Metaheuristic for the Capacitated Multiple Traveling Repairman Problem

Khanh-Phuong Nguyen[1]; Ha-Bang Ban[2]
School of Information and Communication Technology
Hanoi University of Science and Technology, Hanoi, Vietnam
Corresponding authors: Ha-Bang Ban

***Abstract*—The Capacitated Multiple Traveling Repairmen Problem (CmTRP) is an extension of the Multiple Traveling Repairmen Problem (mTRP). In the CmTRP, the number of vehicles is dispatched to serve a set of customers, while each vehicle's capacity is limited by a predefined-value as well as each customer is visited exactly once. The goal is to find a tour that minimizes the sum of waiting times. The problem is NP-hard because it is harder than the mTRP. Even finding a feasible solution is also NP-hard problem. To solve medium and large size instances, a metaheuristic algorithm is proposed. The first phase constructs a feasible solution by combining between the Nearest Neighborhood Search (NNS) and Variable Neighborhood Search (VNS), while the optimization phase develops the feasible solution by the General Variable Neighborhood Search (GVNS). The combination maintains the balance between intensification and diversification to escape local optima. The proposed algorithm is implemented on benchmark instances from the literature. The results indicate that the developed algorithm obtains good feasible solutions in a short time, even for the cases with up to 200 vertices.**

***Keywords*—CmTRP; NNS; VNS; GVNS**

I. Introduction

A. Motivation and Definition

A particular variant of the CmTRP is the Multiple Traveling Repairmen Problem (mTRP) that considers multiple vehicles or travelers to find a tour minimizing the waiting time of all customers [1], [7], [9]. Applications of the mTRP can be found in [1], [7], [9]. The mTRP is based on an assumption that there is no limit to the capacity of each vehicle. That means vehicles can carry as many goods as they want. However, real situations imply that it does not always hold because vehicles have strict regulations on capacity. In this work, the mTRP is involved the capacity constraint. In the CmTRP, the maximum capacity of each vehicle does not exceed a predefined capacity (*Q*). In the CmTRP, there are *k* vehicles at a main depot *s*, and *n* customers. The goal is to find a tour such that each vertex is visited exactly once, while the capacity constraint is satisfied, and the total waiting time of overall customers is minimized.

In this paper, we formulate the CmTRP as followings: We consider a complete graph K_n that has the vertex set $V = \{1, 2, ..., n\}$ and a symmetric distance matrix $C = \{c(i,j) \mid i,j = 1,2,...,n\}$ ($c(i,j)$ is the traveling cost between vertex i and vertex j). Each vertex i has the demand d_i. Let $R = (1,2,...,k)$ be the number of k vehicles which begin at a main depot v_1. Let Q denote the capacity of a vehicle. Assume that a tour $T = (R_1, ..., R_l, ..., R_k)$ includes a set of routes from k vehicles. $R_l = (v_1, ..., v_h, ..., v_m)$ $(1 < m \leq n)$ is a route of vehicle l. Let $P(v_1, v_h)$, $l(P(v_1, v_h))$ be respectively the path from vertex v_1 to vertex v_h on R_l, and its length. The waiting time of v_h $(1 < h \leq m)$ on R_l is the cost of the path from v_1 to v_h:

$$l(P(v_1, v_h)) = \sum_{i=1}^{h-1} c(v_i, v_{i+1}).$$

Let $W(R_l)$ be the sum of waiting times of all vertices. The capacity of this route must satisfy the following constraint:

$$W(R_l) = \sum_{h=2}^{m} l(P(v_1, v_h)).$$

$$D(R_l) = \sum_{i=1}^{m} d_i \leq Q.$$

The objective function of the problem is:

$$W(T) = \sum_{l=1}^{k} W(R_l).$$

The CmTRP asks for a tour, which starts at a depot v_1, visits each vertex once exactly with the waiting time of all vertices being minimized.

B. Related Works

In the general case, the CmTRP has, to the best of our knowledge, previously not been studied much, even though it is an extension of the mTRP case. However, some variants of the CmTRP are introduced in numerous works in the literature as follows:

- The mTRP is a particular case when the capacity constraint does not involve. Numerous works for mTRP can be found in [1], [7], [9]. In the metaheuristic approach, several algorithms [1], [7], [9] produce good solutions fast for instances with up to 200 vertices.
- The mTRP with distance constraint (mTRPD) is a particular case since the route length or maximum duration of each vehicle cannot exceed a predetermined limit. Metaheuristic algorithms in [3], [12] can solve the problem well for instances with up to 200 vertices.
- The mTRP with Profits (mTRPP) aims is to find a travel plan for server maximizing the total revenue. In this problem, not all customers need to be visited. Metaheuristic algorithm in [11] solves the problem well with up to 200 vertices.

- The Traveling Repairman Problem is a special case where there is only a repairman. Numerous metaheuristic algorithms [2], [4], [5], [17] for the problem have proposed in the literature.

The above algorithms are the state-of-the-art algorithms for some variants of the CmTRP case. However, they do not include the capacity constraint, and their corresponding algorithms cannot be adapted to the CmTRP. That means that we cannot use the above algorithms to solve the CmTRP.

C. Our Methodology

Like other NP-hard problems, the CmTRP can be solved by exact or heuristic methods. Exact algorithms obtain the optimal solution but they often take exponential time in the worst case. Heuristic approaches are divided into the classical heuristic and metaheuristic approach. The classical heuristic approach finds one solution quickly, but this solution may have a large disparity in comparison with the best solution. The metaheuristic approach, on the other hand, obtains near-optimal or even global optimal solutions. Therefore, metaheuristic is usually used to reach optimal solutions for the problem with large sizes.

The CmTRP is also NP-hard because it is a generalization case of the mTRP. In many constrained optimization problems like the CmTRP, even building a feasible solution to the problem is also NP-hard problem. It indicates that obtaining a good feasible solution is a challenge. A good metaheuristic needs to ensure the balance between diversification and intensification. In [14], H. Mladenovic et al. show that the VNS generates local optima that are close to the global optimum, in a more straightforward manner than the other metaheuristics. However, the VNS only implements the intensification well. In this work, we developed a metaheuristic consisting of the constructive and optimization phase. The first phase constructs a feasible solution by combining between the Nearest Neighborhood Search (NNS) and Variable Neighborhood Search (VNS), while the optimization phase develops the feasible solution by the General Variable Neighborhood Search (GVNS). The metaheuristic uses the shaking technique to maintain diversification, while the VNS, and GVNS implement intensification. Extensive numerical experiments on benchmark instances show that the proposed algorithm reaches good feasible solutions at a reasonable amount of time, even for the instances with up to 200 vertices.

The rest of this paper is organized as follows. Section 2 introduces our algorithm. Computational evaluations are presented in Section 3, and Sections 4 and 5 discusses and concludes the work, respectively.

II. The Proposed Algorithm

A. Several Variants of VNS

We describe some variants of VNS [14] such as the original VNS, VND, and GVNS, shaking technique [13], respectively.

- The Variable Neighborhood Search (VNS) algorithm is introduced by Mladenovic et al. [14]. It executes neighborhood procedures alternately, and shaking technique to escape from the local optima. At each iteration, the best neighboring solution is chosen from neighboring solutions that are generated from a neighborhood procedure. If it is better than the current best one, the procedure is repeated. Otherwise, the search goes to the next neighborhood procedure.
- The Variable Neighborhood Descent (VND) algorithm, which is a VNS variant, is proposed by Mladenovic et al. [14]. In the VND, a change of neighborhoods is performed in a deterministic way. Assume that an initial solution is given. Local search procedures in their descent phase are used to generate neighborhoods. The final solution is the best solution in all neighborhoods. The difference between the VNS and VND is that the VNS uses Shaking.
- The General Variable Neighborhood Search (GVNS) algorithm [14] is a variant of the VNS. It includes an initial feasible solution, and a shaking procedure followed by the VND local search. The GVNS is a VNS variant where the VND is used as the improvement procedure.

B. Neighborhoods

We use seven neighborhoods in the literature to explore the search space of the problem. Let $N_k(k = 1, ..., k_m)$ be a finite set of pre-selected neighborhood structures. We describe more details about seven neighborhoods:
For Inter-route: It is used to optimize on each route. We then describe five neighborhoods' structure in turn. Assume that, R, and m are a route and its size, respectively.

- **Remove-insert** places each vertex in the route at the end of it. Obviously, the complexity time of $N_1(R)$ is $O(m)$.
- **Swap adjacent** tries to swap each pair of adjacent vertices in the route. The complexity time of $N_2(R)$ is $O(m)$.
- **Swap neighborhood** attempts to swap the positions of each pair of vertices in the route. The complexity time of $N_3(R)$ is $O(m^2)$.
- **3-opt neighborhood** attempts to reallocate three vertices to another position of the route. The complexity time of $N_4(R)$ is $O(m^3)$.
- **4-opt neighborhood** attempts to involve deleting four edges and reconnecting the four sub-tours without changing the orientation of them. The complexity time of $N_5(R)$ is $O(m^4)$.

For intra-route: Intra-route is used to swap vertices between two different routes or remove vertices from a route and then insert them to another.Let R_l, R_h, ml, and mh be two different routes and their sizes in T, respectively.

- **swap-2-routes** tries to exchange two vertices belonging to different routes R_l and R_h. The complexity time of $N_6(R)$ is $O(ml \times mh)$
- **insert-2-routes** removes one vertex R_l and inserts it at the best possible position in R_h. The complexity time of $N_7(R)$ is $O(ml \times mh)$.

Algorithm 1 The proposed algorithm

Input: $T, nloop, t_{max}$ are an initial solution, the number of neighborhoods, and the number of iterations, and the maximum running time, respectively.
Output: the best solution T^*.

repeat
  {Construction phase}
  $T \leftarrow$ **Construction**(v_1, V);
  {Improvement phase}
  $Lvel = 1$;
  T = VND(T);
  while $(Lvel < nloop)$ **do**
    $T^{'}$ = Perturbation(T,*Lvel*);
    {implement VND}
    $T^{'}$ = VND($T^{'}$);
    if $(W(T^{'}) < W(T)) \parallel (W(T^{'}) < W(T^*))$ **then**
      $T = T^{'}$;
      if $(W(T^{'}) < W(T^*))$ **then**
        $T^* = T^{'}$;
      end if
    end if
    if (T is equal T') **then**
      $Lvel = 1$;
    else
      $Lvel + +$;
    end if
  end while
until $time < t_{max}$
return T^*;

Algorithm 2 VND

Input: T, k_{max} are an initial solution, and the number of neighborhoods, respectively.
Output: the best solution T^*.

$k = 1$;
repeat
  Find the best neighborhood $T^{'}$ of $T \in N_k(T)$; {$T^{'}$ must be feasible solution}
  if $(W(T^{'}) < W(T)) \parallel (W(T^{'}) < W(T^*))$ **then**
    $T = T^{'}$; {centre the search around $T^{'}$ and search again in the first neighborhood}
    if $((W(T^{'}) < W(T^*))$ **then**
      $T^* = T^{'}$;
    end if
    $k = 1$;
  else
    $k = k + 1$; {switch to another neighborhood}
  end if
until $k < k_{max}$;
$T^* = T^{'}$;
return T^*;

The proposed algorithm includes two phases. The construction phase finds a feasible solution, whereas the improvement phase tries to improve it. Algorithm 1 depicts the whole process.

Algorithm 3 Construction

Input: v_1, K_n, k, α are a depot, the graph, the number of vehicles, the length of *NL*, respectively.
Output: An initial solution T.

for $(l = 1; l < k; l + +)$ **do**
  $R_l = R_l \cup v_1$; {All routes start at v_1}
end for
while all vertices are not visited **do**
  $l =$ random(k);{a route randomly is chosen}
  {v_e is the last vertex in R_l}
  Generate *NL* list that includes α nearest vertices to v_e in V;
  Select vertex $v \in \{v_i | v_i \in NL$ and $v_i \notin R_l\}$ randomly;
  $R_l \leftarrow \{v_i\}$
end while
if T is feasible **then**
  return T;
else
  Lvel = 1;
end if
while ((T is infeasible) and $(Lvel \leq l\_max)$) **do**
  T'= Shaking($T, Lvel$);
  T' = VND(T');
  if $W(T') < W(T)$ **then**
    $T \leftarrow T'$;
  end if
  if $W(T') == W(T)$ **then**
    $Lvel \leftarrow 1$;
  else
    $Lvel + +$;
  end if
end while
return T;

Algorithm 4 Perturbation$(T, Lvel)$

Input: $T, Lvel$ are the tour, and the parameter to control the strength of the perturbation procedure, respectively.
Output: a new tour T.

$k = 1$;
while $(k < Lvel)$ **do**
  $T^{'}$ = double-brigde(T);
  $T^{'} \leftarrow$ arg min $N_1(T^{'})$;
  {T^* is the optimal solution}
  if $(W(T^{'}) > (1 - \rho) \times W(T^*))$ **then**
    $T = T^{'}$
  else
    $k + +$;
  end if
end while
return T;

C. Construction

Algorithm 2 shows the constructive procedure. The objective function used in this procedure is the sum of all positive differences between the capacity of all vehicles and the capacity limit Q, that is, $\min \sum_{l=1}^{k} \max(0, D(R_l) - Q)$. The algorithm works until it finds a feasible solution. In the first step, a solution is created by Nearest Neighborhood Search [8]. If the solution is feasible, the construction phase stops

TABLE I. OUR RESULTS FOR THE CMTRP ON E-INSTANCES PROPOSED BY [18]

Instances	*Init.Sol*	*Best.Sol*	*Aver.Sol*	*Improv*	*T*
E30k3	3008.64	2419.1	2419.1	19.59	2
E30k4	2052.97	1731.43	1731.43	15.66	3
E51k5	2948.73	2769.1	2769.1	6.09	7
E76k10	3301.85	3064.68	3064.68	7.18	27
E76k14	2409.68	2261.63	2261.63	6.14	14
E76k15	2401.86	2221.59	2221.59	7.51	17
aver				10.36	11.7

TABLE II. OUR RESULTS FOR THE CMTRP ON P-INSTANCES PROPOSED BY [18]

Instances	*Init.Sol*	*Best.Sol*	*Aver.Sol*	*Improv*	*T*
P40k5	2095.22	1884.55	1884.55	10.05	5
P45k5	2672.9	2479.41	2479.41	7.24	8
P50k7	2262.5	1993.24	1993.24	11.90	7
P50k8	2310.09	2310.09	2310.09	0.00	4
P55k7	2325.41	2175.46	2175.46	6.45	13
P55k10	2095.43	1878.64	1878.64	10.35	6
P60k10	2511.53	2322.96	2322.96	7.51	15
P70k10	3297.38	2940.94	2940.94	10.81	17
P76k4	6509.49	6198.5	6198.5	4.78	28
P76k5	5920.72	5637.31	5637.31	4.79	29
aver				7.39	13.2

TABLE III. OUR RESULTS FOR THE CMTRP ON TAI-INSTANCES PROPOSED BY [18]

Instances	*Init.Sol*	*Best.Sol*	*Aver.Sol*	*Improv*	*T*
tai75a	7038	6286.77	6331.913	10.67	32
tai75b	6112.16	5061.1	5156.406	17.20	30
tai75c	5517.07	5023.56	5085.652	8.95	35
tai75d	6347.94	5856.79	5974.241	7.74	37
tai100a	11391.44	10200.9	10290.74	10.45	72
tai100b	10297.75	9755.82	9801.026	5.26	73
tai100c	6025.94	5743.6	5792.192	4.69	70
tai100d	7544.23	7138.13	7203.689	5.38	68
tai150a	16435.68	15913.87	16005.03	3.17	85
tai150b	14115.32	13627.56	13687.99	3.46	81
tai150c	14797.07	13098.63	13208.41	11.48	82
tai150d	15268.64	13530.54	13530.54	11.38	83
aver				8.32	62.3

and outputs it. On the other hand, a local search iterates until finding a feasible solution or l_max is reached. The solution is shaken to escape from the current local optimal. Next, the VNS is applied to obtain the best solution from neighboring solutions. If it is better than the found best solution, it is set to the current solution. Last, *Lvel* is increased by one if the current solution is not improved, or set to 1, otherwise.

D. Improvement

After the construction, the heuristic tries to improve the feasible solution created by the previous phase. In this phase, the objective function is to minimize $W(T)$.

Local search procedure that is developed by combining the seven neighborhoods generates various neighborhoods. The final solution should be a local minimum with respect to all neighborhoods. The order of neighborhoods is fixed. In a preliminary experiment, the other of the neighborhoods are therefore explored in the following one, from "small" to "large" as it is common, i.e., swap-adjacent, move-up, move-down, remove-insert, swap, 2-opt, 3-opt, 4-opt, swap-2-routes, and insert-2-routes.

The Perturbation mechanism is very important to achieve success. When the mechanism has too small perturbation moves, the search can return to the previously visited solution space. Therefore, the search can get stuck into local optimal. On the other hand, large perturbation moves drive the search to undesirable space. In order to overcome these issues, we propose a new shaking technique based on the original double-bridge technique [13]. The detail is described in Algorithm 3.

The algorithm stops after t_{max} seconds or the best-solution is found (t_{max} is the parameter of the algorithm, and its value is determined from preliminary experiments).

III. EVALUATIONS

The proposed algorithm is run on a Pentium 4 core i7 2.40 GHz processor with 8 GB of RAM. On all experiments, parameters α, l_max, ρ, *nloop* are respectively set to 10, 5, 0.3, and 50. These parameters are chosen through empirical tests and, with them, the algorithm seems to produce good solutions at a reasonable amount of time in comparison with the other parameter values.

We also implement the performance of the whole implementation against the state-of-the-art algorithms. We compare both of the numerical results and computational time on the same instances.

A. Instances

The experiments are performed on a set of benchmark for Capacitated VRP in [12], [18]. As testing the proposed algorithm on overall instances can be computationally too expensive, some selected instances as follows: 1) to eliminate the effects of size, instances with approximately from 50 up to 200 customers are chosen; 2) in order not to bias the results by taking "easy" or "hard" instances, we randomly select them. These are: 1) Christofides et al.: This dataset includes seven instances (CMT6, CMT7, ..., CMT14); 2) Taillard et al.: Ten instances are picked randomly such as tai75a, tai75b, tai75c, tai75d, tai100a, tai100b, tai100c, tai100d, tai150a, tai150b, tai150c, and tai150d; 3) Augerat et al.: Fifteen instances of dataset *P* and *E* are selected; 4) S. Nucamendi-Guillén et al.: 150 instances from 60 to 80 vertices are used in the experiments. The optimal solutions for the instances can be extracted from [16].

B. Results

We define the improvement of our algorithm with respect to *Best.Sol* (*Best.Sol* is the best solution found by the proposed algorithm) in comparison with the initial solution from the construction phase as followings:

$$Improv[\%] = \frac{Best.Sol - Init.Sol}{Init.Sol} \times 100\% \qquad (1)$$

In the tables, *OPT*, *Init.Sol*, *Best.Sol*, *Aver.Sol* and *T* correspond to the optimal solution, initial solution, best solution, average

TABLE IV. THE EVOLUTION OF AVERAGE IMPROVEMENT DURING THE ITERATIONS

instances	1 iteration		10 iterations		20 iterations		30 iterations		40 iterations		50 iterations		100 iterations	
E-instance	6.85	0.2	7.65	1.48	8.45	4.32	9.78	6	10.1	8.17	10.4	11.7	10.4	17.33
P-instance	5.5	0.2	5.8	1.7	6	4.9	6.4	6.8	6.8	9.2	7.4	13.2	7.4	25.9
Tai-instances	6.7	1.1	7.1	7.9	7.4	23.0	7.8	31.9	8	43.5	8.3	62.3	8.3	187.9
aver	6.35	0.50	6.85	3.68	7.28	10.73	7.99	14.91	8.30	20.30	8.70	29.07	8.70	77.04

TABLE V. THE DIFFERENCE BETWEEN THE CMTRP AND MTRP'S OBJECTIVE FUNCTION ON SOME INSTANCES

Instances	mTRP	CmTRP	%diff
E30k3	1871.08	2419.1	29.29
E30k4	1643.30	1731.43	5.36
E51k5	2209.64	2769.1	25.32
E76k10	2310.09	3064.68	32.66
E76k14	2005.40	2261.63	12.78
E76k15	1962.47	2221.59	13.20
P40k5	1537.79	1884.55	22.55
P45k5	1912.31	2479.41	29.66
P50k7	1547.89	1993.24	28.77
aver			22.18

TABLE VI. OUR RESULTS FOR THE MTRP-INSTANCES WITH 60 VERTICES PROPOSED BY LUO

Instances	*OPT*	*Init.Sol*	*Best.Sol*	*Aver.Sol*	*T*
pr1002_60_0	530946.01	660211.35	530946.01	530946.01	2.98
pr1002_60_1	356469.79	455893.41	356469.79	356469.79	2.84
pr1002_60_2	344118.14	467498.53	344118.14	344118.14	2.98
pr1002_60_3	429604.2	579392.35	429604.2	429604.2	2.92
pr1002_60_4	435655.25	540342.11	435655.25	435655.25	2.82
pr1002_60_5	668129.73	779776.11	668129.73	668129.73	2.86
pr1002_60_6	406678.77	495022.53	406678.77	406678.77	2.9
pr1002_60_7	311254.73	414296.52	311254.73	311254.73	2.98
pr1002_60_8	469816.84	591638.26	469816.84	469816.84	3
pr1002_60_9	277336.06	377249.41	277336.06	277336.06	2.84
brd14051_60_0	213420.42	267899.2375	213420.42	213420.42	3
brd14051_60_1	218315.68	312468.7714	218315.68	218315.68	2.98
brd14051_60_2	151666.85	207799.2353	151666.85	151666.85	2.9
brd14051_60_3	172199.83	232433.3597	172199.83	172199.83	2.96
brd14051_60_4	133952.5	167792.608	133952.5	133952.5	2.84
brd14051_60_5	203145.14	290606.4286	203145.14	203145.14	2.88
brd14051_60_6	136233.51	171636.975	136233.51	136233.51	2.98
brd14051_60_7	171879.58	248180.3	171879.58	171879.58	2.96
brd14051_60_8	191580.79	241067.3882	191580.79	191580.79	2.98
brd14051_60_9	128178.58	174326.1925	128178.58	128178.58	2.94
fnl4461_60_0	156260.54	194032.1583	156260.54	156260.54	2.8
fnl4461_60_1	103190.13	131569.4961	103190.13	103190.13	2.96
fnl4461_60_2	109739.93	149525.6149	109739.93	109739.93	2.98
fnl4461_60_3	100792.2	136198.0575	100792.2	100792.2	2.94
fnl4461_60_4	149638.18	185947.0777	149638.18	149638.18	2.96
fnl4461_60_5	158679.44	185251.4379	158679.44	158679.44	2.96
fnl4461_60_6	122266.92	149102.6283	122266.92	122266.92	2.88
fnl4461_60_7	107469.11	142108.8532	107469.11	107469.11	2.94
fnl4461_60_8	100531.72	127280.3749	100531.72	100531.72	2.84
fnl4461_60_9	135829.76	183343.8156	135829.76	135829.76	2.94
d15112_60_0	684939.42	851498.8482	684939.42	684939.42	2.8
d15112_60_1	644759.99	819500.5493	644759.99	644759.99	2.86
d15112_60_2	425069.33	583381.5404	425069.33	425069.33	2.82
d15112_60_3	528177.95	662371.45	528177.95	528177.95	2.82
d15112_60_4	586915.82	736112.95	586915.82	586915.82	2.96
d15112_60_5	422195.61	494729.4263	422195.61	422195.61	2.94
d15112_60_6	518793.6	633637.8578	518793.6	518793.6	2.86
d15112_60_7	616918.44	776732.675	616918.44	616918.44	2.98
d15112_60_8	397619.37	500495.3875	397619.37	397619.37	2.8
d15112_60_9	673840.81	910298.6184	673840.81	673840.81	2.9
nrw1379_60_0	64359.77	80086.56654	64359.77	64359.77	2.88
nrw1379_60_1	83410.67	104646.3375	83410.67	83410.67	2.96
nrw1379_60_2	52858.87	70986.81333	52858.87	52858.87	2.96
nrw1379_60_3	62341.36	84434.20476	62341.36	62341.36	2.84
nrw1379_60_4	56012.13	69680.90881	56012.13	56012.13	2.9
nrw1379_60_5	58083.8	72973.325	58083.8	58083.8	2.9
nrw1379_60_6	52224.66	65749.025	52224.66	52224.66	2.92
nrw1379_60_7	58402.97	73290.6375	58402.97	58402.97	2.94
nrw1379_60_8	52145.08	66101.85821	52145.08	52145.08	2.96
nrw1379_60_9	49026.52	66572.84307	49026.52	49026.52	2.86

solution, and the average time in seconds of ten executions obtained by our algorithm, respectively. In this work, we choose several state-of-the-art metaheuristic algorithms for some variants of the CmTRP [1], [6], [10], [12], [15], [16] as a baseline in our research.

Tables I to III show the average gap of the improvement phase in comparison with the construction is 8.70%. The average gap value is not too large. This indicates that the construction phase gives good feasible solutions fast. The proposed algorithm consumes much time for the instances with up to 200 vertices. Therefore, the fist way to decrease the running time is only to run the construction phase. In this case, the proposed algorithm suffers from a slightly loss of 8.70% solution quality on average. Although we cannot compare the results directly to other algorithms in the literature (note that in the algorithms for some variants of the mTRP, the capacity constraint is removed), we succeed in producing feasible solutions for instances with 200 customers. It is an important contribution when finding feasible solutions is also NP-hard.

Table IV shows the evolution of average gap during the iterations. The average gaps are 6.35%, 6.85%, 7.28%, 7.99%, 8.30%, 8.70%, and 8.70% in comparison with the initial solution, obtained by one, five, ten, twenty, thirty, forty, fifty, and one-hundred iterations, respectively. No improvement obtains from fifty to one-hundred iterations. Therefore, additional iterations give a minor improvement while it consumes much time. Hence, the second way to reduce the running time is to use no more than fifty iterations, and the improvement reaches about 8.70%. The fastest option is to run the construction phase and then improve it by using a single iteration, which obtains an average gap of 6.35% and average time of 0.5 seconds.

Table V shows the difference between the objective function of two problems on the same instances. The average difference of 22.18% indicates the capacity constraint also

TABLE VII. OUR RESULTS FOR THE MTRP-INSTANCES WITH 70 VERTICES PROPOSED BY LUO

Instances	*OPT*	*Init.Sol*	*Best.Sol*	*Aver.Sol*	*T*
pr1002_70_0	429557.7	429557.7	429557.7	530946.01	3.12
pr1002_70_1	430048.06	430048.06	430048.06	356469.79	3.08
pr1002_70_2	377233.86	377233.86	377233.86	344118.14	2.68
pr1002_70_3	429562.01	429562.01	429562.01	429604.2	2.64
pr1002_70_4	435659.17	435659.17	435659.17	435655.25	2.96
pr1002_70_5	429584.16	429584.16	429584.16	668129.73	3.34
pr1002_70_6	344534.44	344534.44	344534.44	406678.77	2.82
pr1002_70_7	393558.46	393558.46	393558.46	311254.73	3.02
pr1002_70_8	397072.39	397072.39	397072.39	469816.84	2.74
pr1002_70_0	429557.7	429557.7	429557.7	277336.06	3.12
brd14051_70_0	191843.35	191843.35	191843.35	213420.42	2.76
brd14051_70_1	169340.01	169340.01	169340.01	218315.68	2.98
brd14051_70_2	216195.95	216195.95	216195.95	151666.85	3.12
brd14051_70_3	229328.9	229328.9	229328.9	172199.83	3.28
brd14051_70_4	302498.42	302498.42	302498.42	133952.5	3.34
brd14051_70_5	179470.31	179470.31	179470.31	203145.14	3
brd14051_70_6	231693.74	231693.74	231693.74	136233.51	2.66
brd14051_70_7	284960.31	284960.31	284960.31	171879.58	2.66
brd14051_70_8	167533.17	167533.17	167533.17	191580.79	2.76
brd14051_70_9	253499.74	253499.74	253499.74	128178.58	3.26
fnl4461_70_0	154805.67	154805.67	154805.67	156260.54	2.76
fnl4461_70_1	104585.82	104585.82	104585.82	103190.13	3.24
fnl4461_70_2	161892.44	161892.44	161892.44	109739.93	2.74
fnl4461_70_3	99122.23	99122.23	99122.23	100792.2	3.32
fnl4461_70_4	157106.13	157106.13	157106.13	149638.18	2.84
fnl4461_70_5	112094.64	112094.64	112094.64	158679.44	2.72
fnl4461_70_6	121521	121521	121521	122266.92	2.74
fnl4461_70_7	175859.51	175859.51	175859.51	107469.11	3.06
fnl4461_70_8	122141.15	122141.15	122141.15	100531.72	2.94
fnl4461_70_0	154805.67	154805.67	154805.67	135829.76	2.76
d15112_70_0	517426.18	517426.18	517426.18	684939.42	3.24
d15112_70_1	715678.26	715678.26	715678.26	644759.99	3.02
d15112_70_2	688605.9	688605.9	688605.9	425069.33	3
d15112_70_3	625623.9	625623.9	625623.9	528177.95	3.32
d15112_70_4	532088.98	532088.98	532088.98	586915.82	2.78
d15112_70_5	500455.25	500455.25	500455.25	422195.61	3.18
d15112_70_6	497229.6	497229.6	497229.6	518793.6	3.18
d15112_70_7	599776.85	599776.85	599776.85	616918.44	2.86
d15112_70_8	576957.51	576957.51	576957.51	397619.37	3.02
d15112_70_9	775176.3	775176.3	775176.3	673840.81	2.6
nrw1379_70_0	66839.83	66839.83	66839.83	64359.77	2.58
nrw1379_70_1	65103.43	65103.43	65103.43	83410.67	2.98
nrw1379_70_2	63480.7	63480.7	63480.7	52858.87	3.2
nrw1379_70_3	59273.92	59273.92	59273.92	62341.36	3.32
nrw1379_70_4	70594.56	70594.56	70594.56	56012.13	2.66
nrw1379_70_5	73884.17	73884.17	73884.17	58083.8	3.02
nrw1379_70_6	64306.14	64306.14	64306.14	52224.66	2.94
nrw1379_70_7	90554.87	90554.87	90554.87	58402.97	2.54
nrw1379_70_8	91738.43	91738.43	91738.43	52145.08	2.82
nrw1379_70_9	68024.3	68024.3	68024.3	49026.52	2.68

TABLE VIII. OUR RESULTS FOR MTRP-INSTANCES WITH 80 VERTICES PROPOSED BY LUO

Instances	*OPT*	*Init.Sol*	*Best.Sol*	*Aver.Sol*	*T*
pr1002_80_0	491764.64	656239.68	491764.64	491764.64	9.48
pr1002_80_1	442164.21	613287.46	442164.21	442164.21	9.18
pr1002_80_2	505524.17	609954.56	505524.17	505524.17	9.32
pr1002_80_3	436752.96	614611.29	436752.96	436752.96	9.1
pr1002_80_4	453609.46	587470.83	453609.46	453609.46	9.36
pr1002_80_5	599492.4	771733.95	599492.4	599492.4	9.16
pr1002_80_6	619003.36	805206.35	619003.36	619003.36	9.4
pr1002_80_7	508186.51	658640.85	508186.51	508186.51	9.42
pr1002_80_8	409733.88	518052.51	409733.88	409733.88	9.44
pr1002_80_9	557220.48	670387.49	557220.48	557220.48	9.28
brd14051_80_0	336983.07	403178.34	336983.07	336983.07	9.04
brd14051_80_1	277861.02	348787.38	277861.02	277861.02	9.14
brd14051_80_2	265370.92	321922.47	265370.92	265370.92	9.56
brd14051_80_3	189815.69	240361.74	189815.69	189815.69	9.1
brd14051_80_4	206068.45	275228.43	206068.45	206068.45	9.5
brd14051_80_5	303621.75	348578.27	303621.75	303621.75	9.32
brd14051_80_6	213405.23	266958.37	213405.23	213405.23	9.6
brd14051_80_7	263737.93	308039.16	263737.93	263737.93	9.04
brd14051_80_8	232967.83	298574.84	232967.83	232967.83	9.26
brd14051_80_9	317790.55	368183.51	317790.55	317790.55	9.06
fnl4461_80_0	153124.51	194685.8	153124.51	153124.51	9.58
fnl4461_80_1	174975.64	224516.74	174975.64	174975.64	9
fnl4461_80_2	162755.5	197782.39	162755.5	162755.5	9.46
fnl4461_80_3	160819.04	192927.87	160819.04	160819.04	9.5
fnl4461_80_4	151790.69	187440.09	151790.69	151790.69	9.52
fnl4461_80_5	131045.47	172293.83	131045.47	131045.47	9.04
fnl4461_80_6	125405.93	166418.99	125405.93	125405.93	9.24
fnl4461_80_7	125228.91	164627.71	125228.91	125228.91	9.16
fnl4461_80_8	185280.87	228208.31	185280.87	185280.87	9.48
fnl4461_80_9	130304.95	165022.1	130304.95	130304.95	9.26
d15112_80_0	551900.43	753989.49	551900.43	551900.43	9.56
d15112_80_1	815029.39	979921.76	815029.39	815029.39	9.1
d15112_80_2	828114.32	1080571.45	828114.32	828114.32	9.16
d15112_80_3	689450.94	964458.54	689450.94	689450.94	9.1
d15112_80_4	560385.47	737417.48	560385.47	560385.47	9.08
d15112_80_5	821959.4	1030515.79	821959.4	821959.4	9.52
d15112_80_6	715206.03	882086.8	715206.03	715206.03	9.36
d15112_80_7	958278.86	1155190.13	958278.86	958278.86	9.32
d15112_80_8	990277.77	1174384.17	990277.77	990277.77	9.1
d15112_80_9	672457.47	931587.71	672457.47	672457.47	9.5
nrw1379_80_0	64831.76	96656.39	64831.76	64831.76	9.36
nrw1379_80_1	64967.83	88394.72	64967.83	64967.83	9.22
nrw1379_80_2	73858.13	96499.97	73858.13	73858.13	9.3
nrw1379_80_3	100592.83	131733.34	100592.83	100592.83	9.24
nrw1379_80_4	98228.29	126451.61	98228.29	98228.29	9.04
nrw1379_80_5	75984.21	99492.47	75984.21	75984.21	9.14
nrw1379_80_6	79165.6	105024.23	79165.6	79165.6	9.08
nrw1379_80_7	73194.55	105328.52	73194.55	73194.55	9.1
nrw1379_80_8	83492.62	115793.31	83492.62	83492.62	9.14
nrw1379_80_9	67034.31	92380.75	67034.31	67034.31	9.24

affects the quality of solutions. However, the best solutions for the mTRP are not feasible solutions for the CmTRP. Therefore, the methods designed for the mTRP instances may not be adapted easily to solve the CmTRP.

From Tables VI to VIII show that the efficiency of the proposed algorithm is very good for the mTRP-instances since it can reach the optimal solutions for the instances with up to 80 vertices at a reasonable amount of time. In Table IX, in comparison with Ban et al.'s [1], Ezzine et al.'s [6], and Nucamendi-Guillén's [16] algorithm, our solutions are better than those of Ban et al.'s, and Ezzine et al. in all cases while they are comparable with Nucamendi-Guillén's solutions in the most of instances. In many cases, our algorithm obtains the optimal solution for the instance with 76 vertices. Moreover, for the CCVRP in Table X, our algorithm reaches the known best solutions for the instances with 100 vertices (note that: the best solutions are extracted from [10], [15]).

The running time of the proposed algorithm grows quite moderate compared to the Nucamendi-Guillén's algorithm [16] while it is comparable with those of Ban et al.'s [1], and Ezzine et al.'s algorithm [6].

TABLE IX. COMPARISONS WITH THE PREVIOUS ALGORITHMS FOR MTRP-INSTANCES PROPOSED IN [18]

Instances	Ban et al.	Ezzine et al.	Nucamendi-Guillén et al.	Our results	
				Best.Sol	*T*
E30k3	2108.26	-	-	2097.3	0.25
E30k4	2623.65	-	-	2595.11	0.22
E51k5	2623.65	3320	2209.64*	2209.64	0.41
E76k10	2786.07	4094	2310.09*	2419.89	0.78
E76k14	2201.13	3762	2005.4*	2005.4	0.71
E76k15	2400.17		-	2377.5	0.81
E101k8	-	6383	-	4051.47	2.52
E101k14	-	5048	-	3288.53	2.50
P40k5	1793.14	-	1537.79*	1580.21	0.27
P45k5	2336.43	-	1912.31*	1912.31	0.32
P50k7	1878.81	-	1547.89*	1590.41	0.58

Note that: symbol '*'is the optimal value

TABLE X. COMPARISONS WITH THE PREVIOUS ALGORITHMS FOR CCVRP-INSTANCES PROPOSED BY [18]

Instances	*BKS*	*Best.Sol*	*T*
CMT1	2230.35	2230.35	1.70
CMT2	2391.63	2429.18	6.19
CMT3	4045.42	4073.12	17.93
CMT4	4987.52	4987.52	85.94
CMT5	5809.59	5838.32	295.67
CMT11	7314.55	7314.55	31.09
CMT12	3558.92	3559.43	15.84

IV. DISCUSSIONS

Due to NP-hard problem, metaheuristic approach is suitable approach to solve the problem with large sizes in a short time. Currently, several algorithms for the close variants of CmTRP have been proposed. However, these algorithms cannot apply to the CmTRP because the capacity constraint does not include in them. The results in Table V indicate that the algorithms for the mTRP produce infeasible solutions for the CmTRP. Therefore, developing efficient algorithm for the CmTRP is our contribution. The VNS, and GVNS [14] are a general schemes that are used widely to solve many optimization problems. However, to apply them for a specific problem, they require many efforts. Specifically, how to use and combine neighborhoods to explore and exploit good solution space, how many neighborhoods are used to balance between solution quality and running time, and how to balance between diversity and intensification. Therefore, in the literature, many algorithms are developed on the VNS scheme for a specific problem, but the solution quality of them is different. Applying the VNS scheme to solve the problem effectively is our contribution in this work.

For a metaheusitic approach, a classical and successful recipe is to combine 1) efficient "intensification" procedures via local searches with 2) "diversification" methods. Diversification means to generate diverse solutions to explore the search space on a global scale, while intensification means to focus on the search in a local region by exploiting the information that a current good solution is found in this region. Our algorithm uses the VNS, GVNS to implement intensification while shaking maintains diversification. Though the initial solution is generated far from the global minima, the explored solution region is extended. Therefore, the chance for finding good solutions is high. The CmTRP is harder than the mTRP because it is the general case of the mTRP. For constrained NP-hard problem like the CmTRP, finding feasible solution is also NP-hard. The new contribution in this work is applied the VNS in both of two phases. The first phase generates feasible solution while the post one improve solution quality.

With respect to the instances, it indicates that the proposed algorithm can be used as follows:

- The fist way is to only run the construction phase with a rather loss of 8.7% solution quality on average. It is the fastest option.
- The second way is to run the construction phase and then enhance it by using one iteration. As the result, our algorithm obtains an average improvement of 6.35%, and an average time of 0.5 seconds. It balances between solution quality and running time.
- The last is to run the construction phase and improve it no more than fifty iterations, and the average improvement reaches about 8.7%, and an average time of 29.07 seconds. It is the best option in terms of solution quality.

Moreover, in comparison with some close variants of the mTRP, the proposed algorithm obtains the optimal solutions for the instances with up to 80 vertices in a short time. Moreover, it also reaches the better solutions than the algorithms in [1], [6] while it is comparable with the algorithms in [10], [12], [15], [16]. It shows that our algorithm is still effective for various problems.

V. CONCLUSIONS

In this paper, we propose a metaheuristic algorithm which applies the VNS in both of phases. The first phase creates a feasible solution while the post one improve it. The optimal solutions can be reached for the problem with up to 80 vertices in several seconds. The solution's quality is comparable with the previous algorithms for the other cases. Moreover, we give three options to use the proposed algorithm effectively. However, the running time needs to enhance to meet practical situations. It will be our aim in future research.

ACKNOWLEDGMENT

This research is funded by Hanoi University of Science and Technology (HUST) under grant number T2018-PC-208.

REFERENCES

[1] H.B. Ban, "An effective GRASP+VND metaheuristic for the k-Minimum Latency Problem", Proc. RIVF, 2016, pp. 31-36.

[2] H.B. Ban, "General Variable Neighborhood Search for the Quote-Travelling Repairman Problem", J. IJACSA, No. 4, 2020 (To appear).

[3] H.B. Ban, "A GRASP+VND Algorithm For The Multiple Traveling Repairman Problem With Distance Constraints", J. JCC, Vol.33, No.3, 2017, pp. 272-288.

[4] H.B. Ban, and D.N. Nguyen, "Improved genetic algorithm for Minimum Latency Problem", Proc. SOICT, 2010, pp. 9-15.

[5] H.B. Ban, K. Nguyen, M.C. Ngo, and D.N. Nguyen, "An efficient exact algorithm for Minimum Latency Problem", J. PI, No.10, 2013, pp. 1-8.

[6] I. O. Ezzine, and Sonda Elloumi, "Polynomial Formulation and Heuristic Based Approach For The k-Travelling Repairman Problem", Int. J. Mathematics In Operational Research, Vol. 4, No. 5, 2012, pp. 503-514.

[7] F. Jittat, C. Harrelson, and S. Rao, "The k-Traveling Repairman Problem", Proc. ACM-SIAM, 2003, pp.655-664.

[8] D. S. Johnson, and L. A. Mcgeoch, "The Traveling Salesman Problem: A Case Study In Local Optimization In Local Search In Combinatorial Optimization", E. Aarts and J. K. Lenstra, Eds., pp. 215-310.

[9] R. Jothi, and B. Raghavachari, "Minimum Latency Tours and The k-Traveling Repairmen Problem", Proc. LATIN, 2004, pp. 423–433.

[10] L. Ke, and Z. Feng, "A Two-Phase Metaheuristic For The Cumulative Capacitated Vehicle Routing Problem", J. Computers and Operations Research, Vol. 40, No. 2, pp. 633-638, 2013.

[11] Y. Lua, U. Benlic, Q. Wua, and B. Peng, "Memetic algorithm for the multiple traveling repairman problem with profits", J. EAAI, Vol. 80, 2019, pp. 35-47.

[12] Z. Luo, H. Qin, and A. Lim, "Branch-and-Price-and-Cut for The Multiple Traveling Repairman Problem With Distance Constraints", J. Operations Research, Vol. 234, No. 1, 2013, pp. 49-60.

[13] O. Martin, S. W. Otto, and E. W. Felten, "Large-Step Markov Chains For The Traveling Salesman Problem", J. Complex Systems, Vol. 5, No. 3, 1991, pp. 299-326.

[14] N. Mladenovic, and P. Hansen, "Variable Neighborhood Search", J. Operations Research, Vol.24, No. 11 24, 1997, pp.1097-1100.

[15] SU. Ngueveu, C. Prins, and R. Wolfler Calvo, An Effective Memetic Algorithm for the Cumulative Capacitated Vehicle Routing Problem, J. Computers and Operations Research, Vol 37, No. 11, pp. 1877-1885, 2010.

[16] S. Nucamendi-Guillén, I. Martínez-Salazar, F. Angel-Bello, and J. M. Moreno-Vega, "A Mixed Integer Formulation and An Efficient Metaheuristic Procedure For The K-Travelling Repairmen Problem", J. JORS, Vol. 67, No. 8, 2016, pp. 1121-1134.

[17] A. Salehipour, K. Sorensen, P. Goos, and O.Braysy, "Efficient GRASP+VND and GRASP+VNS Metaheuristics for the Traveling Repairman Problem", J. Operations Research, Vol. 9, No. 2, 2011, pp.189-209.

[18] NEO (2013), http://neo.lcc.uma.es/vrp/vrp-instances/capacitated-vrp-instances.