Expert Systems with Applications 37 (2010) 1620–1627

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Iterated variable neighborhood descent algorithm for the capacitated vehicle routing problem

Ping Chen [a,b,*], Hou-kuan Huang [b], Xing-Ye Dong [b]

[a] TEDA College, Nankai University, 30071 Tianjin, China
[b] School of Computer and Information Technology, Beijing Jiaotong University, 100044 Beijing, China

ARTICLE INFO

Keywords:
Capacitated vehicle routing problem
Iterated local search
Variable neighborhood descent
Perturbation

ABSTRACT

The capacitated vehicle routing problem (CVRP) aims to determine the minimum total cost routes for a fleet of homogeneous vehicles to serve a set of customers. A wide spectrum of applications outlines the relevance of this problem. In this paper, a hybrid heuristic method IVND with variable neighborhood descent based on multi-operator optimization is proposed for solving the CVRP. A perturbation strategy has been designed by cross-exchange operator to help optimization escape from local minima. The performance of our algorithm has been tested on 34 CVRP benchmark problems and it shows that the proposed IVND performs well and is quite competitive with other state-of-the-art heuristics. Additionally, the proposed IVND is flexible and problem dependent, as well as easy to implement.

1. Introduction

Distribution of goods is of paramount importance in logistics and supply chain management. The vehicle routing problem (VRP) plays an important role in reducing the transportation cost in logistics distribution. The standard version of the VRP, called as capacitated vehicle routing problem (CVRP), consists of determining several vehicle routes with minimum cost for serving a set of customers, whose geographical coordinates and demands are known in advance. Each customer is required to be visited only once by one vehicle. Typically, vehicles are homogeneous and have the same capacity restriction. In some cases, a duration upper bound is assigned which limits the maximum travel time for each vehicle route.

For clarity, we define the CVRP on a connected graph G. Let $G = (V, A)$, where $V = \{v_0, v_1, \ldots, v_{n+1}\}$ is a vertex set and $A = \{(v_i, v_j) | v_i, v_j \in V, v_i \neq v_j\}$ is an arc set. Vertices v_0 and v_{n+1} correspond to the depot at which K homogeneous vehicles are based, and the remaining vertices denote the customers. Each arc (v_i, v_j) is associated with a non-negative weight $c_{v_i v_j}$, which represents the travel distance from v_i to v_j. The CVRP considered in this paper is supposed to be symmetric, i.e., $c_{v_i v_j} = c_{v_j v_i} (i, j = 0, 1, \ldots, n+1)$. Each customer has a delivery demand q_i. The CVRP consists of determining a set of least cost vehicle routes such that:

(1) Each route starts and ends at the depot;
(2) Each customer is visited exactly once by exactly one vehicle;
(3) The total demand of the customers assigned to any vehicle must not exceed the vehicle capacity Q;
(4) For the CVRP with duration restriction, the duration of each route must not exceed an upper bound D. In such a case, each customer $v_i (i > 0)$ requires a specified service time st_i. The duration equals to the total travel time and the total service time for servicing all the customers on this route. Here, the travel time from v_i to v_j equals to the Euclidean distance $c_{v_i v_j}$ between them.

The CVRP can be seen as an extension of the well-known NP-hard traveling salesman problem (TSP) (Garey & Johnson, 1979), so the CVRP is also NP-hard. Due to the its practicability in real life and computational complexity in theory, the CVRP has received much attention since it was firstly mentioned by Dantzig and Ramser (1959). In the last several decades, researchers mainly focused on heuristics and metaheuristics which can find quite good solutions within rational time, especially for real life applications. Recently, it has become evident that the use of a sole metaheuristic is rather restrictive. Therefore, a new class of methods called hybrid metaheuristics has attracted more and more concern, which are more efficient and flexible. Actually, many recently published algorithms for solving VRP are hybrid metaheuristics. Reimann, Doerner, and Hartl (2004) proposed a D-ants algorithm for the CVRP which combined ant colony and local search. Prins (2004) presented an effective evolutionary algorithm with local search for the CVRP. Lin, Lee, Ying, and Lee (2009) have just proposed a hybrid metaheuristic algorithm for the CVRP recently, which takes advantages of both simulated annealing and tabu search.

The aim of this paper is to propose an iterated variable neighborhood descent heuristic IVND for the CVRP. Since accuracy,

* Corresponding author. Address: TEDA College, Nankai University, 30071 Tianjin, China. Tel.: +86 010 51683602.
E-mail address: chenpingbjtu@gmail.com (P. Chen).

0957-4174/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2009.06.047

speed, simplicity, and flexibility are the four critical criteria to access heuristics for solving VRPs according to Cordeau, Gendreau, Laporte, Potvin, and Semet (2002). The proposed method is a multi-operators based iterated variable neighborhood descent (VND) (Hansen & Mladenović, 2003) heuristic, which combines with a perturbation strategy. The strengths of the proposed heuristic lie on two aspects. Firstly, it is simple since both the VND procedure and the perturbation are based on several operators that are commonly used in methods for VRP. Secondly, it is flexible since it is problem-independent and can be adapted for other variants of VRPs. Computational study on 14 classical benchmark problems as well as 20 large scale problems from literature were taken to demonstrate the efficiency of the suggested method.

The remainder of this paper is organized as follows. Section 2 describes the problem and reviews the related work. Section 3 outlines the proposed solution methodology and elaborates each of its components. Computational results together with some discussions are reported in Section 4. Finally, Section 5 concludes this paper.

2. Literature review

Abundant literature have discussed methodologies for the CVRP since it was proposed. As the CVRP is NP-hard, many efforts have focused on the heuristics in recent years, which can be roughly divided into three categories: constructive heuristics, improvement heuristics and metaheuristics. As heuristics can find quite good solutions in acceptable time, they are preferred for solving the NP-hard problems with large size, especially for the real life cases.

Constructive heuristics build a feasible solution by adding one component to the current partial solution until a complete solution is got. The most well-known are the *savings algorithm* (Clarke & Wright, 1964) and the *insertion heuristics* (Mole & Jameson, 1976, Christofides, Mingozzi, & Toth, 1979). Such methods can build a feasible solution quite quickly, while the solution quality can not be guarantee, so they are usually used to generate initial solutions for other heuristics.

Improvement heuristics improve the current solution by iteratively exploring its neighbors. They can obtain either intra-route or inter-route improvement. For intra-route improvement, any heuristic method for the TSP such as *2-opt*, *3-opt* and *Or-opt* can be utilized, as each vehicle route in a CVRP can be viewed as a TSP; for inter-route improvement, some multi-route structures are needed, which can be classified as string *cross*, string *exchange*, string *relocation* and string *mix*. More details are referred to Laporte and Semet (2002). The flaw of improvement methods is that they are dependent on initial solutions and easily trapped into local optima.

Metaheuristics are some general solution strategies rather than heuristic rules designed for a specific type of problems. Many metaheuristics have already successfully applied to the VRP, such as simulated annealing (SA), tabu search (TS), and ant colony optimization (ACO) (Gendreau, Laporte, & Potvin, 2002). In recent years, some new metaheuristics have been proposed for the CVRP. Cordeau, Laporte, and Mercier (2001) extended the unified tabu search algorithm (UTSA) to the site dependent VRP. Tarantilis and Kiranoudis (2002) proposed an effective Bone Route adaptive memory procedure for the VRP. It firstly constructs a solution by means of an enhancement of the Clarke and Wright saving algorithm, and a tabu search procedure is followed for solution improvement. The adaptive memory procedure initiates new solutions by combining route segments, called *bones*, instead of full routes, which are extracted from good quality routes. Toth and Vigo (2003) proposed the concept of granular neighborhood and put forward a granular tabu search algorithm. Prins (2004) put forward a hybrid evolutionary algorithm combined with local search for improvement. Berger and Barkaoui (2004) also built a hybrid genetic algorithm. Reimann et al. (2004) proposed a hybrid ant colony algorithm D-ants savings based algorithm. Li, Golden, and Wasil (2005) presented the VRTR algorithm which combined the record-to-record (RTR) principle with a variable-length neighbor list. Ergun, Orlin, and Steele-Feldman (2006) applied the vary large neighborhood search (VLNS) to the VRP. Mester and Bräysy (2005) applied the active guided evolutionary strategies (AGES) algorithm to the CVRP and obtained quite good results. Readers are referred to the survey by Cordeau, Gendreau, Hertz, Laporte, and Sormany (2005) for more details.

3. Solution methodology

The proposed algorithm IVND combines the strengths of iterated local search (Lourenço, Martin, & Stützle, 2003) and variable neighborhood descent. Firstly, an initial solution s is built by a constructive heuristic method. Then the solution s is attempted to be improved by a VND procedure. Once a local optimum solution s^* is found, stop the VND procedure and perform the perturbation on s^*, thus a new solution s' is generated. From the solution s', a new VND procedure starts and finally gets a new local optimum s'^*. According to the solution quality and search history, it is determined by the acceptance criterion that which solution is used to be perturbed for the next loop. Above steps are repeated until the terminational conditions are met. The framework of the proposed IVND is illustrated by Algorithm 1, while each of its components will be described in detail in the following sections.

Algorithm 1. The Framework of the proposed algorithm IVND

1: Initialize parameters
2: $s_0 \Leftarrow$ *GenerateInitialSolution()*
3: $s_0^* \Leftarrow VND(s_0)$, $s^* \Leftarrow s_0^*$
4: Let the best solution found $s^{**} \Leftarrow s^*$
5: **repeat**
6: $s' \Leftarrow Perturbation(s^*)$
7: $s'^* \Leftarrow VND(s')$
8: **if** $f(s'^*) < f(s^{**})$ **then**
9: $s^{**} \Leftarrow s'^*$
10: **end if**
11: $s^* \Leftarrow AcceptanceCriterions^*, s'^*$, history
12: **until** *terminational rule is met*
13: **return** s^{**}

3.1. Initial solution generation

The proposed algorithm starts off by generating an initial feasible solution as the starting point of search by the VND procedure. We applied the saving algorithm by Clarke and Wright (1964) to rapidly obtain a solution, which is based on the notion of savings: firstly, dispatch a vehicle to each customer; then merge two routes into a single one which can generate the maximum distance savings. This heuristic method terminates when there are no more two routes can feasibly be merged, i.e., be merged without violating the route duration or capacity constraints.

3.2. Variable neighborhood descent

The VND procedure is to search for better solutions in the neighborhood defined by different operators, which is described by Algorithm 2. The operators used in the VND procedure are four

operators commonly used for standard VRP, i.e., *insert*, *swap*, *2-opt** and *2-opt*. The aim of using multi-operator is to explore the solution space more extensively. When no further improvement can be obtained, the VND procedure stops. The VND procedure is described by Algorithm 2, where *LocalSearch*$(s,\ N_k)$ refers to the local search in the neighborhood of solution s defined by the operator N_k.

Algorithm 2. The procedure of the VND

```
1:  Define the neighborhood N_{k_max}={relocation, swap, 2-opt*, 2-opt}
2:  repeat
3:    improve_tag ⇐ false
4:    k ⇐ 0
5:    while k < k_max do
6:      s* ⇐ LocalSearch(s, N_k)
7:      if f(s*) < f(s) then
8:        improve_tag ⇐ true
9:        s ⇐ s*
10:   end if
11:   k ⇐ k + 1
12:   end while
13:   until improve_tag = false
14:   return s
```

3.2.1. Operators

The operators used in VND are commonly used in the local search based methods for the standard VRP. Considering for completeness, the use of them in our implementation is briefly explained in this section.

The *relocation* consists of removing a customer from its current place and reinserting it into another position in the same route or an alternate one. The steps of local search w.r.t *relocation* are illustrated by Algorithm 3. In our implementation, the local search is terminated after n consecutive trials that no improvement has been obtained. A *swap* move consists of exchanging the places of two customers belonging to different routes. The procedure of local search w.r.t *swap* is quite similar to Algorithm 3 except for step 4. In *LocalSearch*(*s*, *swap*), generate the best neighbor solution s' of s by swapping que_j with another customer.

Algorithm 3. The steps of LocalSearch(*s*, *relocation*)

```
1:  Generate a random permutation que of all the customers
2:  j ⇐ 0, Cnt ⇐ 0
3:  while Cnt < n_c do
4:    generate the best neighbor solution s' of s by relocating que_j
5:    if f(s') < f(s) then
6:      Cnt ⇐ 0
7:      s ⇐ s'
8:    else
9:      Cnt ⇐ Cnt + 1
10:   endif
11:   j ⇐ (j + 1) mod n
12:   end while
13:   return s
```

The *2-opt** operates on two different routes. Firstly, each route is divided into two parts by removing an arc; then, the first part of one route and the second part of the other are combined to generate a new route by introducing a new arc. The remaining two parts build another new route analogously. The steps of the local search w.r.t *2-opt** are shown by Algorithm 4.

Algorithm 4. The steps of *LocalSearch*(*s*, *2-opt**)

```
1:  r1 ⇐ 1, tag ⇐ false, let sub be the number of routes in solution s
2:  repeat
3:    r2 ⇐ r1 + 1
4:    repeat
5:      try to find a neighborhood solution s' that improves s by
      applying 2-opt* on r1 and r2
6:      if s' is found then
7:        tag ⇐ true
8:        s ⇐ s'
9:        break
10:     else
11:       r2 ⇐ r2 + 1
12:     endif
13:   until r2 > r_sub
14:   if tag = true then
15:     r1 ⇐ 0
16:     tag ⇐ false
17:   else
18:     r1 ⇐ r1 + 1
19:   endif
20: until r1 > r_{sub-1}
21: return s
```

The *2-opt* operator is used for intra-route improvement. In a *2-opt* move, two non-adjacent arcs are replaced by another two new ones, and the visited order of the customers between the two arcs is reversed. In the local search w.r.t *2-opt*, apply *2-opt* to each route until no further improvement can be obtained.

3.2.2. Restricted neighborhood

Since the neighborhood based on the above four operators would be very large, the idea of granular neighborhoods by Toth and Vigo (2003) is applied in the implementation of the IVND. This idea is based on the observation that long arcs have low probabilities in the best solution, thus a threshold of arc cost is defined by $\delta = \alpha \cdot f(s_0)/(n+k)$, where α is a proper positive parameter; $f(s_0)$ is the objective value of initial solution, and k is the number of vehicle routes. Some arcs are considered to be "important", including those incident to the depot and those belonging to the best solution found so far. In the local search procedures, only such moves are considered that add at least one arc that is either "important" or its cost is no more than the arc threshold δ.

3.3. Perturbation

Once the VND procedure is stopped, a perturbation is started. The perturbation should neither be too strong nor too weak. If the perturbation is too strong, the algorithm may reduce to a random restart method; otherwise, the possibility of escaping from the current local optimum is quite low. In the proposed method, the perturbation is realized by applying a randomly cross-exchange move on the current local optimum. Cross-exchange is firstly proposed by Taillard, Badeau, Gendreau, Guertin, and Potvin (1997) and mostly used for solution improvement in the past. Here, it is used to perturb solution while the change of solution quality is ignored. Fig. 1 illustrates a perturbation move, in which a customer segment of customers on the upper route are exchanged with another on the lower one, while solution feasibility is guaranteed.

More specifically, randomly choose two routes r_1 and r_2 and determine the starting point b_1 and b_2 on them, respectively. The length of customer segment involved in the perturbation are denoted by λ_1 and λ_2, which are random integer number in the range of [2,4]. If the number of customers from $b_1(b_2)$ to the route end is less than $\lambda_1(\lambda_2)$, the remaining customers are all included.

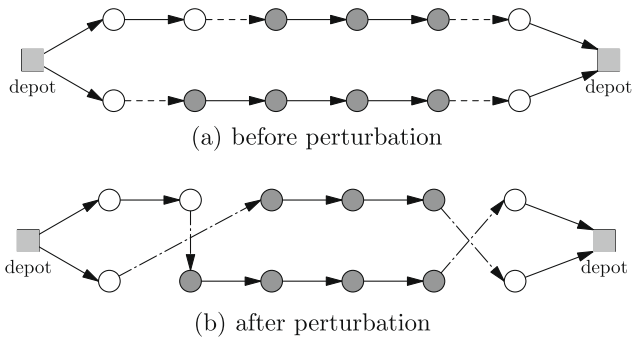


Fig. 1. Illustration of perturbation.

3.4. Acceptance criterion

Acceptance criterion determines which solution is to be perturbed. In the proposed MILS, the acceptance criterion considers both the solution quality and search history information. If it has been for n_b consecutive iterations that no new best solution has been found, use the best found solution s^{**} for perturbation, with the aim to intensify the search around the promising area. $n_b = 50$ is used in our implementation. Otherwise, accept s^* or s'^* by a reminiscent simulated annealing acceptance criterion $SA(s^*, s'^*)$, which does not always accept the better solution, but accepts the worse solution with a certain probability. $SA(s^*, s'^*)$ is given by Eq. (1), where q_0 is randomly generated in the range of [0,1]. The temperature T is initialized as T_0, and updated by $T_{n+1} = \beta \cdot T_n$ every n_T iterations

$$SA(s^*, s'^*) = \begin{cases} s^*, & \text{if } f(s'^*) \geqslant f(s^*) \text{ and } q_0 > \exp(\frac{f(s^*)-f(s'^*)}{T}); \\ s'^*, & \text{if } f(s'^*) < f(s^*) \text{ or } q_0 \leqslant \exp(\frac{f(s^*)-f(s'^*)}{T}). \end{cases} \quad (1)$$

4. Computational results

Computational testings were taken with the aim to evaluate the performance of the proposed IVND. Firstly, the characteristics of the testing instances used in the experiments are described. Then, some discussions are taken about the parameter settings and each component of the proposed heuristic. Finally, the results obtained are reported and some comparisons are drawn between the proposed heuristic and other state-of-the-art heuristics.

4.1. Testing problems

The test problems include 14 benchmark problems by (Christofides & Eilon, 1969) and 20 large scale problems by (Golden, Wasil, Kelly, & Chao, 1998). The main characteristics of these problems are summarized in Tables 1 and 2. For each problem, it is given in these tables that the number of customers (n) and available vehicles (K), the vehicle capacity (Q), the route duration upper bound (D) and the service time (st_i) needed by each customer. Besides, the previous best known solutions are also given in column *Best*, which can be obtained from the webpage of Stefan Ropke (http://www.diku.dk/~sropke/). As for 14 benchmark problems, it should be mentioned that customers in problems C1–C10 are randomly distributed, whereas in problems C11–C14 customers are distributed in clusters.

Table 1
Characteristics of 14 benchmark problems.

Prob.	n	K	Q	D	st_i	Best
C1	50	5	160	–	–	524.61
C2	75	10	140	–	–	835.26
C3	100	8	200	–	–	826.14
C4	150	12	200	–	–	1028.42
C5	199	17	200	–	–	1291.29
C6	50	6	160	200	10	555.43
C7	75	11	140	160	10	909.68
C8	100	9	200	230	10	865.94
C9	150	14	200	200	10	1162.55
C10	199	18	200	200	10	1395.85
C11	120	7	200	–	–	1042.11
C12	100	10	200	–	–	819.56
C13	120	11	200	720	50	1541.14
C14	100	11	200	1040	90	866.37

Table 2
Characteristics of 20 large scale problems.

Prob.	n	K	Q	D	st_i	Best
P1	240	10	550	650	0	5,627.54
P2	320	10	700	900	0	8,447.92
P3	400	10	900	1200	0	11,036.22
P4	480	12	1000	1600	0	13,624.52
P5	200	5	900	1800	0	6,460.98
P6	280	8	900	1500	0	8,412.80
P7	360	9	900	1300	0	10,181.75
P8	440	11	900	1200	0	11,663.55
P9	255	14	1000	–	–	580.60
P10	323	16	1000	–	–	738.92
P11	399	18	1000	–	–	917.17
P12	483	19	1000	–	–	1,107.19
P13	252	27	1000	–	–	857.19
P14	320	30	1000	–	–	1,080.55
P15	396	34	1000	–	–	1,340.24
P16	480	38	1000	–	–	1,622.69
P17	240	22	200	–	–	707.76
P18	300	28	200	–	–	995.39
P19	360	33	200	–	–	1,366.14
P20	420	41	200	–	–	1,820.09

4.2. Parameters setting

The proposed IVND was coded in C++ and run on a Pentium IV 2.93 GHz PC. It is terminated when the best found solution has not been updated for n_t consecutive iterations. In order to determine appropriate settings for parameters, some preliminary experiments have been carried out. In our implementation, we used $T_0 = 2\% \cdot f(s_0^*)$, $\beta = 0.9$, $n_T = 30$, $n_t = \min\{4n, 600\}$.

As parameter α which determine the restricted neighborhood is important for both the accuracy and the convergency of the proposed heuristic. If α is too big, the idea of restricted neighborhood may do not work at all; otherwise, solution quality may be affected. Therefore, a proper value for parameter α plays an important role in the balance of diversification and intensification. In our implementation, α is determined by experiments. A series of algorithms for each benchmark problem with only different setting of α has been carried out, and each of them stops after 1000 iterations. Considering the inherent randomness of the proposed heuristic, each algorithm has run 10 times and the results are given by Fig. 2. In Fig. 2a, *best* denotes the ARPD of the best results obtained, while *avg.* denotes the ARPD of the average results over 10 runs. Here, ARPD is the average RPD over 14 problems. RPD represents the relative percent deviation from the previous best known solution, and is calculated by $RPD = (F_A - F_{best})/F_{best} \times 100$, where F_A is the result found by heuristic A and F_{best} is the best previous known result. Fig. 2b shows the average accumulated computing time in terms of seconds for one run.

According to the *avg.*, it seems that the proposed heuristic performs quite well when α equals to 2.0, 2.5, 3.0 and 3.5, while little difference exists among them. As the *best* has the best results as

$\alpha = 2.5$, so it was adopted in our implementation. Obviously, the bigger α is, the more computing time need as more neighborhood solutions would be examined, which is also demonstrated by the experiment results shown by Fig. 2b.

4.3. Effect of multi-operator

Several validation experiments have been carried out with the aim to examine the effect of multi-operator. Each experiment, denoted by *A*, excludes the operator *A* from the VND procedure while other algorithm settings are all kept the same. It is worth mentioning that the experiment *none* means no operator is excluded. The results of these experiments are shown by Fig. 3.

From Fig. 3a, it can be intuitively seen that the algorithm with none operator excluded (*none*) performs best in terms of solution quality among all five versions, which indicates that each operator contributes to the algorithm. The operators *relocation, swap* and *2-opt** seems effect more than *2-opt* according to the ARPP, however, the algorithm without *2-opt* needs more computing time. These experimental results show that the use of multi-operator does work to enhance the heuristic.

4.4. Effect of the perturbation strategy

In order to validate the effectiveness of the proposed perturbation strategy, we compare the performance of two experiments of the proposed heuristic, e.g., one with the restarting points generated with perturbation and the other with those by random initialization methods, which generate a feasible solution by separating a giant tour into several routes according to the capacity and route duration restrictions. Fig. 4 shows the average results on each problem obtained by these two algorithms. From this figure, it can be intuitively seen that the one with perturbation has always performed better than the one with randomly starting points, and the difference is even significant for some problems, from

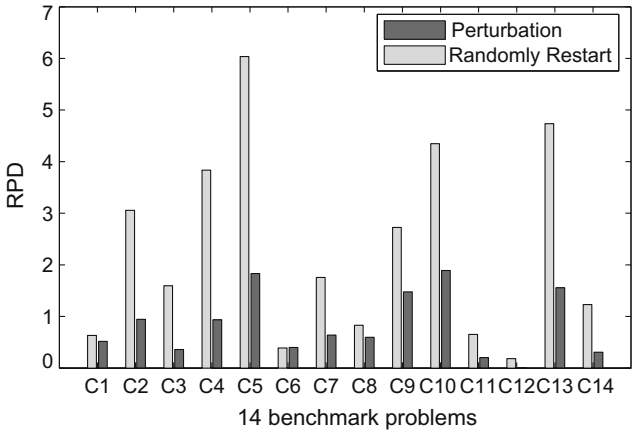
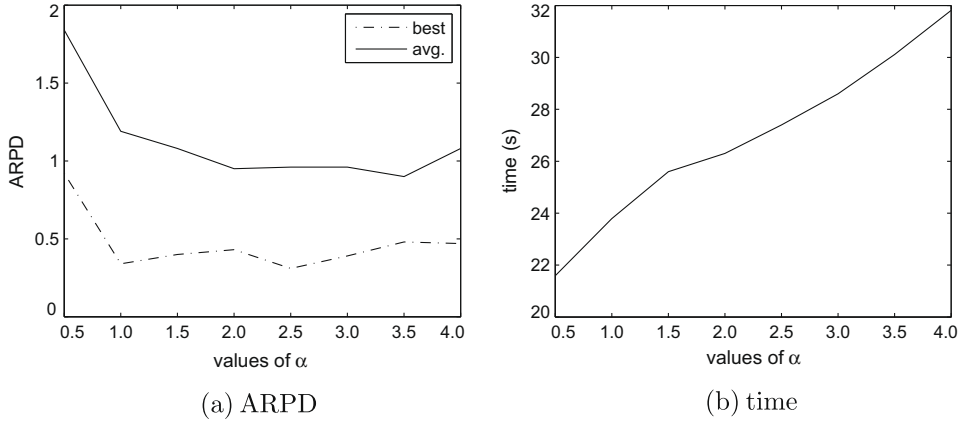
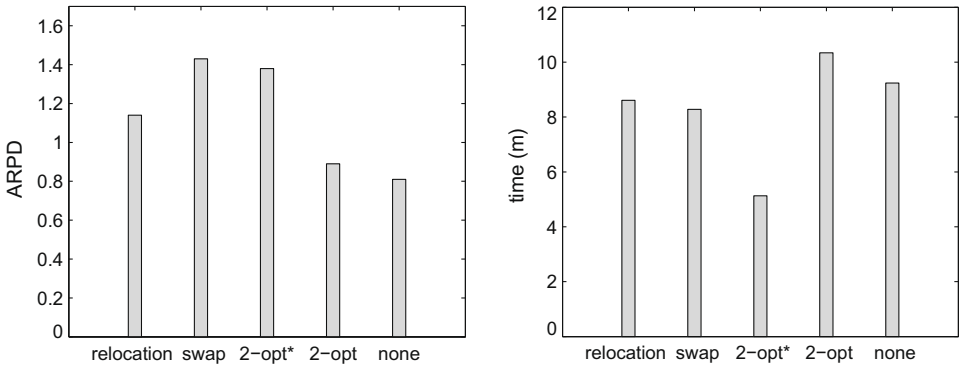


Fig. 4. Performance of the perturbation strategy.



(a) ARPD (b) time

Fig. 2. Determination of α.



(a) ARPD (b) time

Fig. 3. Effect of multi-operator.

which we can draw the conclusion that the proposed perturbation strategy indeed works.

4.5. Results for testing problems

Tables 3 and 4 report the results obtained by the proposed heuristic IVND for 14 classical benchmark problems and 20 large scale problems, respectively. The results are the best ones over 10 independent runs. Additionally, results obtained by recent metaheuristics from literature are also listed in these tables, which include the *USTA* by Cordeau et al. (2001), the *BR* by Tarantilis and Kiranoudis (2002), the *GTS* by Toth and Vigo (2003), the *EA* by Prins (2004), the *D-Ants* by Reimann et al. (2004), the *HGA* by Berger and Barkaoui (2004), the *VRTR* by Li et al. (2005), the *SEPAS* with the best parameter setting by Tarantilis (2005), the *VLNS* by Ergun et al. (2006), the *EAC* by Nagata (2007), the *GH* by Pisinger and Ropke (2007), the *ABHC Sav_S* and *ABHC Sav_P* by Derigs and Kaiser (2007), the *VNS* and *GVNS* by Kytöjoki, Nuortio, Bräysy, and Gendreau (2007), the *AGES(best)* and *AGES(fast)* by Mester and Bräysy (2007), and the *HST* by Lin et al. (2009). These results are reported in terms of RPD from the previous best known solution s_{best}. Tables 3 and 4 have the same table structure, in which column *k* shows the number of vehicles used in the solution obtained by the IVND; row *ARPD* gives the average *RPD* over all the 14 instances; row *Computer* reports the CPU used to run the corresponding method, while row *CPU min* gives the average computing time per problem in minutes. In Table 3, it should be noted that results in column *EAC*, column *GH* and column *IVND* are the best solutions over 10 runs and thus the *CPU min* are the accumulated time for 10 runs. Results in column *SEPAS* were obtained by the algorithm with its best parameter settings.

From Table 3, it can be observed that the proposed IVND found the best known solutions to most problems except for C5, C9–C10 and C13. The average RPD over all problems is 0.12, which is lower than those of other approaches except for the EAC by Nagata (2007), the GH by Pisinger and Ropke (2007) and the AGES by Mester and Bräysy (2007). However, the IVND can find more best known solutions than the GH. The average accumulated computing time per problem for 10 runs is 10.9 min on a Pentium IV 2.93 GHz PC, which indicates that the proposed IVND is quite fast, at least not inferior to other heuristics even the computer differences are considered. For large scale problems, Table 4 shows that the proposed IVND found the best known solutions to 3 problems, i.e., P4–P6. The average RPD over 20 problems is 0.67, which indicates that the proposed IVND outperforms or is quite competitive with most of other heuristics, except for the AGES (best) by Mester and Bräysy (2007). The average accumulated computing time per problem is 284.4 min for 10 runs.

Table 3
Results for 14 classical benchmark problems.

Prob.	k	USTA	BR	GTS	EA	HGA	VRTR	SEPAS	VLNS
C1	5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C2	10	0.00	0.00	0.40	0.00	0.00	0.11	0.00	0.02
C3	8	0.00	0.00	0.29	0.00	0.15	0.15	0.00	0.00
C4	12	0.41	0.24	0.47	0.31	0.75	1.65	0.00	0.76
C5	17	1.90	1.77	2.09	0.69	2.54	0.94	1.56	1.24
C6	6	0.00	0.00	0.00	0.00	0.00	–	0.00	0.00
C7	11	0.00	0.00	1.21	0.29	0.00	–	0.00	0.04
C8	9	0.00	0.00	0.41	0.00	0.27	–	0.00	0.00
C9	14	0.46	0.06	0.91	0.15	0.57	–	0.00	0.20
C10	18	1.50	0.93	2.86	1.74	1.64	–	0.81	0.61
C11	7	3.01	0.00	0.07	0.00	0.10	0.00	0.00	0.00
C12	10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C13	11	0.53	0.19	0.28	0.12	0.78	–	0.19	0.25
C14	11	0.00	0.00	0.00	0.00	0.00	–	0.00	0.00
Number		8	9	4	8	6	3	11	6
ARPD		0.56	0.23	0.64	0.24	0.49	0.41	0.18	0.23
CPU min		24.62	5.22	3.84	5.19	21.25	–	6.96	28.91
Computer		Pentium IV 2G	Pentium II 400M	Pentium 200M	GHz PC 75 MFlops	Pentium II 400M	–	Pentium II 400M	Pentium III 733M

Prob.	k	EAC	GH	ABHC		AGES		HST	IVND
				Sav_S	Sav_P	best	fast		
C1	5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C2	10	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.00
C3	8	0.00	0.00	0.00	0.15	0.00	0.00	0.00	0.00
C4	12	0.00	0.11	0.12	0.26	0.00	0.00	1.00	0.00
C5	17	0.00	0.45	1.01	2.64	0.00	0.23	1.58	0.47
C6	6	–	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C7	11	–	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C8	9	–	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C9	14	–	0.10	0.56	0.14	0.00	0.17	0.03	0.41
C10	18	–	0.72	0.87	0.63	0.38	0.63	1.08	0.75
C11	7	0.00	0.00	0.00	0.00	0.00	0.00	0.33	0.00
C12	10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C13	11	–	0.11	0.35	0.11	0.00	0.14	0.89	0.12
C14	11	–	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Number		7	9	8	8	13	10	9	10
ARPD		0.00	0.11	0.21	0.28	0.03	0.08	0.35	0.13
CPU min		23.74	17.43	5.84	4.54	2.71	0.05	8.21	10.9
Computer		Xeon 3.2G	Pentium IV 3G	Celeron 2.4G		Pentium IV 2.8G		Pentium IV 2.8G	Pentium IV 2.93G

Table 4
Results for 20 large scale instances.

Prob.	k	USTA	BR	GTS	EA	D-ants	VRTR	SEPAS	VLNS	EAC
P1	9	0.97	0.88	1.93	0.34	0.29	0.69	0.88	2.03	–
P2	10	2.48	0.77	1.24	0.00	0.01	0.25	0.14	5.56	–
P3	10	0.01	1.48	3.32	0.00	0.00	0.99	0.00	9.70	–
P4	10	0.85	0.10	9.44	0.00	0.55	0.98	0.10	12.42	–
P5	5	4.57	0.00	3.66	0.00	0.00	0.26	0.00	1.69	–
P6	7	1.48	0.20	6.54	0.00	0.00	1.51	0.02	5.03	–
P7	9	0.84	0.34	3.59	0.14	0.14	1.06	0.34	9.18	–
P8	11	1.77	2.34	3.20	1.42	1.42	2.20	2.34	8.32	–
P9	14	1.17	–	2.20	1.88	1.08	1.32	0.83	1.26	0.00
P10	16	1.87	–	1.72	1.69	1.60	1.43	1.03	1.48	0.00
P11	18	1.30	–	2.06	1.73	1.10	0.95	0.65	1.70	0.00
P12	19	1.11	–	3.61	2.40	3.04	1.88	2.10	2.48	0.12
P13	26	2.18	–	1.35	2.10	0.92	0.93	0.91	1.60	0.00
P14	30	1.99	–	1.45	0.53	1.22	1.59	0.51	1.53	0.00
P15	33	1.75	–	2.18	2.02	1.34	1.58	1.02	2.01	0.00
P16	37	1.50	–	1.83	1.74	0.77	0.79	0.74	1.25	33.81
P17	22	0.45	–	0.47	0.38	0.14	0.56	0.14	1.23	0.00
P18	28	1.93	–	2.15	1.95	0.35	1.50	1.16	2.81	0.00
P19	33	1.29	–	2.55	0.76	0.08	1.20	0.36	2.83	0.00
P20	38	1.88	–	5.26	1.45	0.16	1.69	0.97	3.47	0.02
Number		0	1	0	5	2	0	1	0	9
ARPD		1.57	0.76	2.99	1.03	0.71	1.05	0.71	3.88	2.83
CPU min		56.11	–	17.55	66.9	49.33	–	–	137.05	413.68
Computer		Pentium IV 2G	Pentium II 400M	Pentium 200M	Pentium III 1G	Pentium 900M	–	–	Pentium III 733M	Xeon 3.2G

Prob.	k	GH	ABHC		VNS	GVNS	AGES		HST	IVND
			Sav_S	Sav_P			best	fast		
P1	9	0.42	0.30	0.24	4.27	4.27	0.00	0.29	2.99	0.57
P2	10	0.25	0.03	0.18	0.34	0.34	0.00	0.24	0.64	0.20
P3	10	0.10	0.51	0.04	0.07	0.07	0.00	0.99	2.98	0.21
P4	10	0.08	0.80	0.83	0.10	0.05	0.00	0.59	3.76	0.00
P5	5	0.09	0.00	0.00	0.00	0.00	0.00	0.00	0.79	0.00
P6	7	0.04	0.00	0.01	0.03	0.03	0.00	1.51	1.67	0.00
P7	9	0.00	0.14	0.84	1.14	1.14	0.14	0.58	2.37	0.84
P8	11	0.43	1.80	1.77	1.82	1.79	0.00	2.19	2.77	0.97
P9	14	0.78	1.42	0.78	6.90	6.90	0.48	1.35	1.05	0.80
P10	16	1.35	1.17	0.94	8.39	6.21	0.36	1.89	1.35	0.90
P11	18	0.60	1.38	0.62	9.15	7.59	0.14	0.96	0.82	0.65
P12	19	1.07	1.00	1.04	10.25	9.20	0.00	1.22	1.67	0.97
P13	26	0.87	0.92	0.97	10.28	8.01	0.22	0.93	1.18	0.52
P14	30	1.37	1.41	1.43	9.52	6.91	0.07	1.59	1.69	0.77
P15	33	1.47	1.22	1.11	10.17	9.05	0.37	1.08	1.22	1.20
P16	37	1.01	1.50	1.16	8.19	7.41	0.00	0.76	1.25	0.89
P17	22	0.16	0.27	0.25	5.09	2.58	0.00	0.35	0.64	0.16
P18	28	0.71	0.95	0.73	11.05	8.25	0.34	1.42	2.26	1.52
P19	33	0.59	1.12	1.26	6.53	5.74	0.05	1.15	1.38	1.21
P20	38	0.59	1.45	1.64	9.32	6.48	0.00	1.13	1.97	1.09
Number		1	3	1	1	1	11	1	0	3
ARPD		0.60	0.87	0.79	5.63	4.60	0.11	1.01	1.72	0.67
CPU min		107.61	109.63	111.10	0.18	1.01	24.35	0.22	118.98	284.4
Computer		Pentium IV 3G	Celeron 2.4G		AMD Athlon64 3000+		Pentium IV 2.8G		Pentium IV 2.8G	Pentium IV 2.93G

5. Conclusion

In this paper, a hybrid metaheuristic method IVND is proposed, which is based on five conventional neighborhood operators used for the CVRP, e.g., *relocation, swap, 2-opt*, 2-opt* and *cross-exchange*. The former four are used in a VND scheme for solution improvement, while the fifth is for solution perturbation to generate new starting points. The IVND has a such rather simple algorithm structure that is easy to implement. Computational results on 34 CVRP benchmark problems demonstrate that the proposed IVND is efficient and quite competitive with other state-of-the-art heuristics for the CVRP. Since the heuristic strategies of the proposed IVND are problem independent, it is flexible that can be applied to solving other combinatorial optimization problems in future.

Acknowledgement

This work is supported by the National Basic Research Program (973 program) of China (Project Ref. 2006CB7055000).

References

Berger, J., & Barkaoui, M. (2004). A new hybrid genetic algorithm for the capacitated vehicle routing problem. *Journal of the Operational Research Society, 54*, 1254–1262.

Christofides, N., & Eilon, S. (1969). An algorithm for the vehicle dispatching problem. *Operational Research Quarterly, 20*, 309–318.

Christofides, N., Mingozzi, A., & Toth, P. (1979). The vehicle routing problem. In N. Christofides, A. Mingozzi, P. Toth, & C. Sandi (Eds.), *Combinatorial optimization* (pp. 315–338). Chichester, UK: Wiley.

Clarke, G., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research, 12*, 568–581.

Cordeau, J. F., Gendreau, M., Hertz, A., Laporte, G., & Sormany, J. S. (2005). New heuristics for the vehicle routing problem. In A. Langevin & D. Riopel (Eds.), *Logistics systems: Design and optimization* (pp. 279–297). New York: Springer.

Cordeau, J. F., Gendreau, M., Laporte, G., Potvin, J. Y., & Semet, F. (2002). A guide to vehicle routing heuristics. *Journal of the Operational Research Society, 53*, 512–522.

Cordeau, J. F., Laporte, G., & Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society, 52*, 928–936.

Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. *Management Science, 6*, 80–91.

Derigs, U., & Kaiser, R. (2007). Applying the attribute based hill climber heuristic to the vehicle routing problem. *European Journal of Operational Research, 177*, 719–732.

Ergun, Ö., Orlin, J. B., & Steele-Feldman, A. (2006). Creating very large scale neighborhoods out of smaller ones by compounding moves. *Journal of Heuristics, 12*, 115–140.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. New York: W.H. Freeman and Company.

Gendreau, M., Laporte, G., & Potvin, J. Y. (2002). Metaheuristics for the capacitated VRP. In P. Toth & D. Vigo (Eds.), *The vehicle routing problem* (pp. 129–154). Philadelphia: SIAM Publishing.

Golden, B. L., Wasil, E. A., Kelly, J. P., & Chao, I. (1998). The impact of metaheuristics on solving the vehicle routing problem: Algorithms, problem sets and computational results. In T. G. Grainic & G. Laporte (Eds.), *Fleet management and logistics* (pp. 33–56). Boston: Kluwer.

Hansen, P., & Mladenović, N. (2003). Variable neighborhood search. In F. W. Glover & G. A. Kochenberger (Eds.), *Handbook of metaheuristics* (pp. 145–184). Springer.

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*, 2743–2757.

Laporte, G., & Semet, F. (2002). Classical heuristics for the capacitated VRP. In P. Toth & D. Vigo (Eds.), *The vehicle routing problem* (pp. 110–111). Philadelphia: SIAM Publishing.

Li, F., Golden, B., & Wasil, E. (2005). Very large-scale vehicle routing: New test problems, algorithms, and results. *Computers & Operations Research, 32*, 1165–1179.

Lin, S. W., Lee, Z. J., Ying, K. C., & Lee, C. Y. (2009). Applying hybrid meta-heuristics for capacitated vehicle routing problem. *Expert Systems with Applications, 36*(2P1), 1505–1512.

Lourenço, H. R., Martin, O. C., & Stützle, T. (2003). Iterated local search. In F. W. Glover & G. A. Kochenberger (Eds.), *Handbook of metaheuristics* (pp. 321–368). Springer.

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

Mester, D., & Bräysy, O. (2007). Active guided evolution strategies for the large scale capacitated vehicle routing problems. *Computers & Operations Research, 34*, 2964–2975.

Mole, R. H., & Jameson, S. R. (1976). A sequential route-building algorithm employing a generalized savings criterion. *Operational Research Quarterly, 27*, 503–511.

Nagata, Y. (2007). Edge assembly crossover for the capacitated vehicle routing problem. *Evolutionary Computation in Combinatorial Optimization, LNCS, 4446*, 142–153.

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

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research, 31*, 1985–2002.

Reimann, M., Doerner, K., & Hartl, R. F. (2004). D-ants: savings based ants divide and conquer the vehicle routing problem. *Computers & Operations Research, 31*, 563–591.

Taillard, E., Badeau, P., Gendreau, M., Guertin, F., & Potvin, J. Y. (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science, 31*, 170–186.

Tarantilis, C. D. (2005). Solving the vehicle routing problem with adaptive memory programming methodology. *Computers & Operations Research, 32*, 2309–2327.

Tarantilis, C. D., & Kiranoudis, C. T. (2002). Bone route: An adaptive memory-based method for effective fleet management. *Annals of Operations Research, 115*, 227–241.

Toth, P., & Vigo, D. (2003). The granular tabu search and its application to the vehicle routing problem. *INFORMS Journal of Computing, 15*, 333–348.