Journal of Computer Science and Cybernetics, V.xx, N.x (20xx), 1–
DOI no. 10.15625/1813-9663/xx/x/xxxx

SOLVING MIN-MAX CAPACITATED VEHICLE ROUTING PROBLEM BY LOCAL SEARCH

Abstract. Vehicle routing is a class of combinatorial optimization problems in transportation and logistics. Min-max capacitated vehicle routing is a problem of this class in which the length of the longest route must be minimized. This paper investigates local search approach for solving the min-max capacitated vehicle routing problem with different neighborhood structures. We also propose a combined function instead of the objective function itself for controlling the local search. Experimental results on different datasets show the efficiency of our proposed algorithms compared to previous techniques.
Keywords. vehicle routing, local search, min-max vehicle routing, combinatorial optimization

1. Introduction

A large number of applications involve sets of clients that must be served by vehicles located at a common depot. Problems which optimize the selection of routes for the vehicles, are referred to as *vehicle routing problem* [1, 2]. Solving these problems is very hard and is still an active research topic which attracts the attention of many computer scientists due to their impact to the society and the economy. Many variants of vehicle routing applications have been studied in the literature, for example, Capacitated Vehicle Routing problem (CVRP) [3], Min-Max Vehicle Routing Problem (MMVRP) [4], Vehicle Routing Problem with Time Windows (VRPTW) [5], etc.

We consider in this paper the Min-Max Capacitated Vehicle Routing Problem (MMCVRP). The goal of this problem is to ensure that all clients are served as soon as possible such that the total load of each vehicle does not exceed a predefined value. Due to the hardness of MMVRP, a simpler variant of this problem, MMCVRP is a NP-hard problem [6].

Figure 1 illustrates an example of MMCVRP with 3 clients 1, 2, 3 and the depot 0. The demand of each client is 2. There are two vehicles in which the capacity of each vehicle is 5. In this example, there are 6 solutions (see Figure 1). Among these 6 solutions, the best solution is the solution 2 (Figure 1b) with the objective 5.

1.1. Problem formulation

This section describes the formulation of the min-max capacitated vehicle routing problem.

Input The input consists of following elements:

- N: number of client points
- $C = \{p_1, \dots, p_N\}$: client points
- K: number of vehicles
- s_i, t_i: starting and terminating points of vehicle i ($\forall i = 1, \dots, K$). In case all vehicles depart and return to the same depot, s_i, t_i refer to this physical depot

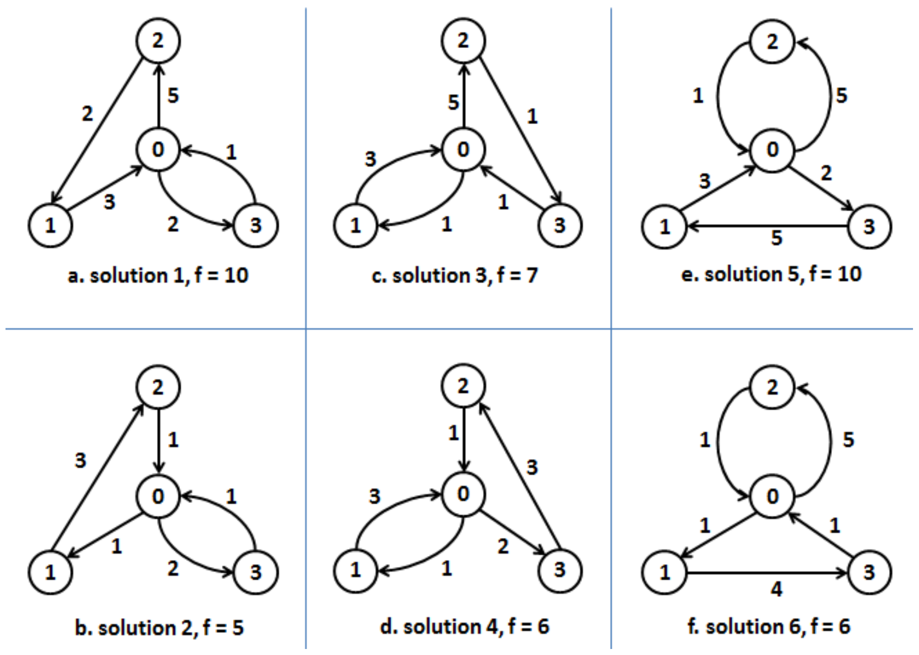


Figure 1: Illustrating example of MMCVRP

- c^i: the capacity of vehicle $i, \forall i = 1, \dots, K$
- $T = \{t_1, \dots, t_K\}$: set of terminating points of vehicles
- $B = \{s_1, \dots, s_K\}$: set of starting points of vehicles
- $D = B \cup T$: set of starting and terminating points of vehicles
- $V = C \cup D$: set of all points
- $c_{p,q}$: the distance from point p to point q, $\forall p, q \in V$
- d_p: the demand of client point p ($\forall p \in V$). $d_p = 0, \forall p \in D$ by convention

Variables

- Decision variables x_p: the successor of point p in the solution. Domain of x_p is $V \setminus B$ ($\forall p \in V \setminus T$)
- Auxiliary variables l_p: represents accumulated demand on the route visiting point p from the starting point of this route until p ($\forall p \in V$).
- Auxiliary variables td_p: represents total distance on the route containing p from the starting point of this route to p ($\forall p \in V$)
- Auxiliary variables I_p: represents the index of route containing point p ($\forall p \in V$)
- Auxiliary variables f: represents the length of the longest path of the solution

Constraints

$$I_{s_i} = i, \forall i = 1, \ldots, K \tag{1}$$

$$I_{t_i} = i, \forall i = 1, \ldots, K \tag{2}$$

$$I_{x_p} = I_p, \forall p \in V \setminus T \tag{3}$$

$$l_{x_p} = l_p + d_p, \forall p \in V \setminus T \tag{4}$$

$$td_{x_p} = td_p + c_{p,x_p}, \forall p \in V \setminus T \tag{5}$$

$$l_{t_i} \leq c^i, \forall i = 1, \ldots, K \tag{6}$$

$$f \geq td_{t_i}, \forall i = 1, \ldots, K \tag{7}$$

Constraints 1, 2 define the index of route containing starting and terminating points of this route. Constraint 3 specifies that a point and its successor must be in the same route. Constraints 1, 2, and 3 ensure that each client is serviced exactly once. Constraints 4 and 5 specify the relation between accumulated demand and distance of two consecutive points on the same route. Constraint 6 is the capacity constraint. Constraint 7 describes the objective function of the problem.

Objective Function The objective function to be minimized is f.

1.2. Related works

Dantzig and Ramser were the first scientists who introduce the "Truck Dispatching Problem" in [7], modelling how a fleet of homogeneous trucks could serve the demand for oil of a number of gas stations from a central hub and with a minimum travel distance. This became known as the 'Vehicle Routing Problem' (VRP), one of the most widely studied topics in the field of Operations Research. The study conducted by Eksioglu et al. in [8] revealed 1021 journal articles with VRP as the main topic, published between 1959 and 2008. The number of solution methods introduced in the academic literature (for old as well as new variants of the VRP) has grown rapidly over the past decades. Moreover, the processing speed and memory capacity of current computers has increased significantly, enabling to solve larger instances of the VRP which spurs the progression in the research field and the development of commercial software for the VRP. According to a recent survey [9], thousands of companies, among others Coca-Cola Enterprises and Anheus-Bush Inbev, nowadays use VRP softwares.

CVRP differs from MMCVRP when its objective is to minimize the total length of routes. The practical importance of this problem leads to much motivation for the effort involved in the development of heuristic algorithms [10, 11] and exact algorithms [12, 13]. In [12], Baldacci et al. have described a branch-and-cut algorithm that is based on a two commodity network flow formulation of the CVRP. The algorithm proposed in [13] is very consistent on solving instances from the literature with up to 135 customers. For the latest research on CVRP, we refer to the researches [11, 14, 15, 16, 17].

VRPTW is an extension of CVRP when it considers additionally time window constraints that the clients must be served within predefined time windows. The most interest meta-heuristics used to solve the VRPTW are Tabu search (TS), genetic algorithm (GA), evolutionary algorithms (EA) and ant colony optimisation algorithm (ACO) [18]. Meta-heuristic controls local search processes, such as tabu search [19, 20], simulated annealing [21], genetic algorithms [22], ... Meta-heuristics controlling a subordinate construction heuristic, such as the greedy randomized search procedure (GRASP) proposed by [23], the RNET meta-heuristic [24] and multiple ant colony systems as proposed by [25]. [26] proposed an exact algorithm for the multiple vehicle routing problem with time windows.

Vehicle Routing Problem with Pick-up and Delivery (VRPPD), which models a real-life problem, is much more complicated than the classical VRP. The problem arises in practice when items need to be transported from the depot to customers and also need to be picked up at customers and brought back to the depot. There are many works that focus on solving this problem, for example in [27] the authors proposed a neighborhood search heuristics to optimize the planned routes of vehicles.

The min-max vehicle routing problem without capacity constraint (MMVRP) have been considered in the literature. Applegate et al. [4] proposed a branch and cut algorithm for solving this problem. Incomplete algorithms have also been proposed, for instance, approximation algorithm [6], neighborhood search algorithms [28], [29], [30], a genetic algorithm [31]. For the min-max capacitate vehicle routing problem (MMCVRP), Golden et al. proposed an algorithm [32] which consists of four distinct steps: initial CVRP solutions, generation of new CVRP solutions, recombination of CVRP solutions, and generation of MMCVRP solutions. Most recently, authors of [33] proposed a local search algorithm for solving MMCVRP using 2 kinds of neighborhoods: one-point move and cross-exchange move which will be detailed later.

In this paper, we propose a local search algorithm for solving MMCVRP. Our algorithm exploits various neighborhood structures proposed in the literature. Moreover, we propose to use a combined function for controlling the search instead of the objective function itself. This combined function will be shown to be efficient in the experiments. The paper is organized as follows. Section 2 describes the proposed local search algorithm. Section 3 presents the experiments. Section 4 concludes that paper and draws some future works.

2. Proposed local search algorithms

We describe in this section the proposed local search algorithm for solving MMCVRP. We start by presenting different neighborhoods proposed in the literature.

2.1. Neighborhoods

The neighborhoods we consider in our algorithm are described in [34] including one-point-move, two-point-move, two-opt-move, or-opt-move neighborhood, three-opt-move, three-point-move, cross-exchange neighborhoods. Due to lack of space, we do not present in detail these neighborhoods. Interested readers can refer to [34] for more detail about these neighborhood structures.

2.2. Local search algorithms

Before describing the local search algorithm, we propose a combined function that will be used as the quality function for controlling the search and which is described below.

2.2.1. Quality function

One of the core of a local search algorithm is a function F that models the quality of solutions. This function is also used to control the local search. Basically, F is the objective function itself, i.e., the length of the longest route among K routes of the solution. In this paper, we propose to combine in a lexicographic order the objective function and the total length of K routes into the control function F. The motivation for this combined function is explained as follows. The solution consists of K

routes in which there might be several routes having the same longest length. A local move on a solution may change only one or two routes, thus cannot reduce the lengths of all longest routes of the solution. In this situation, the objective function cannot differentiate neighbors and the current solution (the presence of a plateau). By combining the objective function and the sum of lengths of K routes to establish the control function F, a best local move with respect to F may keep the objective function unchanged but reduce the total length of K routes (the number of longest routes may reduce). This bring opportunity to reduce the objective function in subsequent best local moves. As the min-max vehicle routing problem has a constraint on the capacity of vehicles, the satisfaction of this constraint must be prioritized most. To this end, the control function F consists of three components in a lexicographic order: the violations of the capacity constraint, the original objective function and the sum of lengths of K routes.

Formally, given a solution $s = \{r_1, \dots, r_K\}$ to the min-max vehicle routing problem which consists K routes $r_1, \dots, r_K$, we denote

- $td(r_i)$ the length of route r_i
- $l(r_i)$ the total demand of clients on route r_i
- $f(s) = \max_{i=1,\dots,K}\{td(r_i)\}$ the objective function of the problem
- $v(r_i) = \max\{0, l(r_i) - c^i\}$ the violations of the capacity constraint related to route r_i
- $v(s) = \sum_{i=1}^{K} v(r_i)$ the violations of the capacity constraint
- $t(s) = \sum_{i=1}^{K} td(r_i)$ the total length of the solution
- $F(s) = \langle v(s), f(s), t(s)\rangle$ the quality function

The function F is treated in a lexicographic order: given two solutions s_1 and s_2, we denote $F(s_1) < F(s_2)$ if:

- $v(s_1) < v(s_2)$ or
- $v(s_1) = v(s_2)$ and $f(s_1) < f(s_2)$ or
- $v(s_1) = v(s_2)$ and $f(s_1) = f(s_2)$ and $t(s_1) < t(s_2)$

2.2.2. Local search

Input:

- $(s_1, t_1), \ldots, (s_K, t_K)$ in which s_i and t_i are the starting and terminating points of the route of vehicle i
- $\mathcal{C}$: set of client points
- $\mathcal{L}$: list of neighborhoods
- Control Function F that measures the quality of solutions

Output: Set of K routes

```
1  $s \leftarrow$GenerateInitialSolution($(s_1, t_1), \ldots, (s_K, t_K), \mathcal{C}, F$);
2  $s^* \leftarrow s$;
3  $nic \leftarrow 1$;
4  while time limit is not expired do
5      Shuffle($\mathcal{L}$);
6      $S \leftarrow \{\}$;
7      $e \leftarrow \infty$;
8      foreach neighborhood $N_i$ in $\mathcal{L}$ do
9          $\langle S, e \rangle \leftarrow$ Explore($N_i, S, e$);
10         if $e < F(s)$ then
11             BREAK;
12         end
13     end
14     $s \leftarrow$ select($S$);
15     if $F(s) < F(s^*)$ then
16         $s^* \leftarrow s$;
17         $nic \leftarrow 1$;
18     else
19         $nic \leftarrow nic + 1$;
20         if $nic > maxStable$ then
21             $s \leftarrow$GenerateInitialSolution($(s_1, t_1), \ldots, (s_K, t_K), \mathcal{C}, F$);
22             $nic \leftarrow 1$;
23         end
24     end
25 end
```

Algorithm 1: LSMMCVRP($(s_1, t_1), \ldots, (s_K, t_K), \mathcal{C}, \mathcal{L}, F$)

The proposed local search is depicted in Algorithm 1. It receives $\mathcal{C}$ as a set of client points, s_i and t_i are the starting and terminating points of the route of vehicle i ($\forall i = 1, \ldots, K$), $\mathcal{L}$ is a list of considered neighborhoods, and a control function F that measures the quality of solutions. The initial solution is generated in line 1 which will be detailed in Algorithm 2. Line 2 updates the best solution found so far s^*. At each iteration of the local search, line 5 shuffles that order of the neighborhoods of $\mathcal{L}$. Lines 8–13 iteratively explore these neighborhoods. Each neighborhood exploration (see Algorithm 3 for more detail) will return a set of selected neighbors S which have the same quality evaluation e. The neighborhood exploration will terminate whenever it discovers a first neighbor which is better than the current solution s (lines 10–12). Line 14 replaces the current solution by a randomly selected neighbor of S. If the selected neighbor is better than the best solution found so far s^*, then s^* is updated (lines 15–16). Otherwise, the search augment the number of consecutive iterations *nic* in which no improvement is found by 1. The search will be restarted if *nic* exceeds a given parameters *maxStable* (see lines 20–23).

```
Input: $(s_1, t_1), \ldots, (s_K, t_K)$ in which $s_i$ and $t_i$ are the starting and terminating points
       (logical points) of the route of vehicle $i$. They refer to the same physical depot of the
       given problem
Output: set of $K$ routes
1  $S \leftarrow \mathcal{C}$;
2  $R \leftarrow \{s_1, \ldots, s_K\}$ ;
3  foreach do
4      $r_i \leftarrow \langle s_i, t_i \rangle$;
5  end
6  $s \leftarrow \{r_1, \ldots, r_K\}$;
7  while $S \neq \emptyset$ do
8      $Cand \leftarrow \{\}$;
9      $e^* \leftarrow \infty$;
10     foreach $p \in S$ do
11         foreach $q \in R$ do
12             $s' \leftarrow$ AddOnePoint$(s, p, q)$;
13             if $F(s') < e^*$ then
14                 $Cand \leftarrow \{\langle p, q \rangle\}$;
15                 $e^* \leftarrow F(s')$;
16             else
17                 if $F(s') = e^*$ then
18                     $Cand \leftarrow Cand \cup \{\langle p, q \rangle\}$;
19                 end
20             end
21         end
22     end
23     $\langle p^*, q^* \rangle \leftarrow$ select$(Cand)$;
24     $s \leftarrow$ AddOnePoint$(s, p^*, q^*)$;
25     $S \leftarrow S \setminus \{p^*\}$;
26     $R \leftarrow R \cup \{p^*\}$;
27 end
28 return $s$;
```

Algorithm 2: GenerateInitialSolution$((s_1, t_1), \ldots, (s_K, t_K), \mathcal{C}, F)$

Algorithm 2 depicts the method for generating an initial solution. The initial solution is generated in a greedy constructive manner. The algorithm receives a set of starting and terminating points (s_i, t_i) of K routes ($\forall i = 1, \ldots, K$), a set $\mathcal{C}$ of client points, a control function F that measures the quality of solutions, and returns a set of K routes visiting all clients $\mathcal{C}$. The algorithm initializes K routes without any client points (lines 2–4) and iteratively inserts greedily a selected client point to one of the K routes. *Cand* records the most potential candidates, each candidate is represented by a pair of two points $\langle p, q \rangle$ (p will be inserted right after point q in the current solution). Lines 10–11 scan all candidates $\langle p, q\rangle$ in which S is the set of client points having not been in the solution and R is the set of points in the solution after which we can insert other client points. Line 12 computes a new solution s' by inserting p right after q in the current solution s. If the quality of s' is better than the best evaluation e^*, then all candidates in *Cand* will be replaced by the new candidate $\langle p, q\rangle$ (lines 13–15). Otherwise, if the quality of s' is equal to the best evaluation e^*, then this candidate will be added into *Cand* (lines 17–19). Lines 23–26 select randomly a candidate from *Cand* for the insertion and update S, R.

Algorithm 3 depicts a procedure that explores a given neighborhood. The procedure receives a neighborhood N and a set S of potential solutions which have been already found so far (i.e., by exploring previous neighborhoods), and returns a new set of best solutions and their evaluation. It scans all solutions of the considered neighborhood N and keeps track of the set of best solution with respect to the function F.

Input:

- N: a neighborhood
- S: set of solutions collected by exploring other neighborhoods so far
- e: evaluation of solutions in S

Output: S: set of new solutions and an evaluation e

```
for $s_i \in N$ do
    if $F(s_i) < e$ then
        $S \leftarrow \{s_i\}$;
        $e \leftarrow F(s_i)$;
    else
        if $F(s_i) = e$ then
            $S \leftarrow S \cup \{s_i\}$
        end
    end
end
return $\langle S, e\rangle$;
```

Algorithm 3: Explore(N, S, e)

3. Experiments

In this section, we conduct two experiments. The first experiment is to compare our proposed algorithm and the local search algorithm of [33], the most recently algorithm for the min-max capacitated vehicle routing problem. The algorithm in [33] employed only two neighborhood structures: one-point move and cross-exchange move and did not use the combined control function. We denote $N2$ this algorithm. Our proposed algorithm exploits multiple neighborhoods and uses a combined control function. We denote *NMCF* our proposed algorithm. All the algorithms have been implemented using the CBLSVR library of [33], a constraint-based local search library for general vehicle routing problems. In the second experiment, we compare the performance of the *NMCF* algorithm and the algorithm proposed by Gold et al. in [32] on the instances described in that paper. In this comparison, we did not obtain exactly the same setting of instances that produced the results presented in [32]. Hence, we re-implemented the algorithm of Golden et al. [32] in Java programming language.

Instances	*N*2				*NMCF*				ρ
	min	max	avg	avg_t (s.)	min	max	avg	avg_t (s.)	
E-n7-k2.vrp	54.00	54.00	54.00	0.00	54.00	54.00	54.00	0.00	0.00
E-n13-k4.vrp	74.00	74.00	74.00	0.32	74.00	74.00	74.00	1.44	0.00
E-n22-k4.vrp	110.00	110.00	110.00	2.03	110.00	110.00	110.00	3.62	0.00
E-n23-k3.vrp	**242.00**	244.00	243.90	7.11	243.00	244.00	**243.45**	27.35	0.18
E-n30-k4.vrp	164.00	164.00	164.00	2.97	164.00	164.00	164.00	2.17	0.00
E-n30-k3.vrp	192.00	206.00	201.30	10.42	**191.00**	195.00	**192.45**	10.85	4.40
E-n31-k7.vrp	75.00	83.00	80.10	20.66	**71.00**	80.00	**74.00**	47.03	7.62
E-n33-k4.vrp	244.00	245.00	244.50	19.99	244.00	244.00	**244.00**	11.07	0.20
E-n51-k5.vrp	113.00	121.00	117.35	78.74	**112.00**	115.00	**113.50**	67.07	3.28
E-n76-k14.vrp	97.00	112.00	106.00	100.05	**90.00**	99.00	**93.60**	27.22	11.70
E-n76-k8.vrp	108.00	129.00	117.70	86.16	**99.00**	102.00	**100.05**	94.57	15.00
E-n76-k15.vrp	90.00	96.00	93.65	129.15	**88.00**	90.00	**88.95**	61.48	5.02
E-n76-k10.vrp	103.00	122.00	114.75	92.96	**95.00**	106.00	**99.55**	31.69	13.25
E-n76-k7.vrp	107.00	129.00	118.20	118.01	**105.00**	110.00	**107.35**	120.49	9.18
E-n101-k8.vrp	127.00	151.00	135.15	157.97	**112.00**	121.00	**117.05**	69.01	13.39
E-n101-k14.vrp	105.00	116.00	111.40	140.71	**100.00**	104.00	**101.50**	68.04	8.89

Table 1: Comparison between *N*2 and *NMCF* on Christophides instances

3.1. Instances and settings

The instances were taken from http://neo.lcc.uma.es/vrp/vrp-instances/capacitated-vrp-instances/ with two data sets. The first data set is from Christofides and Eilon which consists 15 instances. The number of clients in this data set varies from 13 to 101. The second data set is from Golden, Wasil, Kelly and Chao which consists of 20 large-scale instances. The number of clients in this data set varies from 200 to 480.

The experiments were conducted on the machine Intel(R) Core(TM)i7-4790 CPU 3.60GHz with 16GB RAM. Each algorithm was executed 20 times for each instance with the time limit of 5 minutes.

3.2. Experimental results

The first experimental results are shown in Tables 1, 2, 3, 4. The structure of these tables are identical. Each table presents the results of two algorithms (the first algorithm is on the left and the second algorithm is on the right) For example, in Tables 1, 2, columns 2–5 present the minimum, maximum, average of the objective, and the average time to find best solution value among 20 executions of algorithm *N*2. Columns 6–9 presents the same information of algorithm *NMCF*. The last column of each table presents ρ, the percentage of improvement of the second algorithm compared to the first algorithm. More precisely, $\rho = \frac{f_1 - f_2}{f_1} \times 100$ in which f_1 and f_2 are respectively the average objective values of the first and the second algorithms in the table.

Experimental results show that in most of the cases, our proposed algorithm gives better results than *N*2 in term of minimum, maximum, and average objective value among 20 executions for each instance. In the first data set (Christophides), among 320 executions, our proposed algorithm *NMCF* finds better solutions than the algorithm *N2* in 200 executions, while the algorithm *N2* finds better solutions in only 3 executions. In the second data set (Kelly), among 400 executions, our proposed algorithm *NMCF* finds better solution than *N2* in 360 executions, while the algorithm *N2* finds better solutions in only 20 executions.

To evaluate the efficiency of using combined control function, we compare our proposed algorithm

Instances	*N*2				*NMCF*				ρ
	min	max	avg	avg_t (s.)	min	max	avg	avg_t (s.)	
kelly01.txt	640.89	689.71	653.59	290.74	**600.40**	619.66	**610.41**	235.60	6.61
kelly02.txt	940.04	1226.85	1006.36	314.57	**906.66**	952.29	**932.30**	266.47	7.36
kelly03.txt	1239.52	1468.87	1301.62	319.30	**1180.32**	1318.84	**1236.44**	286.28	5.01
kelly04.txt	1348.06	1668.09	1429.13	358.81	**1301.00**	1471.14	**1363.11**	289.77	4.62
kelly05.txt	1442.22	1706.96	1545.20	274.69	**1297.90**	1431.05	**1380.09**	151.72	10.69
kelly06.txt	1176.22	1308.42	1234.17	302.15	**1130.31**	1179.30	**1147.70**	277.44	7.01
kelly07.txt	1288.99	1455.00	1334.43	319.11	**1206.89**	1291.04	**1259.42**	275.13	5.62
kelly08.txt	1232.39	1463.07	1287.31	329.25	**1183.47**	1292.61	**1227.33**	272.61	4.66
kelly09.txt	60.66	76.85	63.68	269.75	60.00	60.62	**60.07**	186.05	5.67
kelly10.txt	69.63	82.63	74.58	305.01	**68.00**	69.16	**68.38**	244.31	8.31
kelly11.txt	81.00	102.87	87.24	303.64	**76.00**	77.60	**76.23**	260.75	12.63
kelly12.txt	97.13	112.36	104.37	318.97	**88.67**	99.34	**94.70**	274.65	9.27
kelly13.txt	43.40	53.30	46.20	187.26	**43.07**	44.61	**43.39**	155.64	6.08
kelly14.txt	51.06	74.80	55.60	273.18	**49.82**	51.98	**50.93**	281.93	8.41
kelly15.txt	57.98	65.67	62.20	276.39	**56.65**	60.84	**58.02**	290.87	6.71
kelly16.txt	67.87	78.92	70.14	300.09	**63.30**	72.00	**66.68**	290.42	4.93
kelly17.txt	41.41	65.67	46.93	248.71	**39.96**	42.26	**40.59**	250.15	13.51
kelly18.txt	51.24	92.15	64.98	212.32	**49.37**	51.39	**50.08**	259.05	22.93
kelly19.txt	65.76	113.89	71.42	282.77	**61.89**	65.70	**63.63**	288.53	10.91
kelly20.txt	77.15	96.36	81.78	187.64	**76.40**	79.66	**77.50**	288.20	5.24

Table 2: Comparison between *N*2 and *NMCF* on Kelly instances

NMCF that uses a combined control function and the version that does not use combined control function (denoted by *NM*). The comparison is presented in Tables 3 and 4. The tables show that in most of the cases, the *NMCF* algorithm finds better than *NM*.

Tables 5 and 6 compares the performance of our proposed algorithm *NMCF* and the algorithm proposed by Golden et al. in [32] (denoted by Golden[32]). We observe that in term of average objective values, the *NMCF* algorithm finds better result than those found by Golden[32] in most of the instances except that last two Fisher instances. However, the improvement of Golden[32] compared to the *NMCF* algorithm is not significant.

Figures 2a and 2b present the behaviour of the two algorithms in an execution example. They plot the value of the objective function and the best objective function found by algorithms *N2* and *NMCF* over iterations on the instance E-n101-k14.vrp. We can see that the algorithm *NMCF* converges faster than the algorithm *N2*.

4. Conclusion

We considered in this paper the min-max capacitated vehicle routing problem and proposed a local search algorithm for solving it. The proposed local search algorithm exploits most of the neighborhood structures in the literature for vehicle routing problems. Experimental results on different data sets show that our proposed algorithm gives better results than the most recently algorithm. We also show the advantage of exploiting a combined control function during the search instead of using the objective function itself. Our future works focus on other metaheuristics search for solving the min-max capacitated vehicle routing problem, especially, we analyze and explore different neighborhoods in a dynamic way including removing ineffective neighborhoods.

Instances	*NM*				*NMCF*				ρ
	min	max	avg	avg_t (s.)	min	max	avg	avg_t (s.)	
E-n7-k2.vrp	54.00	54.00	54.00	0.00	54.00	54.00	54.00	0.00	0.00
E-n13-k4.vrp	74.00	74.00	74.00	0.31	74.00	74.00	74.00	1.44	0.00
E-n22-k4.vrp	110.00	110.00	110.00	4.98	110.00	110.00	110.00	3.62	0.00
E-n23-k3.vrp	**242.00**	243.00	**242.10**	20.50	243.00	244.00	243.45	27.35	-0.56
E-n30-k4.vrp	164.00	164.00	164.00	5.55	164.00	164.00	164.00	2.17	0.00
E-n30-k3.vrp	191.00	197.00	192.95	53.72	191.00	195.00	**192.45**	10.85	0.26
E-n31-k7.vrp	**70.00**	82.00	77.55	81.80	71.00	80.00	**74.00**	47.03	4.58
E-n33-k4.vrp	244.00	245.00	244.15	85.87	244.00	244.00	**244.00**	11.07	0.06
E-n51-k5.vrp	114.00	121.00	118.70	119.50	**112.00**	115.00	**113.50**	67.07	4.38
E-n76-k14.vrp	103.00	118.00	111.10	146.64	**90.00**	99.00	**93.60**	27.22	15.75
E-n76-k8.vrp	110.00	126.00	118.85	140.44	**99.00**	102.00	**100.05**	94.57	15.82
E-n76-k15.vrp	91.00	98.00	94.45	155.69	**88.00**	90.00	**88.95**	61.48	5.82
E-n76-k10.vrp	106.00	125.00	117.40	149.21	**95.00**	106.00	**99.55**	31.69	15.20
E-n76-k7.vrp	113.00	127.00	119.35	115.85	**105.00**	110.00	**107.35**	120.49	10.05
E-n101-k8.vrp	130.00	156.00	141.65	140.07	**112.00**	121.00	**117.05**	69.01	17.37
E-n101-k14.vrp	104.00	119.00	113.25	161.05	**100.00**	104.00	**101.50**	68.04	10.38

Table 3: Comparison between *NM* and *NMCF* on Christophides instances

Instances	*NM*				*NMCF*				ρ
	min	max	avg	avg_t (s.)	min	max	avg	avg_t (s.)	
kelly01.txt	612.93	656.07	630.63	177.01	**600.40**	619.66	**610.41**	235.60	3.21
kelly02.txt	**899.67**	1041.15	949.12	265.01	906.66	952.29	**932.30**	266.47	1.77
kelly03.txt	1192.79	1317.13	1250.64	310.34	**1180.32**	1318.84	**1236.44**	286.28	1.14
kelly04.txt	1304.34	1531.80	1388.07	354.59	**1301.00**	1471.14	**1363.11**	289.77	1.80
kelly05.txt	1394.89	1592.97	1464.86	173.50	**1297.90**	1431.05	**1380.09**	151.72	5.79
kelly06.txt	1156.07	1273.24	1217.88	163.16	**1130.31**	1179.30	**1147.70**	277.44	5.76
kelly07.txt	1241.57	1304.21	1277.28	281.88	**1206.89**	1291.04	**1259.42**	275.13	1.40
kelly08.txt	1193.26	1331.25	**1227.02**	337.62	**1183.47**	1292.61	1227.33	272.61	-0.03
kelly09.txt	60.65	65.22	62.04	168.89	**60.00**	60.62	**60.07**	186.05	3.18
kelly10.txt	69.09	72.65	70.49	247.25	**68.00**	69.16	**68.38**	244.31	3.00
kelly11.txt	77.11	80.02	78.70	310.37	**76.00**	77.60	**76.23**	260.75	3.14
kelly12.txt	88.99	101.09	**94.43**	320.92	**88.67**	99.34	94.70	274.65	-0.28
kelly13.txt	43.08	51.75	47.48	102.21	**43.07**	44.61	**43.39**	155.64	8.61
kelly14.txt	50.04	57.68	53.02	211.42	**49.82**	51.98	**50.93**	281.93	3.94
kelly15.txt	56.17	65.18	59.33	238.13	**56.65**	60.84	**58.02**	290.87	2.20
kelly16.txt	63.76	71.94	66.76	255.37	**63.30**	72.00	**66.68**	290.42	0.12
kelly17.txt	41.48	65.67	45.64	126.34	**39.96**	42.26	**40.59**	250.15	11.06
kelly18.txt	50.71	92.15	69.20	123.83	**49.37**	51.39	**50.08**	259.05	27.63
kelly19.txt	62.69	84.50	64.63	238.43	**61.89**	65.70	**63.63**	288.53	1.54
kelly20.txt	**76.29**	96.36	79.70	121.05	76.40	79.66	**77.50**	288.20	2.76

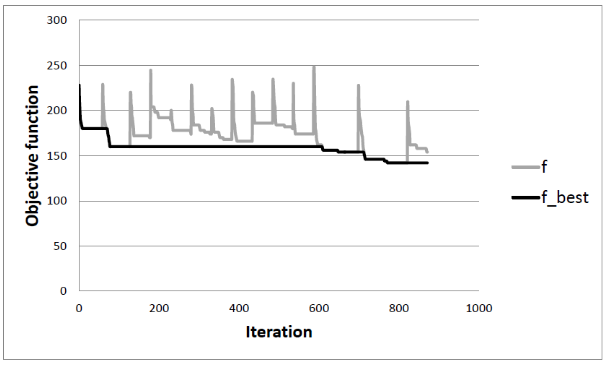
Table 4: Comparison between *NM* and *NMCF* on Kelly instances

Instances	Golden[32]				*NMCF*				ρ
	min	max	avg	avg_t (s.)	min	max	avg	avg_t (s.)	
CMT-1-m5.txt	194	225	212.15	150.1815	113	116	115.05	67.69775	45.77
CMT-1-m6.txt	155	186	170.69	151.461	103	103	103	2.65145	39.66
CMT-1-m7.txt	149	161	151.6	146.5	95	95	95	1.1177	37.34
CMT-2-m10.txt	125	128	125.34	135.371	95	109	100.95	12.32805	19.46
CMT-2-m11.txt	115	115	115	129.19	89	94	90.95	31.00855	20.91
CMT-2-m12.txt	113	113	113	125.191	88	93	89.95	17.3704	20.40
CMT-3-m8.txt	204	204	204	186.6735	116	124	119.75	19.8403	41.30
CMT-3-m9.txt	171	171	171	187.0545	107	115	111.1	34.3317	35.03
CMT-3-m10.txt	145	145	145	187.587	104	114	107.25	12.8914	26.03
CMT-4-m12.txt	168	168	168	396.2255	103	114	110.05	62.4103	34.49
CMT-4-m13.txt	139	139	139	390.7615	100	116	106.2	46.9419	23.60
CMT-5-m16.txt	130	130	130	732.64	108	145	121	189.4161	6.92
CMT-5-m17.txt	119	119	119	727.049	101	111	104.2	114.2479	12.44
CMT-11-m7.txt	229	229	229	242.2955	204	227	209.25	72.92035	8.62
CMT-11-m8.txt	219	219	219	242.2375	198	205	199.85	60.28815	8.74
CMT-12-m10.txt	127	127	127	187.2265	121	126	121.75	29.66425	4.13
CMT-12-m11.txt	124	124	124	186.9405	117	121	118.3	35.5866	4.60

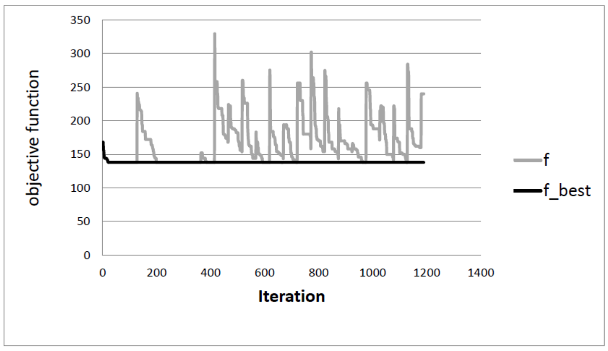
Table 5: Comparison between Golden[32] and *NMCF* on Christophides, Mingozzi and Toth instances

Instances	Golden[32]				*NMCF*				ρ
	min	max	avg	avg_t (s.)	min	max	avg	avg_t (s.)	
F-n72-m4.vrp	98	104	99	118.756	66	68	67.75	59.95555	31.57
F-n72-m5.vrp	85	91	85.46	117.698	62	64	63.2	12.61015	26.05
F-n72-m6.vrp	67	67	67	118.4605	56	58	56.15	27.10635	16.19
F-n135-m7.vrp	293	293	293	302.084	299	309	300.8	74.22805	-2.66
F-n135-m8.vrp	292	292	292	301.8725	295	299	296.65	49.4305	-1.59

Table 6: Comparison between Golden[32] and *NMCF* on Fisher instances



a. Algorithm *N2*



b. Algorithm *NMCF*

Figure 2: Evolution of the objective function found by *N2* and *NMCF* over itreration on the instance E-n101-k14.vrp

ACKNOWLEDGMENT

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number FWO.102.2013.04.

REFERENCES

[1] G. Laporte, "The vehicle routing problem: An overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, no. 3, pp. 345 – 358, 1992. [Online]. Available: http://www.sciencedirect.com/science/article/pii/037722179290192C

[2] R. S. W.-E. A. Golden, Bruce L., *The Vehicle Routing Problem: Latest Advances and New Challenges*, ser. 43. The address: Springer, 2008.

[3] T. Ralphs, L. Kopman, W. Pulleyblank, and L. Trotter, "On the capacitated vehicle routing problem," *Mathematical Programming*, vol. 94, no. 2, pp. 343–359, 2003. [Online]. Available: http://dx.doi.org/10.1007/s10107-002-0323-0

[4] D. Applegate, W. Cook, S. Dash, and A. Rohe, "Solution of a min-max vehicle routing problem," *INFORMS J. on Computing*, vol. 14, no. 2, pp. 132–143, Apr. 2002. [Online]. Available: http://dx.doi.org/10.1287/ijoc.14.2.132.118

[5] M. Desrochers, J. Desrosiers, and M. Solomon, "A new optimization algorithm for the vehicle routing problem with time windows," *Operations Research*, vol. 40, no. 2, pp. 342–354, 1992. [Online]. Available: http://dx.doi.org/10.1287/opre.40.2.342

[6] E. M. Arkin, R. Hassin, and A. Levin, "Approximations for minimum and min-max vehicle routing problems," *Journal of Algorithms*, vol. 59, no. 1, pp. 1 – 18, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0196677405000258

[7] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management Science*, vol. 6, no. 1, pp. 80–91, 1959. [Online]. Available: http://dx.doi.org/10.1287/mnsc.6.1.80

[8] B. Eksioglu, A. V. Vural, and A. Reisman, "The vehicle routing problem: A taxonomic review," *Computers & Industrial Engineering*, vol. 57, no. 4, pp. 1472 – 1483, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360835209001405

[9] R. W. Hall. Vehicle routing software survey. [Online]. Available: RetrievedAugust16,2013, fromhttp://www.orms-today.org/surveys/Vehicle_Routing/vrss.html

[10] L. G. P. J.-Y. Gendreau, M., "Metaheuristics for the capacitated vrp," in *The Vehicle Routing Problem*, V. D. Toth, P., Ed. SIAM Monographs on Discrete Mathematics and Applications, 2002, vol. 9.

[11] Y. Marinakis, "Multiple phase neighborhood search-grasp for the capacitated vehicle routing problem," *Expert Systems with Applications*, vol. 39, no. 8, pp. 6807 – 6815, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417412000176

[12] R. Baldacci, E. Hadjiconstantinou, and A. Mingozzi, "An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation," *Operations Research*, vol. 52, no. 5, pp. 723–738, 2004. [Online]. Available: http://dx.doi.org/10.1287/opre.1040.0111

[13] R. Fukasawa, H. Longo, J. Lysgaard, M. P. d. Aragão, M. Reis, E. Uchoa, and R. F. Werneck, "Robust branch-and-cut-and-price for the capacitated vehicle routing problem," *Mathematical Programming*, vol. 106, no. 3, pp. 491–511, 2006. [Online]. Available: http://dx.doi.org/10.1007/s10107-005-0644-x

[14] R. Baldacci, A. Mingozzi, and R. Roberti, "Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints," *European Journal of Operational Research*, vol. 218, no. 1, pp. 1 – 6, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221711006692

[15] J. Jin, T. G. Crainic, and A. Lkketangen, "A parallel multi-neighborhood cooperative tabu search for capacitated vehicle routing problems," *European Journal of Operational Research*, vol. 222, no. 3, pp. 441 – 451, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S037722171200375X

[16] W. Szeto, Y. Wu, and S. C. Ho, "An artificial bee colony algorithm for the capacitated vehicle routing problem," *European Journal of Operational Research*, vol. 215, no. 1, pp. 126 – 135, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221711005121

[17] R. Liu, Z. Jiang, R. Y. Fung, F. Chen, and X. Liu, "Two-phase heuristic algorithms for full truckloads multi-depot capacitated vehicle routing problem in carrier collaboration," *Computers & Operations Research*, vol. 37, no. 5, pp. 950 – 959, 2010, disruption Management. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0305054809001968

[18] S. Kumar and R. Panneerselvam, "A survey on the vehicle routing problem and its variants," *Intelligent Information Management*, vol. 4, no. 3, pp. 66–74, 2012.

[19] O. Bräysy and M. Gendreau, "Vehicle routing problem with time windows, part i: Route construction and local search algorithms," *Transportation Science*, vol. 39, no. 1, pp. 104–118, Feb. 2005. [Online]. Available: http://dx.doi.org/10.1287/trsc.1030.0056

[20] M. G. F. G. E. Taillard, P. Badeau and J.-Y. Potvin, "A tabu search heuristic for the vehicle routing problem with soft time windows," *Transportation Science*, vol. 31, no. 2, pp. 170–186, Feb. 1997.

[21] W.-C. Chiang and R. A. Russell, "Simulated annealing metaheuristics for the vehicle routing problem with time windows," *Annals of Operations Research*, vol. 63, no. 1, pp. 3–27, 1996. [Online]. Available: http://dx.doi.org/10.1007/BF02601637

[22] J.-Y. Potvin and S. Bengio, "The vehicle routing problem with time windows part ii: Genetic search," *INFORMS Journal on Computing*, vol. 8, no. 2, pp. 165–172, 1996. [Online]. Available: http://dx.doi.org/10.1287/ijoc.8.2.165

[23] G. Kontoravdis and J. F. Bard, "A grasp for the vehicle routing problem with time windows," *ORSA Journal on Computing*, vol. 7, no. 1, pp. 10–23, 1995. [Online]. Available: http://dx.doi.org/10.1287/ijoc.7.1.10

[24] F.-H. Liu and S.-Y. Shen, "The fleet size and mix vehicle routing problem with time windows," *Journal of the Operational Research Society*, vol. 50, pp. 721–732, 1999.

[25] L. M. Gambardella, ric Taillard, and G. Agazzi, "Macs-vrptw: A multiple colony system for vehicle routing problems with time windows," in *New Ideas in Optimization.* McGraw-Hill, 1999, pp. 63–76.

[26] G. Gutirrez-Jarpa, G. Desaulniers, G. Laporte, and V. Marianov, "A branch-and-price algorithm for the vehicle routing problem with deliveries, selective pickups and time windows," *European Journal of Operational Research*, vol. 206, no. 2, pp. 341 – 349, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221710001700

[27] M. Gendreau, F. Guertin, J.-Y. Potvin, and R. Sguin, "Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries," *Transportation Research Part C: Emerging Technologies*, vol. 14, no. 3, pp. 157 – 174, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0968090X06000349

[28] C. R. Delgado Serna and J. Pacheco Bonrostro, *Minmax Vehicle Routing Problems: Application to School Transport in the Province of Burgos.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 297–317. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-56423-9_17

[29] P. M. Frana, M. Gendreau, G. Laporte, and F. M. Mller, "The m-traveling salesman problem with minmax objective," *Transportation Science*, vol. 29, no. 3, pp. 267–275, 1995. [Online]. Available: http://dx.doi.org/10.1287/trsc.29.3.267

[30] S. v. E. P. J. Hemel, T. The manhattan project. [Online]. Available: http://www.win.tue.nl/whizzkids/1996/tsp.html

[31] C. Ren, "Solving min-max vehicle routing problem," *JOURNAL OF SOFTWARE*, vol. 6, no. 9, 2011.

[32] B. L. Golden, G. Laporte, and E. D. Taillard, "An adaptive memory heuristic for a class of vehicle routing problems with minmax objective," *Comput. Oper. Res.*, vol. 24, no. 5, pp. 445–452, May 1997. [Online]. Available: http://dx.doi.org/10.1016/S0305-0548(96)00065-2

[33] P. Q. Dung, L. K. Thu, N. T. Hoang, P. V. Dinh, and B. Q. Trung, "A constraint-based local search for offline and online general vehicle routing," *International Journal on Artificial Intelligence Tools. To appear*, 2016.

[34] C. Groer, B. Golden, and E. Wasil, "A library of local search heuristics for the vehicle routing problem," *Math. Prog. Comp.*, vol. 2, no. 2, pp. 79 – 101, 2010.