second phase with the main objective of minimizing the total travel distance. The result of the first phase will serve as the initial solution in the second phase. The algorithm details of the two phases will be introduced in the next section respectively, following by an evaluation and comparison to previous research based on Solomon's benchmark problems.

3. SOLUTION FRAMEWORK

3.1 Variable Neighborhood Search

As mentioned above, the main objective of VNS is to lower the number of routes, thus its sub procedures should reduce the vehicles to the greatest extent. VNS is chosen to find a feasible solution that uses minimum vehicles. This is because a few neighborhoods are specially designed to eliminate unnecessary routes. Variable neighborhood search (VNS) is a recent metaheuristic for solving combinatorial and global optimization problems. Its basic idea is systematic change of neighborhood within a local search. VNS starts with an initial solution created by a construction method. And then an attempt is made to improve the initial solution by applying a certain neighborhood structure. Then a local search is performed to this neighborhood solution. If no more optimal solution can be updated, change another neighborhood structure to enlarge the search scope, keep iteration until a convergence criterion is reached. In this framework, local search is used in the same neighborhood structure and the new neighborhood is adopted on the basis of the current solution to escape from local optima in the subsequent search process.

3.1.1 Algorithm procedure

VNS framework contains four main sub procedures: initial solution construction, neighborhood solution production, local search and optimal solution updating. The algorithm keeps iteration based on these procedures until the convergence criterion satisfies. The algorithm flow is described in detail as follows:

Step 1. Construct the initial solution x, set the current optimum $x_b \leftarrow x$, select the set of neighborhood structures $N_k (k = 1, \cdots, k_{max})$.

Step 2. Set $\mathrm{k} \leftarrow 1$.

Step 3. If $k > k_{max}$, go to Step 2.

Step 4. If the convergence criterion is met, output the current optimum and exit.

Step 5. Generate solution $x^{'}$ from the *k*th neighborhood structure of x $(x^{'} \in N_k(x)$.

Step 6. Apply local search method with $x^{'}$ as initial solution; denote by $x^{''}$ the obtained local optimum.

Step 7. If this local optimum $x^{''}$ is better than the incumbent x_b, set $x_b \leftarrow x^{''}$, go to Step 4; otherwise set $k \leftarrow k+1$, change another neighborhood structure, go to Step 3.

3.1.2 Sub procedures design

(1) Initial solution construction

A modified insertion algorithm based on Solomon I1 algorithm is designed to construct a high quality initial solution. In Solomon I1, the best customer to be inserted and its best insert position is obtained by comparing cost function $c_2(i,u,j)$ of each customer. It is defined as follows:

$$\begin{cases} c_2(i,u,j) = \lambda d_{0u} - c_1(i,u,j), \lambda \geq 0 \\ c_1(i,u,j) = \alpha c_{11}(i,u,j) + \beta c_{12}(i,u,j), \alpha + \beta = 1 \\ c_{11}(i,u,j) = d_{iu} + d_{uj} - d_{ij} \\ c_{12}(i,u,j) = b_j^{'} - b_j \end{cases} \tag{1}$$

In this formulation, cost function $c_1(i,u,j)$ includes two parts: the added distance $c_{11}(i,u,j)$ and the time delay $c_{12}(i,u,j)$ for service to begin at customer *j* when inserting customer u between customer *i* and *j*, α and β are the corresponding coefficient. d_{ij} means the distance between customer *i* and *j*. b_j and $b_j^{'}$ denote the start time to serve customer *j* before and after inserting customer *u*, respectively.

$c_2(i,u,j)$ is the saving in distance from servicing customer *u* on the same route with customers *i* and *j*, as opposed to individual, direct service.

In Solomon I1, $c_{12}(i,u,j)$ is the time delay for service to begin at customer *j* after inserting *u*, while in this paper we make some improvement, considering the added waiting time of this route. When a vehicle arrives at a customer before its time window opens, it has to wait. So the waiting time of one route is the sum of waiting time of each customer. We redefine $c_{12}(i,u,j)$ as:

$$c_{12}(i,u,j) = w_u - w_j \tag{2}$$

Where w_u is the sum of waiting time of customer *u* and all its successors after inserting customer *u*. And w_j denotes as the waiting time of customer *j* and all its successors before insertion.

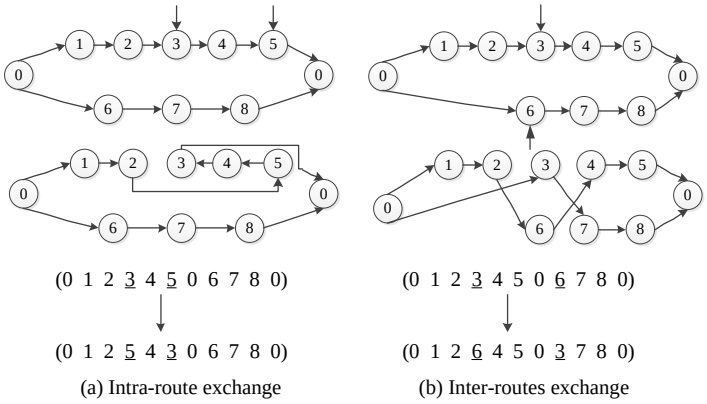
(2) Neighborhood Solution Generation

In this procedure, a new solution is generated using a certain neighborhood structure. Current optimal solution is obtained by comparing with the local optimal solution. So this procedure searches in the new solution space to avoid falling into local optima.

A neighborhood is generated by exchanging some nodes or edges, or relocating them to some other positions based on current solution. We choose three popular and effective route improvement operators, including *exchange*, *2-opt* and *crossexchange*. In our implementation,

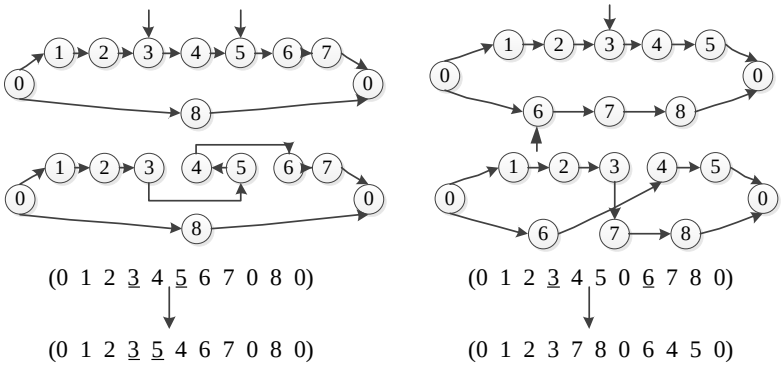
the operators are modified and renamed as *All-exchange*, *All-2-opt* and *All-crossExchange*, due to their nature of applicable for both inter- and intra-routes improvements.

The traditional *exchange* operator (Savelsbergh, 1992) swaps simultaneously two different customers in two different routes, while the ordinary *2-opt* (Flood, 1956) operator improves a single route by replacing two of its arcs by two other arcs. Although improvement made by intra-route operation may not be so significant than that made by inter-route operation because of the restriction of time windows, neglecting intra-route operation may lower the diversity of the neighborhood. Thus, modifications are conducted by extending the swapping and replacing operations to all routes. *CrossExchange* operator is the extension of *exchange* operator, aimed at swapping multi-nodes in different routes. Let 0 represent the depot, and the other numbers indicate the customers. Figures 1, 2 and 3 show examples of *All-exchange*, *All-2-opt* and *All-crossExchange* operators, respectively, with (a) illustrates the intra-route case and (b) illustrates the inter-routes case.



(a) Intra-route exchange (b) Inter-routes exchange

Figure 1 *All-exchange* opertor



(a) Intra-route 2-opt (b) Inter-routes 2-opt

Figure 2 *All-2-opt* operator

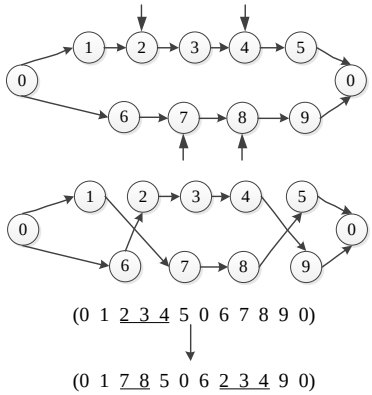


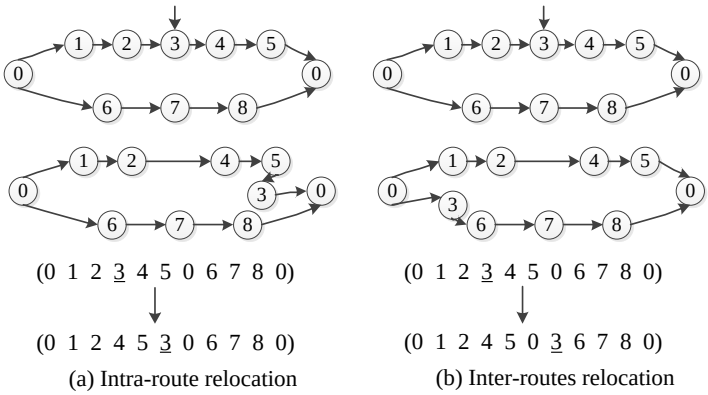
Figure 3 *All-crossExchange* operator

VNS chooses one of these neighborhood structures to generate neighborhood solutions according to some mechanism, and those feasible solutions are accepted. In order to lower the number of vehicles, the sum-of-squares route sizes is maximized in this procedure. This strategy can make the customers gathered to part of the routes, in other words, the number of customers in each route may tend to be saturated (as many as possible) or tend to be zero (so as to move the customers out of those short routes later and finally eliminate such routes).

(3) Local search

The neighborhood solution obtained is submitted to a local search procedure to come up with a local optimal solution. This procedure further improves the solution by trying to minimize the number of vehicles.

Since this is a loop process and time consuming, we choose the *relocate* operator and *ejection chain* operator to improve the solution until no improvement is possible. First proposed by Savelsbergh (1992), *Relocate* operator is to relocate a customer from one route to another. We also extend this operator to intra- and inter-routes scenarios and rename it as *All-relocate*. By this operator, a randomly selected customer is relocated to the best position, whatever it is in the same route or not. *Ejection chain* operator (Bräysy, 2003) tries to move customers among different routes to generate a feasible solution. Both operators can reduce the number of customers on some routes, which improves the possibility of eliminating such routes later. Figures 4 and 5 give examples of *All-relocate* operator and *Ejection Chain* operator.



(a) Intra-route relocation (b) Inter-routes relocation

Figure 4 *All-relocate* operator

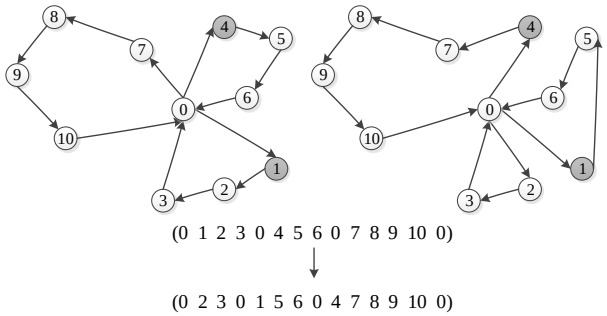


Figure 5 *Ejection Chain* operator

The basic idea of *relocate* operator is to search other feasible positions for each customer. If the new solution obtained is better than the original one, update the current optimal solution and continue to search on the basis of the new solution; otherwise the current optimal solution is not updated, search continues until all customers are searched. During this procedure, the shorter routes (with fewer customers) will be conducted first; also, relocating a customer to the shortest route (with minimum customers) is prohibited, since it has the probability to be eliminated.

The basic idea of *ejection chain* operator is to find the shortest route first, and then relocate the customers on this route to other routes to eliminate this route. Ejection chain operator is embedded into the relocation operator. Every time a new optimal solution is obtained, apply ejection chain operator to relocate the customers on the shortest route to the newly update routes.

(4) Current solution updating

If the local optimal solution obtained in the local search procedure is better than the current optimal solution, update the current optimal solution, and continue to use the current neighborhood structure; otherwise, change another neighborhood structure and keep searching until the convergence criterion is met. We limit the maximum number of iterations, when the current optimal solution hasn't update for certain iterations, exit.

The comparison between these solutions follows three criteria: the number of vehicles, the length of the shortest route, and the total distance. For each criterion, the smaller the value, the better the solution is. The priority of these three criteria reduces in order.

3.2 Tabu Search

While the first phase—VNS aims at minimizing the number of vehicles, whereas the second phase—TS tries to minimize the total travel distance. The solution obtained in the first phase will act as the initial solution of the second phase.

3.2.1 Algorithm procedure

The basic principle of tabu search algorithm is as follows: generate an initial solution x by a construction algorithm or heuristic algorithm or metaheuristic algorithm. Set the current solution and optimal solution x^{best} by x. Define the set of neighborhood structures $N_k(k=1,\cdots,k_{max})$. In each iteration, compute the neighborhood solution set. Choose some better neighborhood solutions to form the candidate solution set $\{x^{candidate} \mid x^{candidate} \in N_k(\mathrm{x})\}$. Find the best candidate solution $x^{candidate}$ no matter whether it is tabu or not, if it is better, update the optimal solution x^{best} and current solution x by $x^{candidate}$, and put it in the tabu list to avoid being searched again; otherwise for every element in the candidate solution set,

choose the best one to be the current solution x and update the tabu list. In the second case, the optimal solution x^{best} is not updated. Through different neighborhood structures, the diversity of candidate solution is enlarged. The algorithm flow is list as follows:

Step 1. Given the parameters and initial feasible solution x, set the current solution and optimal solution $x^{best} \leftarrow x$. Clear the tabu list.

Step 2. If the convergence criterion is met, output the optimal solution x^{best} and exit.

Step 3. Generate the neighborhood solution set $N_k(\mathrm{x})$. Choose some better solutions to form the candidate solution set $\{x^{candidate} \mid x^{candidate} \in N_k(\mathrm{x})\}$.

Step 4. If the best candidate solution $x^{candidate}$ is better than optimal solution x^{best}, set $x^{best} \leftarrow x^{candidate}$, put $x^{candidate}$ into the tabu list, go to Step2; otherwise go to Step5.

Step 5. For every solution in the candidate solution set, choose the best non-tabu one x', set $x \leftarrow x'$, go to Step2.

3.2.2 Key element design

The design of tabu search algorithm relates to a few details, including: initialization, neighborhood structure, tabu list, evaluation of solutions, aspiration criterion, and convergence criterion.

(a) Initial Solution

In this paper, the final solution obtained in the first phase will be set as the initial solution of TS algorithm.

(b) Neighborhood Structure

We use *All-exchange*, *All-2-opt*, *All-relocate* and *All-crossexchange* operator to generate neighborhoods, these operators are the same as those in section 3.1.2.

(c) Tabu List

A tabu list is a short-term set of the solutions that have been visited in the recent past (less than T iterations ago, where T is the number of previous solutions to be stored -- also called the tabu length). This is to memory the solution that has been visited and avoids trapping in local optimum. In this paper, the tabu list simply stored the objective function value.

Tabu length T is a finite value, as the current solution or optimal solution continuously added to the tabu list, the previously tabu solution is popped out. So the search process is able to reach this solution again, since this design is to avoid the loss of the beneficial search direction. T has an influence on the release time of the solution in the tabu list. If T is too big, more solutions will be tabu, so the search space reduces and the quality of the optimal solution is affected. If T is too small, some better solutions lose the opportunity to be released, leading to search around some fixed solutions and so the global optimum is limited. In this paper, different values of T was tested and the best one was adopted.

(d) Evaluation of Solutions

The VRPTW has two main objectives to be optimized: the number of vehicles and total travelling cost. As mentioned above, the priority is given to the first one. Therefore, a feasible solution with a certain number of vehicles dominates over any other feasible solutions requiring more vehicles. For those solutions with the same number of vehicles required, the one with minimum total travelling cost is selected.

(e) Aspiration Criterion

To avoid cycling, if all solutions in the candidate solution set are tabu, choose the best one as the new current solution.

(f) Convergence Criterion

The stopping condition may be e.g. maximum CPU time allowed, maximum number of iterations, or maximum number of iterations between two improvements. Here the search is terminated if either a specified number of iterations have elapsed in total or since the last best solution was found. If the best solution so far hasn't been improved for a specified number of consecutive iterations, the search also stops.

We also implemented the "shaking" step as in VNS. In our test, we found that TS usually could find a good solution in less than 50 iterations. And for the remaining computation process, it is very hard to further improve the quality. So each time the solution cannot be updated for a certain consecutive iterations, e.g. 80, we take the "shaking" step —embed a disturbance factor, and then the search goes on. It is proved that this method is very effective — the procedure can easily escape from the local minima trap, thus both the number of routes and the total travelling cost have the opportunity to be further reduced.

3.3 Data Structure Design

To speed up the algorithm and reduce memory usage, we adopt the one-way linked list data structure in an array (called "next array") to represent solutions (Kytöjoki, 2007). This structure combines the advantages of two data structures: linked list and array. The first customer of each route is recorded in a separate auxiliary array ("start array"). A customer is stored in "next array" with the index that is equal to its immediate predecessor. These two arrays are enough to completely represent solutions.

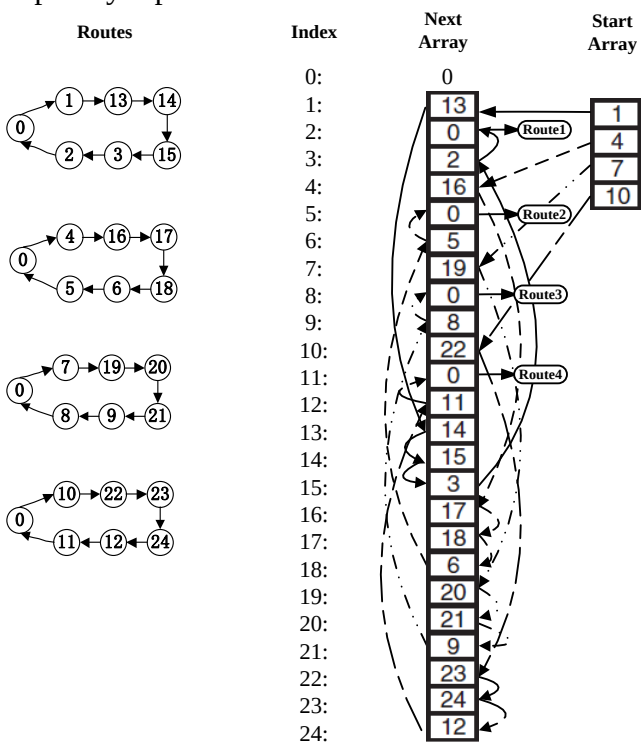


Figure 6 Example of the one-way linked list data structure

As shown in figure 6, there are four short routes: 0-1-13-14-15-3-2-0, 0-4-16-17-18-6-5-0, 0-7-19-20-21-9-8-0 and 0-10-22-23-24-12-11-0. "Start Array" records the first customer of each route: 1, 4, 7, and 10; while "Next Array" stores other customers according to its immediate predecessor. For example, considering route 0-1-13-14-15-3-2-0, store customer 13 in the position with index 1, store customer 14 in the position with index 13

and so on.

Using this structure, changes to the solution can be performed very quickly and in a constant time, without having to loop as in standard array implementations. For example, the addition of a new customer 25 between two adjacent customers 1 and 13 is done simply by changing the "next-values" of 25 to 13 and 1 to 25. Similarly, delete and reverse operations are also high efficient. When deleting customer 13 from this route, one can simply change the "next-values" of 1 from 13 to 14.

4. EXPERIMENTAL RESULTS

In this section, we present and analyze the results obtained with the above two-phase hybrid metaheuristic based on the Solomon's (1987) 100-customer benchmark problems.

4.1 Setting of Parameters

During the first algorithm phase, we tested 8 strategies to select seed customers, according to the customers' location and time windows, they are: farthest customer (to the depot), nearest customer, and customer with earliest/latest service start time, earliest/latest service due time, maximum/minimum time window length. Also we need to determine parameters α,β,γ which are use in the Solomon I1 algorithm. We set the following bounds for the most crucial parameters: α: 0.7-1.0 (in increments of 0.1 units), β: 0-0.3 (in increments of 0.1 units), γ: 0.5-1.7 (in increments of 0.2 units). The value of α and β should satisfy the equation: $\alpha+\beta=1$. The parameter values used depend on the characteristic of the problem. It is advisable to use a different parameter set in different data sets to get better results. The best combinatorial parameters and the best selection scheme are determined by multiple experiments. Another parameter in VNS is the iteration times when the optimal solution hasn't been improved, denoted by *maxiter1*.

In the tabu search phase, parameters to be assigned are the number of candidate solutions *m*, tabu length *T*, iterations *maxiter2*. The larger scale the instance is, the more candidate solutions need. So the number of candidate solutions may be different for different scales of instance.

The setting of parameters is shown in Table 1.

Table 1 Setting of parameters for HM algorithm

Parameters	α	β	λ	*n*	*m*	*T*	*maxiter1*	*maxiter2*
Values	0.7~1.0	0~0.3	0.5~1.7	100	2000	10	5	2000

In this table, *n* is the scale of an instance. With the best parameters setting, the results obtained with two-phase hybrid metaheuristic will be contrasted with results published in the literature or in the Internet. All our calculations were carried out on a PC (AMD processor, 2.3GHz, 1G RAM). The algorithm was coded in C++ and compiled with Visual Studio 2008.

4.2 Comparative Analysis

To evaluate the proposed two-phase hybrid metaheuristic and to tune its parameter, we test it multi-times on 56 benchmark problems proposed by Solomon (1987). These problems have

100 customers, a central depot, capacity constraints, time windows on the time of delivery, and a total route time constraint. Customer locations are either randomly generated (problem sets R1 and R2), clustered (problem sets C1 and C2) or mixed with randomly generated and clustered customers (problem sets RC1 and RC2).

Each of these instances was calculated 5 times, and the best solutions that were achieved for the instances of the respective problem class are listed in Table 2. In this table, averages over each class of problems (C1, C2, R1, R2, RC1, and RC2) are given. We refer to the simplified terms introduced in Homberger *et al.* (1999). *MNV* denotes the mean number of vehicles, *MND* the mean travel distance and *MCT* the mean number of CPU seconds used for finding the solutions. The first row to the top lists the authors. In the bottom line, the cumulated number of vehicles *CNV* and the cumulated total travel distance *CND* are shown.

Table 2 Results for Solomon Instances

Instances	Values	RT(1995)	PB(1996)	LS(1998)	Nazif(2010)	Ursani(2011)	**HM(2013)**
C1	MNV	10	10	10	10	10	**10**
	MND	828.45	838	830.06	828.430	828.38	**828.378**
	MCT	3200	601	1320.6	/	351.863	**15.906**
C2	MNV	3	3	3	3	3	**3**
	MND	590.32	589.9	591.03	589.860	589.86	**589.858**
	MCT	7200	2482	215	/	2261.53	**34.571**
R1	MNV	12.58	12.6	12.17	13.42	13.5	**12.25**
	MND	1197.42	1296.83	1249.57	1209.560	1188.85	**1202.7**
	MCT	2700	679	2657	/	244.57	**63.577**
R2	MNV	3.09	3	2.82	5.36	5.55	**2.72727**
	MND	954.36	1117.7	1016.58	939.110	885.78	**969.29**
	MCT	9800	2384	398		1482.49	**108.023**
RC1	MNV	12.38	12.1	11.88	13.12	13.25	**11.75**
	MND	1369.48	1446.2	1412.87	1375.57	1360.96	**1387.37**
	MCT	2600	673	1828	/	256.29	**53.477**
RC2	MNV	3.62	3.4	3.25	6	6.88	**3.25**
	MND	1139.79	1360.6	1204.87	1096.71	1013.29	**1148.84**
	MCT	7800	2134	427	/	1202.602	**136.768**
All	CNV	427	422	412	487	498	**411**
	CND	57120	62572	59317	56798	55137.04	**57558.5**

Notes: RT—Rochat and Taillard (1995), PB—Potvin and Bengio (1996), LS—Liu and Shen (1998), Nazif — Nazif, H. *et, al.* (2010), Ursani — Ursani, Z. *et, al.* (2011)

Due to limited space, best achieved solution for each of the 56 problems instances are listed in the table in appendices. From the results of the comparative test and detail solutions in these two tables, we can see that HM algorithm can get a solution as good as current published optimum.

For problem groups C1 and C2, two-phase hybrid metaheuristic has got all the best solutions. The mean travel distance of class C1 and C2 is better than ever published literatures. For class R1, three best solutions have been found. For class R2, the best known mean *MNV* and *MND* are both renewed. For class RC1 and RC2, two-phase hybrid metaheuristic leads

new best means *MNV*. In the former, two best known solutions have been found.

In all, the cumulated number of vehicles *CNV* obtained by two-phase hybrid metaheuristic is the lowest, in comparison with the other methods; meanwhile, the total travel distance *CND* is also very competitive. Since the number of routes is set as the primary objective, two-phase hybrid metaheuristic performs better than other heuristic.

HM also outperforms other algorithms on the computing time, regardless the effect of different hardware. Although previous literatures were computed on different computer platforms, the difference of computing time between these heuristics and HM are really obvious—HM only occupies 1/5 to 1/30 of the mean CPU time. This for one hand may due to the one-way linked list data structure, for the other hand to our specially designed operators and evaluation criteria which may decrease the objective more quickly.

To study the consistency and reliability of the results obtained by two-phase hybrid metaheuristic, we also analyzed the variance as introduced in Tan *et al.* (2006). 8 repeated simulations with different seed customers' selection strategies have been performed for the Solomon's data sets. We only take the first instance in every problem class (C101, C201, R101, R201, RC101 and RC201) as a representative. The simulation results are presented in Table 3.

In this table, the first row of each class describes the number of vehicles used, while the second represents the total travelling cost under this strategy. The last two columns list the standard deviations (denotes as *S*) and coefficient of variation (denoted as *CV*) for the various simulation results.

Table 3 Reliability performance for the algorithm

	1	2	3	4	5	6	7	8	*S*	*CV* (10^{-3})
C101	10 828.937	10 828.937	10 828.937	10 828.937	10 828.937	10 828.937	10 828.937	10 828.937	0	0
C201	3 591.557	3 591.557	3 591.557	3 591.557	3 591.557	3 591.557	3 591.557	3 591.557	0	0
R101	19 1650.8	19 1653.53	19 1652.17	19 1652.47	19 1652.17	19 1653.84	19 1654.22	19 1652.63	1.10	0.668
R201	4 1261.87	4 1286.16	4 1266.43	4 1300.03	4 1257.41	4 1278.29	4 1283.7	4 1257.41	15.55	12.207
RC101	15 1664.13	16 1678.69	16 1680.02	15 1699.07	15 1724.37	15 1699.64	16 1690.48	15 1679.81	18.34	10.858
RC201	4 1455.06	4 1462.69	4 1462.89	4 1466.92	4 1487.55	4 1462.86	4 1429.41	4 1455.61	16.08	11.008

From this table, for class C101 and C201, all the simulations have obtained the same best optimal solutions. For category of type R, customer locations are randomly distributed; small differences exist between these tests. But the coefficient of variation, which reflects the ratio between the standard deviation and the mean value, is so small that the strong robustness of the new two-phase hybrid metaheuristic is obvious. For category of type R and RC, an approximate solution which is very close to the best published solution is obtained among these 8 experiments. We also observe that results for category of type 1 (C101, R101, and RC101) have smaller standard deviation values as compared to those for other test cases.

We also notice that different calculation result exist between different instances, two main reasons is list as follows:

(1) The calculation results have certain randomness. Randomness is embodied in neighborhood operator selection and candidate solution generation. This paper chooses the best result among several tests. Some new optimal solution may be generated if repeated more times.

(2) The computing time limitation. Eight different parameters are used in two-phase hybrid metaheuristic (see Table 1). In our test, we find that they contribute a lot to the quality of final solution. In the normal case, the search process can easily trap in local optimum. But testing all the parameter combination is time-consuming and nearly impossible. We only random choose some combinations, which may have an influence on the quality of the final optima. If more test performed, some better results may be obtained.

5. CONCLUSIONS

This paper introduces a new two-phase hybrid metaheuristic based on variable neighborhood search and tabu search to solve the classical VRP. Five neighborhood structures (*All-exchange*,

All-2-opt, *All-crossexchange*, *All-relocate* and ejection chain) are designed. We modify Solomon's I1 heuristic by redesign the cost function so as to consider the waiting time of each customer. More seed selection strategies are tested to get better initial solution. To further lower the number of vehicles, the sum-of-squares route sizes is maximized in the first phase. Computational testing on 56 benchmark problems has shown that the overall solution quality of this algorithm is competitive with existing metaheuristics

For further research, a few improvements can be attempted to obtain better results and lower running time. More combination of parameters setting can be explored. Meanwhile, currently the evaluation of solutions of the second phase is based on the number of vehicles or the total travelling cost, whereas the number of customers on the route may also be taken into consideration. The data structure also has room to be improved so as to reduce the running time.

ACKNOWLEDGEMENTS

This work is supported by the National Natural Science Foundation of China (Number 70702003, 71272030), Science & Technology Foundation of Dongguan City (Number 201010810107), and fund from Guangdong provincial department of science and technology (Number 2011B090400384).

APPENDICES

The best published results and results by HM for Solomon's data sets (1987) are listed in table 4. For detailed best-known results to Solomon's benchmarks, we refer the reader to the web site http://neo.lcc.uma.es/radi-aeb/WebVRP/.

Table 4 Best results for Solomon (1987)

Instance	Best Published Solution			Best Solution (HM)		Comparison	
	MNV	MND	Reference	MNV	MND	Veh (%)	Cost (%)
C101	10	828.94	Rochat and Taillard (1995)	10	828.937	0.00%	0.00%
C102	10	828.94	Rochat and Taillard (1995)	10	828.937	0.00%	0.00%
C103	10	828.06	Rochat and Taillard (1995)	10	828.065	0.00%	0.00%
C104	10	824.78	Rochat and Taillard (1995)	10	824.777	0.00%	0.00%
C105	10	828.94	Rochat and Taillard (1995)	10	828.937	0.00%	0.00%
C106	10	828.94	Rochat and Taillard (1995)	10	828.937	0.00%	0.00%
C107	10	828.94	Rochat and Taillard (1995)	10	828.937	0.00%	0.00%
C108	10	828.94	Rochat and Taillard (1995)	10	828.937	0.00%	0.00%
C109	10	828.94	Rochat and Taillard (1995)	10	828.937	0.00%	0.00%
C201	3	591.56	Rochat and Taillard (1995)	3	591.557	0.00%	0.00%
C202	3	591.56	Rochat and Taillard (1995)	3	591.557	0.00%	0.00%
C203	3	591.17	Rochat and Taillard (1995)	3	591.173	0.00%	0.00%
C204	3	590.6	Rochat and Taillard (1995)	3	590.599	0.00%	0.00%
C205	3	588.88	Rochat and Taillard (1995)	3	588.876	0.00%	0.00%
C206	3	588.49	Rochat and Taillard (1995)	3	588.493	0.00%	0.00%
C207	3	588.29	Rochat and Taillard (1995)	3	588.286	0.00%	0.00%
C208	3	588.32	Rochat and Taillard (1995)	3	588.324	0.00%	0.00%

R101	19	1645.79	Homberger (2000)	19	1650.8	0.00%	-0.30%
R102	17	1486.12	Rochat and Taillard (1995)	17	1486.12	0.00%	0.00%
R103	13	1292.68	Li, Lim and Huang (2001)	13	1294.64	0.00%	-0.15%
R104	9	1007.24	Mester *et al.* (2005)	9	1013.26	0.00%	-0.60%
R105	14	1377.11	Rochat and Taillard (1995)	14	1377.11	0.00%	0.00%
R106	12	1251.98	Mester *et al.* (2005)	12	1263.31	0.00%	-0.90%
R107	10	1104.66	Shaw (1997)	11	1073.34	-10.00%	2.84%
R108	9	960.88	Berger *et al.* (2001)	9	960.876	0.00%	0.00%
R109	11	1194.73	Homberger and Gehring (1999)	12	1155.46	-9.09%	3.29%
R110	10	1118.59	Mester *et al.* (2005)	11	1093.01	-10.00%	2.29%
R111	10	1096.72	Rousseau *et al.* (2002)	11	1060.65	-10.00%	3.29%
R112	9	982.14	Gambardella *et al.* (1999)	9	1003.77	0.00%	-2.20%
R201	4	1252.37	Homberger and Gehring (1999)	4	1257.41	0.00%	-0.40%
R202	3	1191.7	Rousseau *et al.* (2002)	3	1195.3	0.00%	-0.30%
R203	3	939.5	Woch and Lebkowski (2009)	3	949.764	0.00%	-1.09%
R204	2	825.52	Bent and Van Hentenryck (2001)	2	843.48	0.00%	-2.18%
R205	3	994.42	Rousseau *et al.* (2002)	3	1004.8	0.00%	-1.04%
R206	3	906.14	Schrimpf *et al.* (2000)	3	929.604	0.00%	-2.59%
R207	2	890.61	Ropke and Pisinger (2007)	2	890.608	0.00%	0.00%
R208	2	726.75	Mester *et al.* (2005)	2	747.395	0.00%	-2.84%
R209	3	909.16	Homberger (2000)	3	933.655	0.00%	-2.69%
R210	3	939.34	Mester *et al.* (2005)	3	952.754	0.00%	-1.43%
R211	2	885.71	Woch and Lebkowski (2009)	2	957.42	0.00%	-8.10%
RC101	14	1696.94	Taillard *et al.* (1997)	15	1642.09	-7.14%	3.23%
RC102	12	1554.75	Taillard *et al.* (1997)	13	1493.85	-8.33%	3.92%
RC103	11	1261.67	Shaw (1998)	11	1264.64	0.00%	-0.24%
RC104	10	1135.48	Cordeau *et al.* (2000)	10	1173.77	0.00%	-3.37%
RC105	13	1629.44	Berger *et al.* (2001)	13	1629.44	0.00%	0.00%
RC106	11	1424.73	Berger *et al.* (2001)	11	1424.73	0.00%	0.00%
RC107	11	1230.48	Shaw (1997)	11	1290.16	0.00%	-4.85%
RC108	10	1139.82	Taillard *et al.* (1997)	10	1180.31	0.00%	-3.55%
RC201	4	1406.91	Mester *et al.* (2005)	4	1429.41	0.00%	-1.60%
RC202	3	1,365.645	Grabysz *et al.* (2004)	3	1389.57	0.00%	-1.75%
RC203	3	1049.62	Czech and Czarnas (2002)	3	1098.17	0.00%	-4.63%
RC204	3	798.41	Mester *et al.* (2005)	3	810.179	0.00%	-1.47%
RC205	4	1297.19	Mester *et al.* (2005)	4	1309	0.00%	-0.91%
RC206	3	1146.32	Homberger (2000)	3	1218.83	0.00%	-6.33%
RC207	3	1061.14	Bent and Van Hentenryck (2001)	3	1084.61	0.00%	-2.21%
RC208	3	828.14	Ibaraki *et al.* (2001)	3	850.915	0.00%	-2.75%

REFERENCES

Badeau, P., Guertin, F., Gendreau, M., Potvin, J., Taillard, E. (1997). A parallel tabu search

heuristic for the vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies, 5*(2), 109-122.

Baldacci, R., Christofides, N., Mingozzi, A. (2008). An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming, 115*(2), 351-385.

Bent, R., Hentenryck, P. V. (2006). A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. *Computers & Operations Research, 33*(4), 875-893.

Blocho, M., Czech, Z. J. (2012). A parallel EAX-based algorithm for minimizing the number of routes in the vehicle routing problem with time windows. *High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS)*

Bräysy, O. (2003). A reactive variable neighborhood search for the vehicle-routing problem with time windows. *INFORMS Journal on Computing, 15*(4), 347-368.

Bräysy, O., Gendreau, M. (2002). Tabu search heuristics for the vehicle routing problem with time windows. *Top, 10*(2), 211-237.

Bräysy, O., Gendreau, M. (2005a). Vehicle routing problem with time windows, part II: Metaheuristics. *Transportation science, 39*(1), 119-139.

Bräysy, O., Gendreau, M. (2005b). Vehicle routing problem with time windows, Part I: Route construction and local search algorithms. *Transportation science, 39*(1), 104-118.

Chen, H., Hsueh, C., Chang, M. (2006). The real-time time-dependent vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review, 42*(5), 383-408.

Chiang, W., Russell, R. A. (1997). A reactive tabu search metaheuristic for the vehicle routing problem with time windows. *INFORMS Journal on computing, 9*(4), 417-430.

Cordeau, J., Laporte, G., Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational research society*, 928-936.

Dantzig G B, Ramser J H. (1959). The truck dispatching problem. *Management science*, 6(1),80-91.

Flood, M. M. (1956). The traveling-salesman problem. *Operations Research, 4*(1), 61-75.

Fu, Z., Eglese, R., Li, L. Y. (2004). A new tabu search heuristic for the open vehicle routing problem. *Journal of the operational Research Society, 56*(3), 267-274.

Gehring, H., Homberger, J. (1999). A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. *Proceedings of EUROGEN99*, Finland.

Gehring, H., Homberger, J. (2002). Parallelization of a two-phase metaheuristic for routing problems with time windows. *Journal of heuristics, 8*(3), 251-276.

Glover, F. (1989). Tabu search—part I. *ORSA Journal on computing, 1*(3), 190-206.

Goel, A., Gruhn, V. (2008). A general vehicle routing problem. *European Journal of Operational Research, 191*(3), 650-660.

Hansen, P., Mladenović, N. (1997). Variable neighborhood search for the p-median. *Location Science, 5*(4), 207-226.

Hemmelmayr, V. C., Doerner, K. F., Hartl, R. F. (2009). A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research, 195*(3), 791-802.

Homberger, J., Gehring, H. (2005). A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational Research, 162*(1), 220-238.

K. C. TAN, Y. H. CHEW. (2006). A Hybrid Multiobjective Evolutionary Algorithm for Solving Vehicle Routing Problem with Time Windows. *Computational Optimization*

and Applications, 34, 115-151.

Kytöjoki, J., Nuortio, T., Bräysy, O., Gendreau, M. (2007). An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers & Operations Research, 34*(9), 2743-2757.

Laporte, G. (2007). What you should know about the vehicle routing problem. *Naval Research Logistics (NRL), 54*(8), 811-819.

Mester, D., Bräysy, O. (2005). Active guided evolution strategies for large-scale vehicle routing problems with time windows. *Computers & Operations Research, 32*(6), 1593-1614.

Mladenović, N., Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research, 24*(11), 1097-1100.

Nazif, H., Lee, L. S. (2010). Optimized Crossover Genetic Algorithm for Vehicle Routing Problem with Time Windows. *American Journal of Applied Sciences*, 7(1), 95-101.

Pisinger, D., Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research, 34*(8), 2403-2435.

Polacek, M., Hartl, R. F., Doerner, K., Reimann, M. (2004). A variable neighborhood search for the multi depot vehicle routing problem with time windows. *Journal of Heuristics, 10*(6), 613-627.

Potvin, J., Bengio, S. (1996). The vehicle routing problem with time windows part II: genetic search. *INFORMS journal on Computing, 8*(2), 165-172.

Prescott Gagnon, E., Desaulniers, G., Rousseau, L. M. (2009). A branch‐and‐price‐based large neighborhood search algorithm for the vehicle routing problem with time windows. *Networks, 54*(4), 190-204.

Qi, M. Y., Li, N., Zhang, J. J., Miao, L. X. (2010). Variable Neighborhood Search Heuristic for Large Scale Real-time Time-dependent Vehicle Routing Problem with Time Windows, *3rd T- Log International Conference*. Japan.

Qi, M. Y., Zhang, J. J., Li, N. (2010). A Variable Neighborhood Search Heuristic for Large Scale Real-time Time-dependent Vehicle Routing Problem with Time Windows, *China logistics academic conference*. Nan Jing.

Renaud, J., Laporte, G., Boctor, F. F. (1996). A tabu search heuristic for the multi-depot vehicle routing problem. *Computers & Operations Research, 23*(3), 229-235.

Rochat, Y., Taillard, É. D. (1995). Probabilistic diversification and intensification in local search for vehicle routing. *Journal of heuristics, 1*(1), 147-167.

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

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research, 35*(2), 254-265.

Ursani, Z., Essam, D., Cornforth, D., Stocker, R. (2011). Localized genetic algorithm for vehicle routing problem with time windows. *Applied Soft Computing*, 11(8), 5375-5390.