Computing and Informatics, Vol. 38, 2019, 883–916, doi: 10.31577/cai_2019_4_883

AN EFFECTIVE METAHEURISTIC FOR MULTIPLE TRAVELING REPAIRMAN PROBLEM WITH DISTANCE CONSTRAINTS

Ha-Bang Ban

School of Information and Communication Technology
Hanoi University of Science and Technology
Hanoi, Vietnam
&
FPT University, Hanoi, Vietnam
e-mail: BangBH@soict.hust.edu.vn

Duc-Nghia Nguyen

School of Information and Communication Technology
Hanoi University of Science and Technology
Hanoi, Vietnam
e-mail: NghiaND@soict.hust.edu.vn

Kien Nguyen

Graduate School of Science and Engineering, Chiba University, Japan
e-mail: nguyen@chiba-u.jp

Abstract. Multiple Traveling Repairman Problem with Distance Constraints (MTRPD) is an extension of the NP-hard Multiple Traveling Repairman Problem. In MTRPD, a fleet of identical vehicles is dispatched to serve a set of customers with the following constraints. First, each vehicle's travel distance is limited by a threshold. Second, each customer must be visited exactly once. Our goal is to find the visiting order that minimizes the sum of waiting times. To solve MTRPD we propose to combine the Insertion Heuristic (IH), Variable Neighborhood Search (VNS), and Tabu Search (TS) algorithms into an effective two-phase metaheuristic that includes a construction phase and an improvement phase. In the former phase, IH is used to create an initial solution. In the latter phase, we use VNS to generate various neighborhoods, while TS is employed to mainly prohibit from getting trapped into cycles. By doing so, our algorithm can support the search to escape local optima. In addition, we introduce a novel neighborhoods' structure and a constant time operation which are efficient for calculating the cost of each neighboring solution. To show the efficiency of our proposed metaheuristic algorithm, we extensively experiment on benchmark instances. The results show that our algorithm can find the optimal solutions for all instances with up to 50 vertices in a fraction of seconds. Moreover, for instances from 60 to 80 vertices, almost all found solutions fall into the range of 0.9 %–1.1 % of the optimal solutions' lower bounds in a reasonable duration. For instances with a larger number of vertices, the algorithm reaches good-quality solutions fast. Moreover, in a comparison to the state-of-the-art metaheuristics, our proposed algorithm can find better solutions.

Keywords: Traveling repairmen problem, distance constraints, insertion heuristic, tabu search, variable neighborhood search

1 INTRODUCTION

The Traveling Repairman Problem (TRP), which is also known as the Minimum Latency Problem (MLP) or the Deliveryman Problem (DMP), has been studied in the number of previous works [1, 2, 3, 4, 5, 6, 10, 11, 18]. The problem arises when repairmen or servers have to accommodate a set of requests to minimize the total or average waiting times [1, 2, 8, 10]. A direct generalization of the TRP is the Multiple Traveling Repairman Problem (MTRP) that considers multiple vehicles or travelers. Similar to TRP, there are several prior studies in the literature for MTRP [22, 9, 23, 28, 29]. Applications of the MTRP can be found in routing Pizza deliverymen or scheduling machines to minimize mean flow time for jobs [17]. In this paper, we study an extension of MTRP, namely the Multiple Traveling Repairmen Problem with Distance Constraints (MTRPD), which involves distance constraints. In MTRPD, the route length or maximum duration of each vehicle cannot exceed a predetermined limit (MD). This type of constraint usually stems from regulations on working hours for workers. Other examples of vehicle routing models that incorporate the distance constraint can be found in [30]. In MTRPD, we consider 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, the distance constraint is respected and the total waiting time of all customers is minimized.

MTRPD is at least as hard as TRP and MTRP. MPTRPD, which is also NP-hard problem, can be formulated as follows.

Given a complete graph K_n with the vertex set $V = \{1, 2, \dots, n\}$, a symmetric distance matrix $C = \{c(i,j) \mid i,j = 1, 2, \dots, n\}$, where $c(i,j)$ is the distance between two vertices i and j, and a predetermined limit L. Let $R = (1, 2, \dots, k)$

be a set of k vehicles which begin at the main depot v_1. Suppose that the tour $T = (R_1, \ldots, R_l, \ldots, R_k)$ is a set of obtained routes from k vehicles. Let $R_l = (v_1, \ldots, v_h, \ldots, v_m)$ $(1 < m \leqslant n)$ be a route of vehicle l $(l \in R)$. $P(v_1, v_h)$ is the path from v_1 to v_h on the route R_l and $l(P(v_1, v_h))$ is its length. The waiting time of a vertex v_h $(1 < h \leqslant m)$ on R_l is the length of the path from starting vertex v_1 to v_h:

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

The waiting time of R_l is defined as the sum of waiting times of all vertices in this route. It must satisfy the below constraint:

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

$$L(R_l) = \sum_{i=1}^{m-1} c(v_i, v_{i+1}) \leqslant MD.$$

The total waiting time of T is the sum of all the vertices' waiting times:

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

MTRPD asks for a k-route, which starts at a given vertex v_1, visits each vertex in the graph once exactly with the total waiting time of all vertices being minimized. Like other NP-hard problems, there are three main approaches to solve MTRPD:

1. exact algorithms,
2. approximation algorithms, and
3. heuristic algorithms.

The first approach guarantees to find the optimal solution that takes exponential time in the worst case. However, the exact algorithm only solves with up to 50 vertices [25]. In the second approach, we denote an approximation algorithm as p-approximation when the algorithm finds the solution at most p times worse than the optimal one. Here p is the approximation ratio, which has a constant value. Up to date, the best approximation ratio is 16.994 for the MTRP [22], which is still far from the optimal solution. In the third approach, the proposed heuristic algorithms perform well in practice and their performance is validated on an experimental benchmark of interesting instances. The metaheuristic algorithm also falls in the third approach.

Research on the MTRPD has not studied much and only one meta-heuristic approach for this problem has been proposed in [9]. Ban's algorithm in [9] is mainly based on the principles of the Variable Neighborhood Decent (VND). However, Ban's algorithms might become trapped into cycles. That means they return to the points previously explored in the solution space. Consequently, the algorithms can get stuck in local optima. In this article, we investigate the global structure of the MTRPD solution space. Based on the investigation, a meta-heuristic algorithm that combines the Tabu search (TS) and Variable Neighborhood Search (VNS) is proposed. In the algorithm, TS is used to avoid getting trapped into cycles. Therefore, it supports the search to escape from local optima. In a cooperative way, VNS is employed to generate various neighborhoods for the TS. Moreover, we also introduce a novel neighborhoods' structure for VNS and present a constant time operation for calculating the cost of each neighboring solution. Therefore, the extension of explored part of the solution space obtained by using various neighborhoods, which can increase chances of finding better solution, is not time-consuming in our algorithm. Extensive computational experiments on benchmark instances show that the proposed algorithm is able to find the optimal solutions for all instances with up to 50 vertices in a fraction of seconds. Moreover, almost all found solutions for instances from 60 to 80 vertices fall into the range of 0.9 %–1.1 % of the lower bounds of the optimal solutions at reasonable amount of time. For larger instances, our algorithm obtains good-quality solutions fast and the new best solutions are found in comparison with the state-of-the-art metaheuristics.

The rest of this article is organized as follows. Section 2 introduces the global structure of the solution space of MTRPD. Section 3 presents the proposed algorithm. Section 4 contains the evaluation. Finally, Section 5 concludes the article.

2 INVESTIGATION OF MTRPD SOLUTION SPACE

The structure of the MTRPD solution takes an important part in improving a suitable algorithm to solve the problem. However, to the best of our knowledge, there has not been a previous work that would solve the global structure of the MTRPD problem. That motivates us to investigate the global structure of the MTRPD solution space.

In an intuitive way, the distance between two tours T_1 and T_2 of the problem is defined as the minimum number of transformations from T_1 to T_2, denoted by $d(T_1, T_2)$. Since no polynomial method for computing $d(T_1, T_2)$ has been known, we define $d(T_1, T_2)$ to be n minus the number of vertices which have the same position in both T_1 and T_2. We see that this distance approximates the number of 2-opt operations (2-opt is a local search described in Section 3) required to transform one tour into another, to within a factor of two. Therefore, $d(T_1, T_2)$ is the good measure of proximity between solutions produced by 2-opt.

We have selected two instances (d15112-x and pr1002-x) from the dataset in [25] and implemented with 2-opt. The selection reason is that the optimal solutions of both instances are provided in [25]. Running each instance with 2-opt, we obtain locally optimal solutions. The larger the number of 2-opt runs, the better the visualization of the MTRPD solution space. The pilot experiment shows that the value

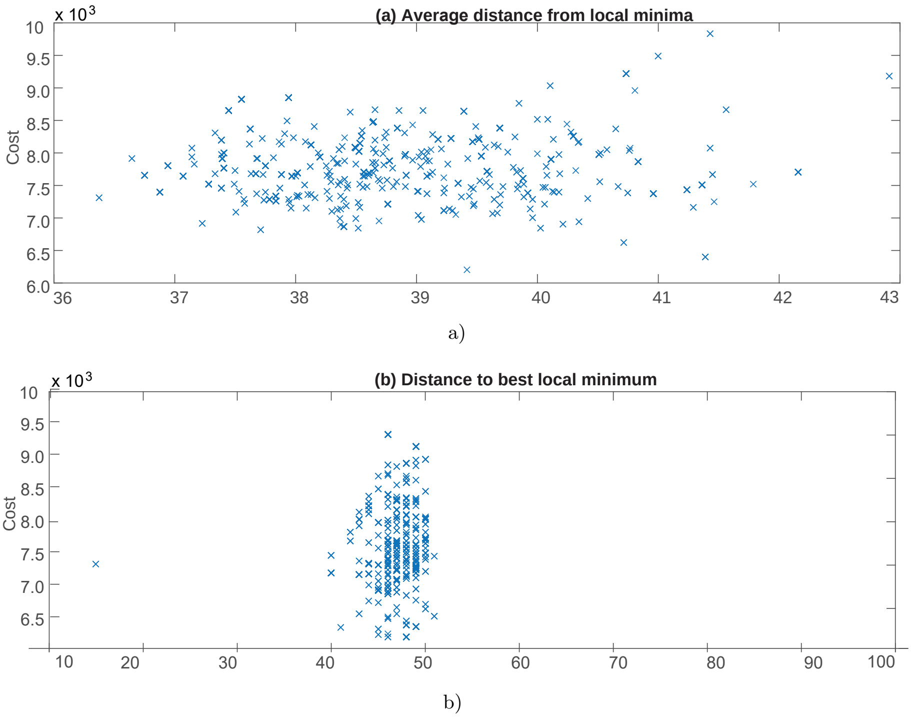
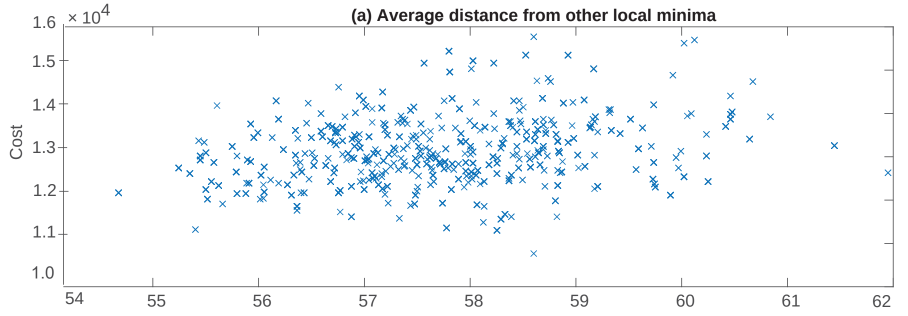


Figure 1. 2 000 random 2-opt local minima for d15112-x. Tour cost (vertical axis) is plotted against a) mean distance to the 1 999 other local minima and b) distance to the global minimum.

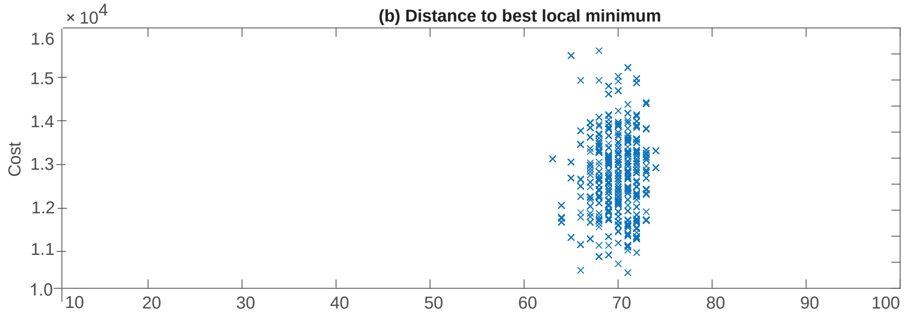
of 2 000 is good enough for the investigation. We run 2-opt 2 000 times to produce 2 000 locally optimal solutions. Then, for each of those solutions, we compute the average distance to the other 1999 solutions, measured by the distance metric d. The results for d15112-x and pr1002-x are presented in Figures 1 and 2, respectively.

In Figures 1 a) and 2 a), we can realize a clear correlation as follows. The optimal minimum appears to be central to all other local minima. Moreover, indeed, a prominent valley structure can be said to govern the set of locally minimum solutions. We can gain further insights from Figures 1 b) and 2 b), which plot the costs of the same 2 000 local minima against their distances from the optimal minimum solution found. It indicates that the average distance between two random solutions is just under $(n - 2)$. The experiment shows that the MTRPD solution space exhibits a global convex (i.e., the so-called big valley structure in Figure 3). That means the set of local optima appears convex with one central global optimum.

As mentioned earlier, MTRPD has shown to be NP-Hard because it is a generalization of TRP, which is NP-hard [10, 18]. Therefore, a metaheuristic needs to be developed to provide near-optimal solutions within a short computation time



a)



b)

Figure 2. 2 000 random 2-opt local minima for pr1002-x. Tour cost (vertical axis) is plotted against a) mean distance to the 1 999 other local minima and b) distance to the global minimum.

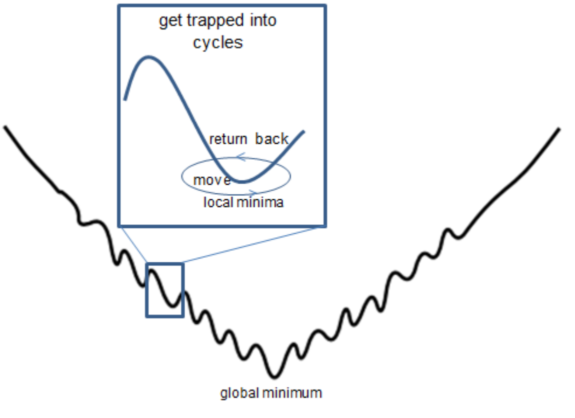


Figure 3. Intuitive scheme of the "big valley" solution space structure

for large instance sizes. Moreover, the big valley structure suggests the idea of the hybrid approach that combines the TS and VNS algorithms. First, in the valley structure, the best elite solutions created by the VNS dispersed over it. Second, TS is perfectly attracted to big valley area. Even though the initial solution was set far from the valley, TS still can prevent from getting trapped into cycles to drive the search to the big valley. The above observations indicate that the combination between TS and VNS is suitable for finding good solutions inside the big valley.

3 OUR METAHEURISTIC APPROACH

We propose the efficient and straightforward metaheuristic algorithm that brings together the components of IH, TS [19], and VNS [27]. The proposed algorithm includes two phases:

1. IH in the construction phase and
2. VNS and TS in the improvement phase.

The two phases could be divided into five detailed steps, as shown in Algorithm 1. In Step 1, the algorithm starts with an initial solution obtained from IH. The following four main steps are repeated until a stop condition is met. In Step 2, we introduce a novel neighborhoods' structure in VNS [27]. Moreover, in order to avoid tabu move, tabu lists are used. The idea of voiding possibility repeatedly in TS [19] is to make tabu lists of the recent types of moves in the space solution, and prohibit reversing these moves. The move here is a transition from one solution to another. In Step 3, a list of promising solutions is built up, and the list serves as input for Step 4. The step aims at exploiting the current solution space. To explore the entire solution space, a diversification phase is added in Step 5. Further in this section we describe the five steps of our algorithm in more details.

Step 1: We use the insertion heuristic which is given in Algorithm 2 for finding an initial solution. Consider a partial tour, and define the set $\overline{V}$ as the set of all non-visited nodes, $\overline{V} \subseteq V$. To improve the partial tour, a node from $\overline{V}$ should be added. This process requires two decisions: which vertex to insert and where to place it in the tour. We use two insertion schemes to keep the balance between pure greediness and overall layout of the tour. The major difference between the two is the order in which the vertices are inserted.

Cheapest insertion: Among all vertices not inserted so far, choose a vertex whose insertion causes the lowest increase in the cost of the tour. The idea behind this strategy is undoubtedly pure greediness.

Farthest insertion: Insert the vertex whose minimal distance to a tour vertex is maximal. The idea behind this strategy is to fix the overall layout of the tour early in the insertion process.

Several main steps in IH-procedure is repeated until a feasible solution is found or a stop condition is met. If any feasible solution is found, it is considered as

Algorithm 1 Our VNS + TS Metaheuristic Algorithm

Input: v_1, K_n, k are a starting vertex, the complete graph, and the number of vehicles, respectively.
Output: the best solution T^*.

```
Step 1 (Generate an initial solution):
T ← IH-Procedure(v_1, V, k); {initiate the best solution}
T* ← T {LT is the list of promising solutions}
LT ← ∅
while stop criteria not met do
  Step 2 (VNS):
  for i : 1 → 10 do
    T' ← arg min N_i(T); {local search}
    if ((W(T') < W(T) and T' is not tabu) or (W(T') < W(T*))) then
      T ← T'
      i ← 1
      update tabu lists;
      if (W(T') < W(T*)) and (T' must be a feasible solution) then
        T* ← T';
      end if
    else
      i++
    end if
  end for
  Step 3 (Built up promising solutions list L_T):
  if W(T) < (1 + ST)W(T*) then
    LT ← LT ∪ T;
  end if
  if (| LT |== sLT) then
    go to Step 2;
  end if
  Step 4 (Implement Intensification):
  for j : 1 → sLT do
    perform VNS as in Step 2 without tabu list with an element of LT as start
    solution;
  end for
  Step 5 (Implement Diversification):
  Clear all tabu lists and update attribute matrix M;
  select a random tour T in LT;
  T ← shaking-procedure(T);
end while
return T*;
```

Algorithm 2 IH-Procedure(v_1, K_n, k)

Input: v_1, K_n, k are a starting vertex, the complete graph, and the number of vehicles, respectively.
Output: An initial solution T. $\{LIT$ is the list of infeasible tours$\}$

$LT = \phi$;
$T = v_1$;
while stop criteria not met **do**
  for $(l = 1; l < k; l{+}{+})$ **do**
    $R_l = R_l \cup v_1$; $\{$The l^{th} route of the tour T starts at a main depot $v_1\}$
  end for $\{L$ is the list of visited vertices in $K_n\}$
  $L = \phi$;
  while $|T| < n$ **do**
    $l = \text{random}(k)$; $\{$Choose a route randomly in k routes$\}$
    $rd = \text{random}(2)$; $\{$Choose an insertion scheme randomly$\}$
    if $rd == 1$ **then**
      Arbitrary select a vertex v that is not yet in the partial route and an inserted position $j < |R_l|$ at time t_j so that the cost of R_l' ($R_l' = \text{Insert}(R_l, j, v)$) is minimal; $\{$Cheapest Insertion$\}$
    else
      Arbitrary select a vertex v that is not yet in the partial tour and an inserted position $j < |R_l|$ at time t_j so that $c(v_j, v, j)$ is minimal and the cost of R_l' ($R_l' = \text{Insert}(R_l, j, v)$) is maximal; $\{$Farthest Insertion$\}$
    end if
    $R_l \leftarrow R_l'$
  end while
  if T is a feasible solution **then**
    return T;
  else
    $LIT = LIT \cup \{T\}$;
  end if
  if $|LIT| > n - 1$ **then**
    choose a tour with the minimum cost in LIT;
    exit();
  end if
end while

the initial solution. Conversely, it is added into the list LIT that is used to store all infeasible tours. Since the size of LIT is n, we choose a tour with the minimum cost in LIT as the initial solution.

Step 2: In this step, ten neighborhoods investigated are divided into two categories: intro-route, and intra-route. Intro-route is used as a post-optimizer on single vehicle routes. It includes remove-insert, swap-adjacent, swap, move-up(down), move-forward(backward)-k-vertices [21]. Meanwhile, solution improvements can

Algorithm 3 Shaking(T, M, pos)

Input: T, M, pos are the tour, attribute matrix, and the number of swap, respectively.
Output: a new route T'.
Select randomly a route R_l in T;
T = Shaking-intro-route(R_l, M, l, pos).
Select randomly two routes R_l and R_h in T;
T = Shaking-intra-routes(R_l, R_h, pos).
return T;

Algorithm 4 Shaking-intro-route(R_l, M, l, pos)

Input: R_l, M, l, pos are the $l-th$ route, attribute matrix, and the number of times an edge is present in an element of the promising solutions list, the number of swap, respectively.
Output: a new solution T.
while ($pos > 0$) **do**
  {select i, j from $[1, n]$ at random}
  $i \longleftarrow$ Random$(1, n)$;
  $j \longleftarrow$ Random$(1, n)$;
  if ($i \neq j$) **then**
    if (edge($R_l[i], R_l[j]$) and edge($R_l[i], R_l[j+1]$) are not in M more than l times) **then**
      Insert $R_l[i]$ between $R_l[j]$ and $R_l[j+1]$;
      $pos \longleftarrow pos - 1$;
    end if
  end if
end while
update R_l in T;
return T;

Algorithm 5 Shaking-intra-routes(R_l, R_h, pos)

Input: R_l, R_h, pos are the l^{th}, h^{th} route, the number of vehicles and the number of swap, respectively.
Output: a new solution T.
while ($pos > 0$) **do**
  select i^{th} and j^{th} positions from R_l and R_h at random, respectively;
  swap $R_l[i]$ between $R_h[j]$;
  $pos = pos - 1$;
end while
update T;
return T;

be obtained by moving vertices belonging to two or more different routes in intra-route. In this work, we introduce new neighborhoods in intra-route such as swap-2-route, and insert-2-route. For a given current solution T, neighborhood explores the neighboring solution space set $N(T)$ of T iteratively and tries to replace T by the best solution $T' \in N(T)$. The main operation in exploring the neighborhood is the calculation of the cost of a neighboring solution. In a straightforward implementation in the worst case, this operation requires $Tsol = O(n)$. In this paper, by using the known cost of the current solution, we show that this operation can be done in constant time for some considered neighborhoods. Thus, we speed up the running time of exploring these neighborhoods. Now, let $T = (R_1, R_2, \ldots, R_k)$ be a tour with k routes, we then introduce a novel neighborhoods' structure and complexity of its exploration.

For Intro-route: Intro-route is used to optimize on a single route. Assume that R and m ($m < n$) are a route and its length, respectively. We then introduce eight neighborhoods' structure in turn.

Remove-insert neighborhood considers each vertex v_i in the route at the end of the route. This neighborhood of R is defined as a set $N_1(R) = \{R_i = (v_1, v_2, \ldots, v_{i-1}, v_{i+1}, \ldots, v_m, v_i) : i = 2, 3, \ldots, m-1\}$. Obviously, the size of $N_1(R)$ is $O(m)$.

Property 1. The time complexity of exploring $N_1(R)$ is $O(m^2)$.

Proof. Let us consider an initial solution $R = v_1, v_2, \ldots, v_{i-1}, v_i, v_{i+1}, \ldots, v_m$. The neighborhood generates a neighboring solution $R_i = v_1, v_2, \ldots, v_{i-1}, v_{i+1}, \ldots, v_m, v_i$. The costs of R and R_i are calculated as follows:

$$\begin{aligned} L(R) = (m-1)c(v_1, v_2) + \ldots + (m-i+1)c(v_{i-1}, v_i) + (m-i)c(v_i, v_{i+1}) \\ + (m-i-1)c(v_{i+1}, v_{i+2}) + \ldots + c(v_{m-1}, v_m) \end{aligned} \tag{1}$$

and

$$\begin{aligned} L(R_i) = (m-1)c(v_1, v_2) + \ldots + (m-i+1)c(v_{i-1}, v_{i+1}) + (m-i)c(v_{i+1}, v_{i+2}) \\ + \ldots + 2c(v_{m-1}, v_m) + c(v_m, v_i). \end{aligned}$$

Thus,

$$\begin{aligned} L(R_i) = L(R) - \sum_{k=i-1}^{m-1} (m-k)c(v_k, v_{k+1}) + (m-i+1)c(v_{i-1}, v_{i+1}) \\ + \sum_{k=i+1}^{m-1} (m-k+1)c(v_k, v_{k+1}) + c(v_m, v_i). \end{aligned} \tag{2}$$

It takes $O(m)$ time to calculate the formulation in (2). Therefore, the time complexity of exploring $N_1(R)$ is $O(m^2)$. $\square$

Swap adjacent neighborhood attempts to swap each pair of adjacent vertices in the route. This neighborhood of R is defined as a set $N_2(R) = \{R_i = (v_1, v_2, \ldots, v_{i-2}, v_i, v_{i-1}, v_{i+1}, \ldots, v_m) : i = 3, 4, \ldots, m-1\}$. The size of the neighborhood is $O(m)$.

Property 2. The time complexity of exploring $N_2(R)$ is $O(m)$.

Proof. The initial tour R and $L(R)$ are the same as in (1). The neighborhood generates a neighboring tour $R_i = v_1, v_2, \ldots, v_{i-2}, v_i, v_{i-1}, v_{i+1}, \ldots, v_m$. The latency of R_i is calculated as follows:

$$\begin{aligned}L(R_i) = {} & (n-1)c(v_1, v_2) + \ldots + (m-i+2)c(v_{i-2}, v_i) + (m-i+1)c(v_i, v_{i-1}) \\ & + (m-i)c(v_{i-1}, v_{i+1}) + (m-i-1)c(v_{i+1}, v_{i+2}) + \ldots + c(v_{m-1}, v_n).\end{aligned}$$

We have

$$\begin{aligned}L(R_i) = {} & L(R) - (m-i+2)c(v_{i-2}, v_{i-1}) - (m-i)c(v_i, v_{i+1}) \\ & + (m-i+2)c(v_{i-2}, v_i) + (m-i)c(v_{i-1}, v_{i+1}). \qquad (3)\end{aligned}$$

It is obvious that we can calculate $L(R_i)$ by the formulation (3) in $O(1)$ time. Therefore, the complexity of exploring $N_2(R)$ is $O(m)$. $\square$

Swap neighborhood attempts to swap the positions of each pair of vertices in the route. This neighborhood of R is defined as a set $N_3(R) = \{R_{ij} = (v_1, v_2, \ldots, v_{i-1}, v_j, v_{i+1}, \ldots, v_{j-1}, v_i, v_{j+1}, \ldots, v_m) : i = 2, 3, \ldots, m-3; j = i+3, \ldots, m\}$. The size of the neighborhood is $O(m^2)$.

Property 3. The complexity of exploring $N_3(R)$ is $O(m^2)$.

Proof. Initially, we have a solution $R = v_1, v_2, \ldots, v_{i-1}, v_i, v_{i+1}, \ldots, v_{j-1}, v_j, v_{j+1}, \ldots, v_m$ $(i+2 < j)$. Swap generates a neighboring solution $R_{ij} = v_1, v_2, \ldots, v_{i-1}, v_j, v_{i+1}, \ldots, v_{j-1}, v_i, v_{j+1}, \ldots, v_m$. The costs of R and R_i are calculated as follows:

$$\begin{aligned}L(R) = {} & (m-1)c(v_1, v_2) + \ldots + (m-i+1)c(v_{i-1}, v_i) + (m-i)c(v_i, v_{i+1}) \\ & + \ldots + (m-j+1)c(v_{j-1}, v_j) + (m-j)c(v_j, v_{j+1}) \\ & + \ldots + c(v_{m-1}, v_m). \qquad (4)\end{aligned}$$

$$\begin{aligned}L(R_{ij}) = {} & (m-1)c(v_1, v_2) + \ldots + (m-i+1)c(v_{i-1}, v_j) + (m-i)c(v_j, v_{i+1}) \\ & + \ldots + (m-j+1)c(v_{j-1}, v_i) + (m-j)c(v_i, v_{j+1}) + \ldots + c(v_{m-1}, v_m).\end{aligned}$$

It follows that

$$
\begin{aligned}
L(R_{ij}) = L(R) &- (m-i+1)c(v_{i-1}, v_i) - (m-i)c(v_i, v_{i+1}) \\
&- (m-j+1)c(v_{j-1}, v_j) - (m-j)c(v_j, v_{j+1}) + (m-i+1)c(v_{i-1}, v_j) \\
&+ (m-i)c(v_j, v_{i+1}) + (m-j+1)c(v_{j-1}, v_i) + (m-j)c(v_i, v_{j+1}). \quad (5)
\end{aligned}
$$

Hence, we can calculate $L(R_{ij})$ by the formulation (5) in $O(1)$ time. Therefore, the complexity of exploring $L(R_{ij})$ is $O(m^2)$. □

Or neighborhood attempts to reallocate three adjacent vertices to another position of the route. This neighborhood of R is defined as a set $N_4(R) = \{R_i = (v_1, v_2, \ldots, v_{i-1}, v_i, v_{j+1}, \ldots, v_k, v_{i+1}, \ldots, v_j, v_{k+1}, \ldots, v_m) : i = 2, 3, \ldots, m-5, j = 4, \ldots, m-3, k = 6, \ldots, m-1\}$. The size of the neighborhood is $O(m^3)$.

Property 4. The complexity of exploring $N_4(R)$ is $O(m^3)$.

Proof. In fact, this neighborhood implements the swap neighborhoods twice (we exchange between v_{j+1} and v_{i+1} and between v_j and v_k) in the route. It takes $O(1)$ time for swap operation. Therefore, calculating $L(R_{ijk})$ is $2 \times O(1)$ time and the complexity of exploring $L(R_{ijk})$ is $O(m^3)$. □

2-opt neighborhood removes each pair of edges from the solution and reconnects the vertices. This neighborhood of T is defined as a set $N_5(T) = \{T_{ij} = (v_1, v_2, \ldots, v_i, v_j, v_{j-1}, \ldots, v_{i+2}, v_{i+1}, v_{j+1}, \ldots, v_m) : i = 1, \ldots, n-4; j = i+4, \ldots, m\}$. The size of the neighborhood is $O(m^2)$.

Property 5. The complexity of exploring $N_5(T)$ is $O(m^3)$.

Proof. The initial tour and $L(T)$ are the same as in (4). The neighborhood generates a neighboring tour $T_{ij} = (v_1, v_2, \ldots, v_i, v_j, v_{j-1}, \ldots, v_{i+2}, v_{i+1}, v_{j+1}, \ldots, v_m)$. The costs of T and T_i are calculated as follows:

$$
\begin{aligned}
L(T_{ij}) = (m-1)c(v_1, v_2) + \ldots &+ (m-i)c(v_i, v_j) + (m-i-1)c(v_j, v_{i+2}) \\
&+ \ldots + (m-j+1)c(v_{j-1}, v_{i+1}) + (m-j)c(v_{i+1}, v_{j+1}) + \\
&+ \ldots + (v_{m-1}, v_m).
\end{aligned}
$$

We have

$$
\begin{aligned}
L(T_{ij}) = L(T) &- (m-i)c(v_i, v_{i+1}) - \sum_{h=1}^{j-i-1}(m-i-h)c(v_{i+h}, v_{i+h+1}) \\
&- (m-j)c(v_j, v_{j+1}) + (m-i)c(v_i, v_j) \\
&+ \sum_{h=1}^{j-i-1}(m-i-h)c(v_{j-h+1}, v_{j-h}) + (m-j)c(v_{i+1}, v_{j+1}). \qquad (6)
\end{aligned}
$$

It is obvious that we can calculate $L(T_{ij})$ by the formulation (6) in $O(m)$ time. Therefore, the complexity of exploring $N_5(T)$ is $O(m^3)$. □

Move-forward-k-vertices neighborhood of T is defined as a set $N_6(T) = \{T_{ijk} = (v_1, v_2, \dots, v_i, v_{i+k+1}, v_{i+k+2}, \dots, v_j, v_{i+1}, v_{i+2} \dots, v_{i+k}, v_{j+1}, \dots, v_m) : i = 1, 2, \dots, m-k-1; i+k < j \leqslant m\}$ with $k = 2, \dots, l$. The size of the neighborhood is $O(m^2)$.

Move-backward-k-vertices neighborhood of T is defined as a set $N_7(T) = \{T_{ijk} = (v_1, v_2, \dots, v_i, v_{i+k+1}, v_{i+k+2}, \dots, v_{i+1}, v_{i+2} \dots, v_{i+k}, v_j, v_{j+1}, \dots, v_m) : i = 1, 2, \dots, m-k-1; i+k < j \leqslant m\}$ with $k = 2, \dots, l$. The size of the neighborhood is $O(m^2)$.

Property 6. The complexity of exploring $N_6(R)$ and $N_7(R)$ is $O(m^3)$.

Proof. We prove Property 6 for move-forward-k-vertices and the same argument holds for move-backward-k-vertices. For a tour $R_{ijl} \in N_6(R)$, it can be shown that

$$
\begin{aligned}
L(T_{ijk}) = L(T) &- (m-i)c(v_i, v_{i+1}) - \sum_{h=i+1}^{j-1}(m-h)c(v_h, v_{h+1}) \\
&- (m-i)c(v_i, v_{i+1}) + (m-i)c(v_i, v_{i+k+1}) \\
&+ \sum_{h=1}^{j-i-k-2}(m-i-h)c(v_{i+k+h}, v_{i+k+h+1}) + (m-j+k+1)c(v_{j-1}, v_j) \\
&+ (m-j+k)c(v_j, v_{i+1}) + \sum_{h=1}^{k-1}(m-j+k-h)c(v_{i+h}, v_{i+h+1}) \\
&+ (m-j+1)c(v_{i+k}, v_{j+1}). \qquad (7)
\end{aligned}
$$

It is obvious that we can calculate $L(T_{ijk})$ by the formulation (7) in $O(m)$ time. Therefore, the complexity of exploring $N_6(T)$ is $O(m^3)$. □

It is realized that the calculation of a neighboring solution cost by using the known cost of the current solution in (6) and (7) cannot be done in constant time. As a result, the algorithm spends $O(m^3)$ operations for a full neighborhood search. However, Silva et al. [31] suggest a move evaluation procedure, which only requires $O(1)$ amortized operations since the number of edge exchanges is bounded. In this work, we use their evaluation procedure for 2-opt and move forward(backward)-k-vertices. Therefore, the time complexity of exploring all neighborhoods in the worst case is performed in $O(m^3)$.

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

The swap-2-routes neighborhood tries to exchange the positions of each pair of vertices in R_l and R_h in turn. The swap-2-route neighborhood of R_l and R_h is defined as a set $N_8(T) = \{T_i = (R_1, \ldots, R_2, \ldots, R_l = (v_{1l}, v_{2l}, \ldots, v_{ih}, v_{il+1}, \ldots, v_{ml}), \ldots, R_h = (v_{1h}, v_{2h}, \ldots, v_{il}, v_{ih+1}, \ldots, v_{mh}), \ldots, R_k) : il = 2, 3, \ldots, ml-1, ih = 2, 3, \ldots, mh-1\}$. The size of the neighborhood is $O(ml \times mh)$.

Property 7. The complexity of exploring $N_8(T)$ is $O(ml \times mh)$.

Proof. We have an initial tour T and its two routes are $R_l = (v_{1l}, v_{2l}, \ldots, v_{il}, v_{il+1}, v_{il+2} \ldots, v_{ml})$ and $R_h = (v_{1h}, v_{2h}, \ldots, v_{ih}, v_{ih+1}, v_{ih+2}, \ldots, v_{hl})$. The neighborhood generates a neighboring tour $T_i = (R_1, \ldots, R_2, \ldots, R_l^{'} = (v_{1l}, v_{2l}, \ldots, v_{ih}, v_{il+1}, \ldots, v_{ml}), \ldots, R_h^{'} = (v_{1h}, v_{2h}, \ldots, v_{il}, v_{ih+1}, \ldots, v_{mh}), \ldots, R_k)$. The costs of R_l, and R_h are calculated as follows:

$$
\begin{aligned}
L(R_l) &= (ml-1)c(v_{1l}, v_{2l}) + \ldots + (ml-i+1)c(v_{il-1}, v_{il}) + (ml-i)c(v_{il}, v_{il+1}) \\
&\quad + (ml-i-1)c(v_{il+1}, v_{il+2}) + \ldots + c(v_{ml-1}, v_{ml}), \\
L^{'}(R_l) &= (ml-1)c(v_{1l}, v_{2l}) + \ldots + (ml-i+1)c(v_{il-1}, v_{ih}) + (ml-i)c(v_{ih}, v_{il+1}) \\
&\quad + (ml-i-1)c(v_{il+1}, v_{il+2}) + \ldots + c(v_{ml-1}, v_{ml}), \qquad (8)
\end{aligned}
$$

$$
\begin{aligned}
L(R_h) &= (mh-1)c(v_{1h}, v_{2h}) + \ldots + (mh-i+1)c(v_{ih-1}, v_{ih}) \\
&\quad + (mh-i)c(v_{ih}, v_{ih+1}) + (mh-i-1)c(v_{ih+1}, v_{ih+2}) \\
&\quad + \ldots + c(v_{mh-1}, v_{mh}), \qquad (9) \\
L^{'}(R_h) &= (mh-1)c(v_{1h}, v_{2h}) + \ldots + (mh-i+1)c(v_{ih-1}, v_{il}) \\
&\quad + (mh-i)c(v_{il}, v_{ih+1}) - (mh-i-1)c(v_{ih+1}, v_{ih+2}) \\
&\quad + \ldots + c(v_{mh-1}, v_{mh}).
\end{aligned}
$$

Therefore,

$$\begin{aligned} L(T_i) = L(T) &- (ml - i + 1)c(v_{il-1}, v_{il}) - (ml - i)c(v_{il}, v_{il+1}) \\ &- (mh - i - 1)c(v_{ih-1}, v_{ih}) - (mh - i)c(v_{ih}, v_{ih+1}) \\ &+ (ml - i + 1)c(v_{il-1}, v_{ih}) + (ml - i)c(v_{ih}, v_{il+1}) \\ &+ (mh - i + 1)c(v_{ih-1}, v_{il}) + (mh - i)c(v_{il}, v_{ih+1}). \end{aligned} \tag{10}$$

□

Hence, we can calculate $L(T_i)$ by the formulation (10) in $O(1)$ time. The complexity of exploring $N_8(T)$ is $O(ml \times mh)$.

The insert-2-routes neighborhood considers each vertex v_i in R_l and inserts it into each position in R_h. Insert-2-route neighborhood of R_l and R_h is defined as a set $N_11(T) = \{T_i = (R_1, \ldots, R_2, \ldots, R_l = (v_{1l}, v_{2l}, \ldots, v_{ih-1}, v_{ih}, v_{il+1}, \ldots, v_{ml}), \ldots, R_h = (v_{1h}, v_{2h}, \ldots, v_{ih-1}, v_{ih+1}, \ldots, v_{mh}), \ldots, R_k) : il = 2, 3, \ldots, ml - 1, ih = 2, 3, \ldots, mh - 1\}$. The size of the neighborhood is $O(ml \times mh)$.

Property 8. The complexity of exploring $N_9(T)$ is $O(ml \times mh)$.

Proof. The initial tour and $L(R)$ are the same as in (8) and (9). The neighborhood generates a neighboring tour $T_i = (R_1, \ldots, R_2, \ldots, R_l' = (v_{1l}, v_{2l}, \ldots, v_{ih}, v_{il}, v_{il+1}, \ldots, v_{ml}), \ldots, R_h' = (v_{1h}, v_{2h}, \ldots, v_{ih-1}, v_{ih+1}, \ldots, v_{mh}), \ldots, R_k)$. The costs of R_l', R_h' are calculated as follows:

$$\begin{aligned} L'(R_l) &= (ml - 1)c(v_{1l}, v_{2l}) + \ldots + (ml - i + 1)c(v_{il-1}, v_{il}) + (ml - i)c(v_{il}, v_{ih}) \\ &\quad + (ml - i - 1)c(v_{ih}, v_{il+1}) + (ml - i - 2)c(v_{il+1}, v_{il+2}) \\ &\quad + \ldots + c(v_{ml-1}, v_{ml}), \\ L'(R_h) &= (mh - 1)c(v_{1h}, v_{2h}) + \ldots + (mh - i + 1)c(v_{ih-1}, v_{ih+1}) \\ &\quad + (mh - i)c(v_{ih+1}, v_{ih+2}) + \ldots + c(v_{mh-1}, v_{mh}). \end{aligned}$$

Therefore,

$$\begin{aligned} L(T_i) = L(T) &- (ml - i)c(v_{il}, v_{il+1}) - (mh - i + 1)c(v_{ih-1}, v_{ih}) \\ &- (mh - i)c(v_{ih}, v_{ih+1}) - \sum_{k=1h}^{ih-1} c(v_k, v_{k+1}) \\ &+ (ml - i)c(v_{il}, v_{ih}) + (mh - i - 1)c(v_{ih}, v_{il+1}) \\ &+ (mh - i + 1)c(v_{ih-1}, v_{ih+1}) + \sum_{k=1l}^{il-1} c(v_k, v_{k+1}). \end{aligned} \tag{11}$$

□

Hence, we can calculate $L(T_i)$ by the formulation (11) in $O(max(mh, ml))$ time. Therefore, the complexity of exploring $N_9(T)$ is $O(max(mh, ml) \times ml \times mh)$.

In each iteration, the best neighboring solution is accepted if it is non-tabu and improving, or tabu, but globally improving. Due to the use of different neighborhood structures, three tabu lists are built. A move of the type remove-insert, swap-adjacent, or move-up(down) is stored in the first tabu list, the second is for 2-opt and 2-edge-opt moves, and the last one is for swap-2-routes, insert-2-routes. We do not use tabu list for move-forward-k-vertices, and move-backward-k-vertices.

Step 3: After finding a local optimum in Step 2, Step 3 starts to build up a promising solution LT. When the objective value of any local optimum lies within 5–10 % of the best-found solution, it is added into LT. If the size of LT is equal to sTL, then the algorithm goes to Step 4. The size of LT is chosen to be five because a small value for the size of LT enhances more implementations to the intensification and diversification steps. However, the search can be moved to another area of the solution space without the previous area explored. Otherwise, if the value of sTL is large, less intensification and diversification steps are performed.

Step 4: If the promising solution list LT is full, an intensification step starts. Each solution of LT is returned to Step 2 without any tabu move. When a new local optimum is found, the algorithm goes to Step 5 in which a diverse solution to reinitialize the search is created.

Step 5: We update an attribute matrix M, whose entries represent the number of times edge (i, j) occurred in an element of the promising solutions list. The Shaking procedure in Algorithm 3 will use the matrix; hence allows guiding the search towards an unexplored part of the solution space. In this work, two shaking mechanisms can be used to give a new solution as follows:

1. In intro-route, we use the shaking mechanism in Algorithm 4, called double-bridge, originally developed by [26]. The structure of the double-bridge move derives from a special 4-opt neighborhood where edges added and dropped need not be successively adjacent. This mechanism can also be seen as a permutation of two disjoint segments of a route.
2. In intra-route, we randomly choose two routes and after that, exchange some vertices in them or insert some vertices from a route into another. The steps in the intra-route are described in Algorithm 5. This solution obtained from shaking procedures does not include the edges which appear more than l times in the M matrix. We finally return to Step 2 with this solution.

The last aspect to discuss is the stop criterium of the VNS+TS algorithm. A balance must be made between computation time and efficiency. Here, the algorithm stops if no improvement is found after the number of the loop (NL).

The running time of the VNS + TS algorithm is mostly during the VNS step. In that step, insert-2-routes neighborhood consumes time at least as the others do. Assume that if these neighborhoods are invoked k_1 times, then the complexity of neighborhoods' exploration is $O(k_1 \times max(mh, ml) \times ml \times mh) \sim O(k_1 \times n^3)$ (in the worst case the size of mh or ml is n). It is the theoretical complexity of our algorithm.

4 COMPUTATIONAL EVALUATIONS

4.1 Metrics

In order to evaluate the efficiency of a metaheuristic algorithm, we can compare its solution to

1. the optimal solution (OPT);
2. the lower bound (LB); and
3. the initial solution of the construction phase (*Init.Sol*) or a good upper bound of the state-of-the-art metaheuristic algorithm (UB).

We define the improvement of the algorithm concerning *Best.Sol*, when *Best.Sol* is the best solution found by our algorithm, in comparison with the optimal solution ($Gap_1[\%]$), a lower bound ($Gap_2[\%]$), and an initial solution ($Improv[\%]$) in percent, respectively, as follows:

$$Gap_1[\%] = \frac{Best.Sol - OPT}{OPT} \times 100\,\%,$$

$$Gap_2[\%] = \frac{Best.Sol - LB}{LB} \times 100\,\%,$$

$$Improv[\%] = \frac{Best.Sol - Init.Sol}{Init.Sol} \times 100\,\%.$$

The exact algorithm can find optimal solutions as in [25]. However, the algorithms only solve the problems with small sizes. The optimal solutions have been unknown with large instance sizes. In such cases, our best solutions can be compared to the tight lower bounds (i.e., defined by Nucamendi-Guillén et al. in [29]) or the initial solutions (i.e., the output of the insertion heuristic). As mentioned ealier, the MTRP is a relaxation of the MTRPD since $MD = 0$. Therefore, we can consider the optimal solutions published by Nucamendi-Guillén et al. [29] for the MTRP as the tight lower bounds in our experiments.

4.2 Datasets

The experimental data includes two datasets. In all instances, every distance between vertices satisfies the triangle inequality. Each instance contains the coordinate of n vertices and one vertex was arbitrary designated as the depot. We divide these instances into two types (i.e., type 1 and type 2). The former one consists of the instances in which the optimal solutions have been known, otherwise the other depends on type 2.

We inherit several small instances in [25] and name them dataset 1 in our experiments. As a result, we can obtain the optimal solutions for these instances by using the exact algorithm in [25]. The dataset includes six TSP instances from the TSPLIB such as brd14051, d15112, d18512, fnl4461, nrw1379, and pr1002. For each TSP instance, they generate ten MTRPD instances by randomly selecting ten subsets of n vertices, where $n = 30$, 40 and 50. Therefore, in total, fifty MTRPD instances are used in our experiment.

The numerical analysis was performed on a set of benchmark problems for Capacitated VRP in [34]. As testing the proposed algorithm on all instances would be computationally too expensive, we applied our numerical analysis on some selected instances. First, to eliminate the effects of size, problems with approximately 50 up to 561 customers are chosen. Moreover, in order not to bias the results by taking "easy" or "hard" instances we randomly select them. We put them into a group named dataset 2. These are:

1. Christofides et al.: This dataset includes seven instances (CMT6, CMT7, ..., CMT14), which vary the number of vertices from 50 to 200 and vehicles from 5 to 18;
2. Taillard et al.: Nine instances from 75 to 150 vertices are picked randomly, specifically: tai75a, tai75b, tai75c, tai100a, tai100b, tai100c, tai150a, tai150b, tai150c;
3. Augerat et al.: Fifteen instances of dataset P and E are selected, which vary the number of vertices from 30 to 76 and vehicles from 2 to 15. In this dataset, we can obtain the lower bounds of the optimal solutions for the instances in [29];
4. Golden et al.: Six larger instances are picked randomly from G1 to G8, which vary the number of vertices from 240 to 480 and vehicles from 5 to 10;
5. Nucamendi-Guillén et al.: One hundred and fifty instances from 60 to 80 vertices are used in our experiments. The optimal solutions for the instances can be extracted from [29].

Moreover, our algorithm is also tested with some TRP instances. These are:

6. Silva et al. [31]: Three of these sets are generated, where each of them is composed of 20 instances with 50, 100, and 200 customers, respectively;
7. Abeledo et al. [2]: Nine instances from 48 to 100 vertices are chosen. The optimal solutions for these instances are extracted from [2].

In all instances in dataset 1, and several instances in dataset 2 (such as Christofides et al.'s and Golden et al.'s instances), the maximum total distance traveled in a route is available. However, in the others, there does not exist the distance constraint. For each of these instances, we generated three possible distance constraints as a function of the distance to the farthest vertice from the depot (d_{max}). The distance constraint gets the values $2 \times d_{max}, 2.5 \times d_{max}$, and $3 \times d_{max}$. The similar generation for travel distance limit can be found in [12, 13, 25].

4.3 Results and Discussion

We conducted the experiments on a personal computer, which is equipped with an Intel Pentium Core i7 duo 2.10 GHz CPU and 4 GB RAM.

We experimented with the above datasets. For the instances in dataset 1, their optimal solutions let us evaluate precisely the efficiency of the TS + VNS algorithm. For the instances in dataset 2, because their optimal solutions have been unknown, our solutions only compare to the upper bounds or the known best solutions instead of the optimal ones. Therefore, the TS+VNS algorithm's efficiency is only evaluated relatively.

Through preliminary experiments, we observed that the values $pos = 5$, $sLT = 5$, $l = 5$, and $NL = 50$ resulted in a good trade-off between solution quality and run time. In addition, in a pilot study, the performance of the algorithm relatively depends on the order in which the neighborhoods are used. Generally speaking, the neighborhoods which have a smaller size are explored first. Since the algorithm becomes stuck in local optimum, the larger neighborhoods are used. That is, larger sized neighborhoods may help escape from local optimum. In this paper, the order of the neighborhoods is as follows: swap adjacent, remove-insert, swap, 2-opt, or, swap-2-route, and insert-2-route.

For each instance, our algorithm runs ten times, and the results are shown in Tables 1–18. In all the tables, we denote *Best.Sol* and *Aver.Sol* as the best and average solution of our metaheuristic, respectively. Table 1 includes the comparison between our algorithm and the optimal solutions in [25]. Table 2 shows the average values of Table 1. In Tables 3–11, we compare the results of the algorithm with the lower bound of the optimal solution. In these cases, the optimal solution of MTRP is the lower bound of the optimal solution of MTRPD. The optimal solution of MTRP is obtained in [29]. Moreover, they are also compared with the initial solutions by using insertion heuristic. Table 12 illustrates the evolution of the average deviation to the initial solutions during the iterations in some instances. In Tables 13–18, we show the results of our algorithm against the state-of-the-art metaheuristic algorithms in several MTRPD variants. Let T be the running time in seconds for our metaheuristic. *cTime* represents scaled run times, which is estimated on a Pentium 4, 2.4 GHz by means of the factors of Dongarra in [14] by second (note that: The experiments of Ezzineet et al. (IOE) [15], Ke et al. (CCVRP) [23], Ngueveu (MA1) [28], Nucamendi-Guillén et al. (SNG) [29], Riberio et al. (ALNS) [30], Silva et al. (MS) [31], and Ban (GRASP + VND) [9] were implemented on Pentium 4, 1 GHz, Pentium 4, 2.4 GHz, Pentium 4, 2 GHz, Intel Core 2 Duo 3 GHz, Pentium 4, 2 GHz, Pentium 4, 2.4 GHz, Pentium core i7 2.93 GHz, and Pentium core i7 duo 2.10 GHz, respectively).

4.3.1 Experimental Results for Datasets in Type 1

The experimental results are illustrated in Table 2, which are the average values calculated from Table 1. In Table 2, we denote $\overline{Gap_1}$ and $\overline{T}$ as the average values of Gap_1 and T for each dataset, respectively.

Table 2 shows that the algorithm is capable of finding the optimal solutions for all instances in dataset 1 in a reasonable amount of time, even for the cases of 50 vertices. That means our solutions are better than the ones in our previous work [9], which fails to find the optimal solutions for all instances with 50 vertices.

Instances		*n*=30, *k*=6				*n*=40, *k*=8				*n*=50, *k*=10			
		OPT	*Best.Sol*	*Aver.Sol*	*T*	*OPT*	*Best.Sol*	*Aver.Sol*	*T*	*OPT*	*Best.Sol*	*Aver.Sol*	*T*
pr1002	0	168,188.80	168,188.80	272,908.40	0.19	233,387.20	233,387.20	272,908.40	0.38	272,908.40	272,908.40	273754.69	0.72
	1	195,805.80	195,805.80	249,275.40	0.18	218,781.40	218,781.40	249,275.40	0.38	249,275.40	249,275.40	250561.75	0.73
	2	182,635.00	182,635.00	277,959.10	0.20	211,241.70	211,241.70	277,959.10	0.40	277,959.10	277,959.10	278260.69	0.69
	3	139,784.70	139,784.70	298,846.10	0.18	196,120.40	196,120.40	298,846.10	0.42	298,846.10	298,846.10	299368.26	0.72
	4	164,916.70	164,916.70	262,518.40	0.17	227,450.10	227,450.10	262,518.40	0.50	262,518.40	262,518.40	263424.19	0.68
	5	163,642.30	163,642.30	273,318.80	0.18	194,802.20	194,802.20	273,318.80	0.43	273,318.80	273,318.80	275714.50	0.68
	6	160,585.90	160,585.90	280,317.30	0.19	229,730.10	229,730.10	280,317.30	0.48	280,317.30	280,317.30	281015.54	0.72
	7	166,887.00	166,887.00	246,341.00	0.20	236,896.10	236,896.10	246,341.00	0.39	246,341.00	246,341.00	247693.88	0.73
	8	161,025.80	161,025.80	256,971.40	0.18	230,126.60	230,126.60	256,971.40	0.47	256,971.40	256,971.40	257688.31	0.74
	9	144,167.00	144,167.00	267,596.70	0.18	192,179.90	192,179.90	267,596.70	0.48	267,596.70	267,596.70	268104.67	0.75
brd14051	0	97,380.30	97,380.30	133,642.30	0.19	97,630.70	97,630.70	133,642.30	0.43	133,642.30	133,642.30	134565.72	0.68
	1	96,322.50	96,322.50	123,212.70	0.18	110,671.10	110,671.10	123,212.70	0.42	123,212.70	123,212.70	123459.68	0.69
	2	64,109.40	64,109.40	137,175.10	0.19	127,629.70	127,629.70	137,175.10	0.46	137,175.10	137,175.10	137550.65	0.75
	3	89,582.50	89,582.50	150,209.80	0.18	99,527.60	99,527.60	150,209.80	0.37	150,209.80	150,209.80	151117.47	0.75
	4	87,615.70	87,615.70	116,278.70	0.19	123,881.80	123,881.80	116,278.70	0.49	116,278.70	116,278.70	116771.48	0.75
	5	75,079.50	75,079.50	124,648.20	0.19	98,329.40	98,329.40	124,648.20	0.49	124,648.20	124,648.20	124860.29	0.69
	6	94,540.80	94,540.80	121,190.40	0.20	110,676.60	110,676.60	121,190.40	0.36	121,190.40	121,190.40	122040.23	0.72
	7	81,515.80	81,515.80	124,077.60	0.19	103,775.50	103,775.50	124,077.60	0.42	124,077.60	124,077.60	124459.33	0.68
	8	74,160.80	74,160.80	125,446.00	0.19	101,387.30	101,387.30	125,446.00	0.41	125,446.00	125,446.00	125750.98	0.72
	9	90,628.10	90,628.10	118,925.00	0.18	87,945.10	87,945.10	118,925.00	0.48	118,925.00	118,925.00	119701.47	0.75
fnl4461	0	51,192.20	51,192.20	81,562.20	0.19	63,096.00	63,096.00	81,562.20	0.48	81,562.20	81,562.20	81931.39	0.68
	1	44,154.70	44,154.70	79,804.80	0.18	66,882.60	66,882.60	79,804.80	0.38	79,804.80	79,804.80	80132.27	0.71
	2	46,571.20	46,571.20	73,309.10	0.18	70,151.40	70,151.40	73,309.10	0.39	73,309.10	73,309.10	73657.61	0.69
	3	48,591.40	48,591.40	79,335.10	0.19	58,843.90	58,843.90	79,335.10	0.43	79,335.10	79,335.10	79480.48	0.67
	4	54,485.90	54,485.90	75,052.00	0.18	61,654.90	61,654.90	75,052.00	0.38	75,052.00	75,052.00	75154.18	0.70
	5	47,907.30	47,907.30	76,738.10	0.18	56,144.50	56,144.50	76,738.10	0.42	76,738.10	76,738.10	77076.34	0.75
	6	45,882.10	45,882.10	75,268.90	0.18	61,274.90	61,274.90	75,268.90	0.43	75,268.90	75,268.90	75553.79	0.70
	7	44,545.30	44,545.30	72,956.30	0.20	65,698.30	65,698.30	72,956.30	0.38	72,956.30	72,956.30	73058.48	0.72
	8	50,365.30	50,365.30	70,244.00	0.20	64,260.90	64,260.90	70,244.00	0.48	70,244.00	70,244.00	70593.77	0.72
	9	49,179.60	49,179.60	82,157.00	0.18	58,717.50	58,717.50	82,157.00	0.49	82,157.00	82,157.00	82319.62	0.73
d15112	0	225,070.20	225,070.20	353,657.80	0.18	287,734.80	287,734.80	353,657.80	0.37	353,657.80	353,657.80	354976.61	0.73
	1	213,332.30	213,332.30	355,115.20	0.19	256,987.10	256,987.10	355,115.20	0.39	355,115.20	355,115.20	356258.02	0.72
	2	208,323.60	208,323.60	392,196.10	0.19	307,407.10	307,407.10	392,196.10	0.46	392,196.10	392,196.10	393694.14	0.75
	3	222,870.40	222,870.40	350,821.80	0.19	292,602.70	292,602.70	350,821.80	0.42	350,821.80	350,821.80	351422.23	0.72
	4	216,056.00	216,056.00	341,493.60	0.19	299,259.00	299,259.00	341,493.60	0.35	341,493.60	341,493.60	342212.93	0.74
	5	235,215.80	235,215.80	360,717.40	0.18	269,559.40	269,559.40	360,717.40	0.35	360,717.40	360,717.40	362220.31	0.72
	6	207,139.00	207,139.00	390,251.40	0.19	320,989.40	320,989.40	390,251.40	0.45	390,251.40	390,251.40	392014.44	0.72
	7	280,309.00	280,309.00	327,701.90	0.20	287,270.70	287,270.70	327,701.90	0.41	327,701.90	327,701.90	328900.79	0.73
	8	244,015.40	244,015.40	344,600.50	0.19	303,263.90	303,263.90	344,600.50	0.38	344,600.50	344,600.50	345172.84	0.68
	9	238,976.20	238,976.20	347,783.60	0.20	282,412.30	282,412.30	347,783.60	0.40	347,783.60	347,783.60	348249.82	0.72
nrw1379	0	31,249.40	31,249.40	39,206.10	0.17	35,655.20	35,655.20	39,206.10	0.45	39,206.10	39,206.10	39299.26	0.74
	1	33,138.50	33,138.50	61,449.00	0.17	33,000.70	33,000.70	61,449.00	0.39	61,449.00	61,449.00	44859.42	0.68
	2	31,872.00	31,872.00	45,914.20	0.19	39,928.10	39,928.10	45,914.20	0.49	45,914.20	45,914.20	46111.68	0.72
	3	31,777.10	31,777.10	46,208.00	0.18	36,685.40	36,685.40	46,208.00	0.50	46,208.00	46,208.00	46317.23	0.68
	4	26,671.20	26,671.20	43,557.90	0.18	36,168.60	36,168.60	43,557.90	0.38	43,557.90	43,557.90	43635.93	0.75
	5	29,010.30	29,010.30	46,718.40	0.18	38,005.40	38,005.40	46,718.40	0.42	46,718.40	46,718.40	46946.35	0.71
	6	30,398.10	30,398.10	49,421.10	0.19	31,837.30	31,837.30	49,421.10	0.36	49,421.10	49,421.10	49553.53	0.73
	7	30,765.50	30,765.50	49,960.10	0.19	39,394.80	39,394.80	49,960.10	0.49	49,960.10	49,960.10	50037.67	0.75
	8	28,796.40	28,796.40	41,560.90	0.20	36,674.50	36,674.50	41,560.90	0.37	41,560.90	41,560.90	41703.33	0.74
	9	26,271.20	26,271.20	44,404.00	0.18	36,447.70	36,447.70	44,404.00	0.43	44,404.00	44,404.00	44518.42	0.68

Table 1. Results for dataset type 1

4.3.2 Experimental Results for Datasets in Type 2

Similar to the dataset type 1, we show the average values in Tables 9 and 12. The values in Table 9 are calculated from the ones in Tables 3–8. Meanwhile, Table 12

Instances	n=30		n=40		n=50	
	Gap_1	T	Gap_1	T	Gap_1	T
brd14051-x	0.00	0.19	0.00	0.38	0.00	0.72
d15112-x	0.00	0.18	0.00	0.38	0.00	0.73
fnl4461-x	0.00	0.20	0.00	0.40	0.00	0.69
nrw1379-x	0.00	0.18	0.00	0.42	0.00	0.72
pr1002-x	0.00	0.17	0.00	0.50	0.00	0.68

Table 2. Average results for daset type 1

Instances	*LB*	*Init.Sol*	$MD = 2\times d_{max}$			$MD = 2.5\times d_{max}$			$MD = 3\times d_{max}$		
			Best.Sol	*Aver.Sol*	T	*Best.Sol*	*Aver.Sol*	T	*Best.Sol*	*Aver.Sol*	T
pr1002_60_0	530946.01	660211.35	532444.24	537533.19	1.49	531865.86	533825.5	1.45	531757.16	534609.7	1.45
pr1002_60_1	356469.79	455893.41	359952.84	370184.99	1.42	358368.98	367379.55	1.45	358942.18	367163.14	1.45
pr1002_60_2	344118.14	467498.53	344660.51	354297.21	1.49	344923.03	351734.69	1.42	344310.30	351678.69	1.42
pr1002_60_3	429604.2	579392.35	430930.61	435619.75	1.46	430738.16	433873.27	1.45	430341.65	434434.23	1.45
pr1002_60_4	435655.25	540342.11	436868.42	443304.34	1.41	437187.11	441177.4	1.46	436487.61	441214.38	1.46
pr1002_60_5	668129.73	779776.11	670146.51	674511.75	1.43	669780.58	673040.01	1.47	669092.64	673298.58	1.47
pr1002_60_6	406678.77	495022.53	408185.76	413479.93	1.45	407782.79	411889.62	1.44	407328.01	412153.54	1.44
pr1002_60_7	311254.73	414296.52	315012.06	325939.25	1.49	314351.03	322249.21	1.44	314631.61	323133.36	1.44
pr1002_60_8	469816.84	591638.26	471572.56	478597.47	1.50	470579.14	476756.73	1.50	471383.23	476823.63	1.50
pr1002_60_9	277336.06	377249.41	280140.52	292281.76	1.42	281572.42	288733.31	1.40	278504.55	288503.61	1.40
brd14051_60_0	213420.42	267899.2375	214319.39	216860.88	1.50	213818.46	215295.73	1.49	213614.02	215749.05	1.49
brd14051_60_1	218315.68	312468.7714	218728.14	220264.95	1.49	218880.78	220373.77	1.49	218556.31	220728.3	1.49
brd14051_60_2	151666.85	207799.2353	153198.75	156139.96	1.45	152543.04	154954.36	1.48	153057.91	154911.12	1.48
brd14051_60_3	172199.83	232433.3597	172875.34	175903.46	1.48	172877.53	174923.62	1.41	172882.12	174934.8	1.41
brd14051_60_4	133952.5	167792.608	135660.89	139384.34	1.42	134963.93	138301.02	1.42	135622.01	138319.39	1.42
brd14051_60_5	203145.14	290606.4286	203424.5	205266.4	1.44	203348.6	204742.68	1.43	203576.11	204746.58	1.43
brd14051_60_6	136233.51	171636.975	137309.58	140813.51	1.49	137072.2	139742.31	1.47	137666.96	139931.2	1.47
brd14051_60_7	171879.58	248180.3	173726.21	176795.14	1.48	173395.86	175603.68	1.42	172880.97	175530.15	1.42
brd14051_60_8	191580.79	241067.3882	192145.73	196064.3	1.49	191949.55	194840.17	1.47	192277.69	195207.18	1.47
brd14051_60_9	128178.58	174326.1925	129452.37	132113.8	1.47	129039.81	131038.32	1.41	128554.14	131136.74	1.41
fnl4461_60_0	156260.54	194032.1583	156482.1	157364.48	1.40	156502.53	156867.68	1.47	156508.88	157085.13	1.47
fnl4461_60_1	103190.13	131569.4961	103881.33	105989.96	1.48	103571.39	105059.35	1.45	103533.38	104978.45	1.45
fnl4461_60_2	109739.93	149525.6149	110236.87	111979.43	1.49	110112.9	111088.62	1.48	109795.94	111158.19	1.48
fnl4461_60_3	100792.2	136198.0575	101299.08	103234.98	1.47	100961.18	102382.31	1.47	101131.62	102305.81	1.47
fnl4461_60_4	149638.18	185947.0777	150338.84	151703.66	1.48	150322.42	151338.95	1.49	150154.33	151438.45	1.49
fnl4461_60_5	158679.44	185251.4379	159206.73	160478.43	1.48	158930.62	160072.49	1.49	158926.13	160089.54	1.49
fnl4461_60_6	122266.92	149102.6283	122947.07	124435.02	1.44	122916.9	123926.83	1.43	122817.55	123825.42	1.43
fnl4461_60_7	107469.11	142108.8532	108053.05	109887.69	1.47	107925.52	109376.06	1.47	107716.14	109006.11	1.47
fnl4461_60_8	100531.72	127280.3749	101450.39	104630.87	1.42	101592.06	103707.02	1.42	101118.58	103710.54	1.42
fnl4461_60_9	135829.76	183343.8156	136148.74	137640.52	1.47	136160.29	137330.96	1.40	136163.91	137267.96	1.40
d15112_60_0	684939.42	851498.8482	686712.6	699720.00	1.40	685359.09	694349.73	1.48	686192.80	694446.4	1.48
d15112_60_1	644759.99	819500.5493	647040.61	654258.07	1.43	646425.6	651802.15	1.45	645901.74	652152.95	1.45
d15112_60_2	425069.33	583381.5404	430094.57	444157.23	1.41	428827.48	439403.47	1.45	426883.39	438826.4	1.45
d15112_60_3	528177.95	662371.45	529897.16	541938.08	1.41	529082.69	539617.02	1.49	529129.18	538786.1	1.49
d15112_60_4	586915.82	736112.95	588890.36	596517.59	1.48	587802.34	593603.46	1.46	588157.01	593755.74	1.46
d15112_60_5	422195.61	494729.4263	425174.86	435267.63	1.47	423386.01	432465.17	1.46	422698.69	432637.93	1.46
d15112_60_6	518793.6	633637.8578	522485.21	534777.58	1.43	521841.6	530514.27	1.48	520984.95	529540.23	1.48
d15112_60_7	616918.44	776732.675	621386.14	631195.53	1.49	620561.63	627712.93	1.48	619844.40	627215.03	1.48
d15112_60_8	397619.37	500495.3875	400396.31	417380.61	1.40	400772.54	412773.75	1.46	400191.93	412712.49	1.46
d15112_60_9	673840.81	910298.6184	675975.95	682660.36	1.45	674759.18	681252.34	1.42	675686.08	680839.7	1.42
nrw1379_60_0	64359.77	80086.56654	64587.82	65752.75	1.44	64588.24	65216.27	1.42	64570.11	65388.53	1.42
nrw1379_60_1	83410.67	104646.3375	83717.07	84295.24	1.48	83453.87	84149.95	1.49	83577.60	84304.19	1.49
nrw1379_60_2	52858.87	70986.81333	53240.11	55622.46	1.48	53731.31	55606.98	1.40	53699.79	55545.65	1.40
nrw1379_60_3	62341.36	84434.20476	62799.04	64203.28	1.42	63054.46	64243.88	1.45	63155.92	64314.58	1.45
nrw1379_60_4	56012.13	69680.90881	56337.25	57630.46	1.45	56306.76	57462.91	1.42	56401.26	57331.24	1.42
nrw1379_60_5	58083.8	72973.325	58378.66	59960.51	1.45	58551.84	59986.25	1.50	58611.01	59945.96	1.50
nrw1379_60_6	52224.66	65749.025	52599.22	54626.86	1.46	52433.62	54526.7	1.47	52476.05	54450.03	1.47
nrw1379_60_7	58402.97	73290.6375	58632.51	59997.69	1.47	58495.04	60194.16	1.45	58679.87	59937.75	1.45
nrw1379_60_8	52145.08	66101.85821	52687.3	53873.21	1.48	52682.18	53653.94	1.45	52550.57	53726.75	1.45
nrw1379_60_9	49026.52	66572.84307	49436.13	51019.36	1.43	49471.03	51283.54	1.41	49385.68	50955.34	1.41

Table 3. Results for ANMM-instances ($n = 60$, $k = 12$)

Instances	*LB*	*Init.Sol*	*MD* = 2×d_{max}			*MD* = 2.5×d_{max}			*MD* = 3×d_{max}		
			Best.Sol	*Aver.Sol*	*T*	*Best.Sol*	*Aver.Sol*	*T*	*Best.Sol*	*Aver.Sol*	*T*
pr1002_70_0	429557.7	535485.69	432866.54	439866.58	1.56	432866.54	439866.58	1.86	432866.54	439866.58	1.86
pr1002_70_1	430048.06	530744.28	433424.51	440956.86	1.54	433424.51	440956.86	2.10	433424.51	440956.86	2.10
pr1002_70_2	377233.86	524943.18	379605.76	389076.55	1.34	379605.76	389076.55	1.95	379605.76	389076.55	1.95
pr1002_70_3	429562.01	557187.53	432804.3	441224.85	1.32	432804.3	441224.85	1.85	432804.3	441224.85	1.85
pr1002_70_4	435659.17	574628.71	439726.32	447473.63	1.48	439726.32	447473.63	2.17	439726.32	447473.63	2.17
pr1002_70_5	429584.16	558284.19	431998.56	443795.32	1.67	431998.56	443795.32	2.13	431998.56	443795.32	2.13
pr1002_70_6	344534.44	495280.87	348013.29	355996.4	1.41	348013.29	355996.4	1.95	348013.29	355996.4	1.95
pr1002_70_7	393558.46	543537.08	396618.63	406657.01	1.51	396618.63	406657.01	1.99	396618.63	406657.01	1.99
pr1002_70_8	397072.39	560744.46	400993.88	412836.03	1.37	400993.88	412836.03	1.97	400993.88	412836.03	1.97
pr1002_70_0	429557.7	535485.69	432866.54	439866.58	1.56	432866.54	439866.58	1.86	432866.54	439866.58	1.86
brd14051_70_0	191843.35	248655.93	193433.59	196300.43	1.38	193433.59	196300.43	1.82	192664.90	195906.35	1.82
brd14051_70_1	169340.01	227929.44	169848.29	172027.95	1.49	169848.29	172027.95	1.91	170141.57	171980.54	1.91
brd14051_70_2	216195.95	274964.88	217119.82	218723.61	1.56	217119.82	218723.61	1.78	216904.65	219570.63	1.78
brd14051_70_3	229328.9	287350.77	230828.7	235001.63	1.64	230828.7	235001.63	2.12	230700.27	234893.9	2.12
brd14051_70_4	302498.42	352533.21	303665.65	305149.45	1.67	303665.65	305149.45	1.75	302997.60	305008.65	1.75
brd14051_70_5	179470.31	239730.53	180336.91	182335.7	1.50	180336.91	182335.7	1.78	180402.62	182346.26	1.78
brd14051_70_6	231693.74	299183.67	232654.01	234908.12	1.33	232654.01	234908.12	1.75	232686.01	234963.25	1.75
brd14051_70_7	284960.31	354174.33	285658.8	288385.84	1.33	285658.8	288385.84	1.78	285974.24	288104.83	1.78
brd14051_70_8	167533.17	246354.08	168487.82	171604.89	1.38	168487.82	171604.89	1.89	169011.06	171829.52	1.89
brd14051_70_9	253499.74	304870.99	255311.77	259334.3	1.63	255311.77	259334.3	1.82	255500.92	259488.88	1.82
fnl4461_70_0	154805.67	195026.89	155064.12	156113.94	1.38	155064.12	156113.94	2.16	155064.12	156113.94	2.16
fnl4461_70_1	104585.82	138431.3	105739.55	108511.43	1.62	105739.55	108511.43	1.88	105739.55	108511.43	1.88
fnl4461_70_2	161892.44	202490.16	162356.87	163138.77	1.37	162356.87	163138.77	1.75	162356.87	163138.77	1.75
fnl4461_70_3	99122.23	136888.98	100079.14	101756.85	1.66	100079.14	101756.85	2.15	100079.14	101756.85	2.15
fnl4461_70_4	157106.13	215133.56	157517.01	158373.7	1.42	157517.01	158373.7	2.19	157517.01	158373.7	2.19
fnl4461_70_5	112094.64	154967.72	113643.87	116157.3	1.36	113643.87	116157.3	1.89	113643.87	116157.3	1.89
fnl4461_70_6	121521	163833.57	122307.91	124262.7	1.37	122307.91	124262.7	1.71	122307.91	124262.7	1.71
fnl4461_70_7	175859.51	219145.84	176440.38	177290.01	1.53	176440.38	177290.01	1.79	176440.38	177290.01	1.79
fnl4461_70_8	122141.15	168186.87	122884.09	125065.93	1.47	122884.09	125065.93	1.88	122884.09	125065.93	1.88
fnl4461_70_0	154805.67	195026.89	155064.12	156113.94	1.38	155064.12	156113.94	2.16	155064.12	156113.94	2.16
d15112_70_0	517426.18	692065.87	523553.26	531145.65	1.62	523553.26	531145.65	1.48	523553.26	531145.65	1.79
d15112_70_1	715678.26	886361.97	722362.06	728250.24	1.51	722362.06	728250.24	1.45	722362.06	728250.24	1.98
d15112_70_2	688605.9	892883.4	690001.05	695602.83	1.50	690001.05	695602.83	1.45	690001.05	695602.83	2.04
d15112_70_3	625623.9	852706.46	630340.44	637526.15	1.66	630340.44	637526.15	1.49	630340.44	637526.15	1.77
d15112_70_4	532088.98	747897.34	536030.13	544894.97	1.39	536030.13	544894.97	1.46	536030.13	544894.97	1.72
d15112_70_5	500455.25	639173.85	504012.97	516029.99	1.59	504012.97	516029.99	1.46	504012.97	516029.99	1.82
d15112_70_6	497229.6	708355.1	494630.23	501861.27	1.59	494630.23	501861.27	1.48	494630.23	501861.27	1.82
d15112_70_7	599776.85	766690.15	604254.21	609315.82	1.43	604254.21	609315.82	1.48	604254.21	609315.82	1.88
d15112_70_8	576957.51	734369.78	582333.88	587065.31	1.51	582333.88	587065.31	1.46	582333.88	587065.31	1.93
d15112_70_9	775176.3	1018784.7	776211.48	780571.39	1.30	776211.48	780571.39	1.42	776211.48	780571.39	1.70
nrw1379_70_0	66839.83	91823.05	67133.95	68149.96	1.29	67133.95	68149.96	1.42	67141.89	67842.3	1.79
nrw1379_70_1	65103.43	86177.5	65403.57	67281.39	1.49	65403.57	67281.39	1.49	65836.06	67162.39	2.09
nrw1379_70_2	63480.7	86971.23	64215.06	65872.58	1.60	64215.06	65872.58	1.40	63992.33	65884.47	1.67
nrw1379_70_3	59273.92	78578.67	60111.63	61528.06	1.66	60111.63	61528.06	1.45	59705.82	61585.84	2.16
nrw1379_70_4	70594.56	90450.43	71095.2	71700.76	1.33	71095.2	71700.76	1.42	70953.48	71755.91	2.05
nrw1379_70_5	73884.17	95059.75	74190.23	74969.23	1.51	74190.23	74969.23	1.50	74081.86	75045.23	1.92
nrw1379_70_6	64306.14	87586.7	65019.63	66399.38	1.47	65019.63	66399.38	1.47	64995.77	66682.88	1.97
nrw1379_70_7	90554.87	113522.25	90716.75	91348.07	1.27	90716.75	91348.07	1.45	90743.23	91407.05	1.78
nrw1379_70_8	91738.43	125234.93	91924.59	92556.51	1.41	91924.59	92556.51	1.45	91961.09	92591.73	1.90
nrw1379_70_9	68024.3	92003.32	68219.67	69163.47	1.34	68219.67	69163.47	1.41	68512.64	69234.01	2.18

Table 4. Results for ANMM-instances ($n = 70$, $k = 14$)

contains the average values calculated from Tables 10 to 11. In Tables 9 and 12, we denote $\overline{Gap_i}(i = 1, 2)$ and $\overline{T}$ as the average of $Gap_i(i = 1, 2)$ and T for each dataset, respectively.

From Tables 9 to 11, for all instances, it is indicated that the proposed algorithm can improve the solutions in comparison with the initial solutions. Specifically, the average of *Improv* lines between 19.4 % and 27.1 %. Besides, in Table 9, for most instances, our solutions fall into the range of 0.88 %–3.77 % of the lower bound of the optimal solution. Therefore, we can conclude that for the instances solved, our algorithm finds near-optimal solutions, even for the large instances. In comparison with GRASP + VND [9], our average Gap_2 (about 1.09) is much better than GRASP + VND (about 3.62). Obviously, the VNS + TS algorithm outperforms

Instances	LB	Init.Sol	$MD = 2\times d_{max}$			$MD = 2.5\times d_{max}$			$MD = 3\times d_{max}$		
			Best.Sol	Aver.Sol	T	Best.Sol	Aver.Sol	T	Best.Sol	Aver.Sol	T
pr1002_80_0	491764.64	656239.68	494687.68	504055.32	4.74	494687.82	501827.11	4.52	495216.69	502028.79	4.52
pr1002_80_1	442164.21	613287.46	446829.95	452810.93	4.59	446638.68	451662.89	4.77	446105.29	451219.62	4.77
pr1002_80_2	505524.17	609954.56	509224.22	516775.04	4.66	507827.3	514010.41	4.78	509460.12	513907.95	4.78
pr1002_80_3	436752.96	614611.29	439342.64	448311.86	4.55	439573.59	445577.18	4.65	438557.21	445402.53	4.65
pr1002_80_4	453609.46	587470.83	457842.39	467356.11	4.68	458557.63	464161.5	4.65	454449.43	463546.66	4.65
pr1002_80_5	599492.4	771733.95	601443.35	606392.15	4.58	601220.79	605159.05	4.60	601030.48	605036.78	4.60
pr1002_80_6	619003.36	805206.35	620905.87	626627.49	4.70	620390.26	624888.79	4.77	620980.59	624674.78	4.77
pr1002_80_7	508186.51	658640.85	511992.03	520148.95	4.71	512678.26	518684.16	4.61	511049.54	518270.58	4.61
pr1002_80_8	409733.88	518052.51	414454.61	431925.53	4.72	416386.88	426285.82	4.53	417195.97	426199.67	4.53
pr1002_80_9	557220.48	670387.49	559471.39	566974.28	4.64	560004.93	565634.88	4.73	560663.69	565474.87	4.73
brd14051_80_0	336983.07	403178.34	338615.5	340638.58	4.52	339017.38	341621.98	4.62	338752.62	341253.06	4.62
brd14051_80_1	277861.02	348787.38	278718.88	280954.58	4.57	278712.45	280898.21	4.57	279056.42	281444.44	4.57
brd14051_80_2	265370.92	321922.47	266964.57	271196.43	4.78	266965.71	271284.78	4.62	267098.27	271731.14	4.62
brd14051_80_3	189815.69	240361.74	190981.53	192988.97	4.55	190970.44	193195.6	4.53	190524.94	192859.4	4.53
brd14051_80_4	206068.45	275228.43	207846.42	211233.89	4.75	208142.71	211570.64	4.54	206986.40	211677.53	4.54
brd14051_80_5	303621.75	348578.27	304790.13	307187.47	4.66	304570.04	307066.74	4.78	304731.36	306944.31	4.78
brd14051_80_6	213405.23	266958.37	215496.73	219693.15	4.80	215905.29	220134.25	4.78	214846.74	220392.91	4.78
brd14051_80_7	263737.93	308039.16	265261	267579.79	4.52	265166.47	267383	4.67	265491.60	268425.6	4.67
brd14051_80_8	232967.83	298574.84	234215.29	236778.42	4.63	234262.37	238062.49	4.52	235200.12	238042.96	4.52
brd14051_80_9	317790.55	368183.51	318860.4	320911.18	4.53	319258.92	321512.59	4.57	319305.30	321576.77	4.57
fnl4461_80_0	153124.51	194685.8	154240.65	155854.44	4.79	154017.32	155260.49	4.61	154016.10	155538.61	4.61
fnl4461_80_1	174975.64	224516.74	175564.02	176753.56	4.50	175521.26	176567.39	4.75	175432.81	176488.95	4.75
fnl4461_80_2	162755.5	197782.39	163709.43	165437.58	4.73	163927.72	165265.48	4.51	163681.90	165198.58	4.51
fnl4461_80_3	160819.04	192927.87	161721.6	164211.03	4.75	162093.53	164341.05	4.52	161950.22	164289.55	4.52
fnl4461_80_4	151790.69	187440.09	152909.48	154583.49	4.76	152741.67	154474.03	4.55	152941.34	154525.71	4.55
fnl4461_80_5	131045.47	172293.83	131674.42	134594.78	4.52	132508.99	134525.24	4.69	132384.34	134838.76	4.69
fnl4461_80_6	125405.93	166418.99	126309.93	128634.42	4.62	126716.15	128784.35	4.72	126469.44	128585.69	4.72
fnl4461_80_7	125228.91	164627.71	127382.05	129202.79	4.58	126969.72	129172.82	4.69	126973.64	129372.03	4.69
fnl4461_80_8	185280.87	228208.31	185832.25	187148.81	4.74	185826.34	187068.01	4.64	185922.90	187144.12	4.64
fnl4461_80_9	130304.95	165022.1	131835.98	133896.72	4.63	131690.15	133899.8	4.66	131257.09	134025.52	4.66
d15112_80_0	551900.43	753989.49	556725.13	564166.47	4.78	552906.19	562737.67	4.59	552906.19	562737.67	4.59
d15112_80_1	815029.39	979921.76	816533.21	820893.85	4.55	817987.76	820839.25	4.72	817987.76	820839.25	4.72
d15112_80_2	828114.32	1080571.45	830372.82	834934.02	4.58	831176.05	836837.77	4.55	831176.05	836837.77	4.55
d15112_80_3	689450.94	964458.54	693037.65	702133.55	4.55	694369.37	708522.84	4.71	694369.37	708522.84	4.71
d15112_80_4	560385.47	737417.48	566738.01	578449.8	4.54	566738.01	578449.8	4.55	566738.01	578449.8	4.55
d15112_80_5	821959.4	1030515.79	825064.27	830641.04	4.76	825064.27	830641.04	4.61	825064.27	830641.04	4.61
d15112_80_6	715206.03	882086.8	719021.95	728868.7	4.68	719021.95	728868.7	4.68	719021.95	728868.7	4.68
d15112_80_7	958278.86	1155190.13	960032.55	964445.29	4.66	960032.55	964445.29	4.73	960032.55	964445.29	4.73
d15112_80_8	990277.77	1174384.17	990277.77	992123.77	4.55	990277.77	992123.77	4.52	990277.77	992123.77	4.52
d15112_80_9	672457.47	931587.71	672457.47	677541.47	4.75	672457.47	677541.47	4.78	672457.47	677541.47	4.78
nrw1379_80_0	64831.76	96656.39	65739.01	67598.57	4.68	65725.8	67571.78	4.73	65550.91	67927.83	4.73
nrw1379_80_1	64967.83	88394.72	66163.83	67947.85	4.61	65927.2	67899.4	4.65	65929.48	67713.73	4.65
nrw1379_80_2	73858.13	96499.97	74824.24	76023.55	4.65	74664.85	75881.09	4.63	74495.65	75981.26	4.63
nrw1379_80_3	100592.83	131733.34	101010.22	101650.71	4.62	100706.51	101643.99	4.63	100858.38	101810.63	4.63
nrw1379_80_4	98228.29	126451.61	98545.56	99177.28	4.52	98519.47	99278.86	4.59	98418.60	99239.92	4.59
nrw1379_80_5	75984.21	99492.47	76685.61	78143.45	4.57	76525.87	78206.56	4.65	76524.34	78122.3	4.65
nrw1379_80_6	79165.6	105024.23	79636.62	80402.19	4.54	79757.37	80363.95	4.65	79757.37	80363.95	4.65
nrw1379_80_7	73194.55	105328.52	74106.95	75589.58	4.55	74106.95	75589.58	4.75	74106.95	75589.58	4.75
nrw1379_80_8	83492.62	115793.31	83710.72	85128.66	4.57	83710.72	85128.66	4.74	83710.72	85128.66	4.74
nrw1379_80_9	67034.31	92380.75	67853.71	69638.94	4.62	67853.71	69638.94	4.69	67853.71	69638.94	4.69

Table 5. Results for ANMM-instances ($n = 80$, $k = 16$)

Instances	LB	Init.Sol	$MD = 2\times d_{max}$			$MD = 2.5\times d_{max}$			$MD = 3\times d_{max}$		
			Best.Sol	Aver.Sol	T	Best.Sol	Aver.Sol	T	Best.Sol	Aver.Sol	T
E-n30-k3	1643.3	2604.62	1670.41	1670.41	0.06	1670.41	1670.41	0.06	1670.41	1670.41	0.06
E-n30-k4	1643.3	2490.48	1645.87	1645.87	0.07	1645.87	1645.87	0.06	1645.87	1645.87	0.07
E-n33-k4	2819.43	3545.31	2819.43	2819.43	0.08	2819.43	2819.43	0.08	2819.43	2819.43	0.08
E-n51-k5	2209.64	3357.72	2292.60	2292.60	1.01	2292.60	2292.60	1.05	2292.60	2292.60	0.96
E-n76-k10	2310.09	3775.91	2417.55	2589.31	2.05	2417.55	2417.55	2.12	2417.55	2417.55	2.14
E-n76-k14	2005.4	3237.01	2049.33	2145.23	2.13	2030.43	2030.43	2.09	2030.43	2030.43	2.05
E-n76-k15	1962.47	3116.23	2031.95	2145.90	2.11	2031.95	2047.43	2.17	2031.95	2047.43	2.03

Table 6. Results for E-instances

Instances	LB	Init.Sol	$MD = 2\times d_{max}$			$MD = 2.5\times d_{max}$			$MD = 3\times d_{max}$		
			Best.Sol	Aver.Sol	T	Best.Sol	Aver.Sol	T	Best.Sol	Aver.Sol	T
P40k5	1537.79	2146.68	1587.52	1690.1559	0.08	1587.52	1568.86	0.08	1587.52	1568.86	0.08
P45k5	1912.31	2688.95	1968.57	2143.5088	0.09	1968.57	1961.12	0.10	1968.57	1961.12	0.09
P50k7	1547.89	2267.93	1580.65	1726.5957	0.97	1580.65	1575.59	0.97	1580.65	1575.59	1.01
P55k7	1766.56	2716.8	1840.22	2006.9836	1.07	1840.22	1838.15	0.99	1840.22	1838.15	1.07
P60k10	1676.35	2492.09	1723.04	1830.2216	1.18	1723.04	1707.72	1.16	1723.04	1707.72	1.20
P76k4	4686.92	6474.01	5059.4	5666.5279	2.18	5059.4	5023.78	2.16	5059.40	5023.78	2.16
P76k5	3820.02	5962.72	3820.02	3820.02	2.14	3820.02	3820.02	2.04	3820.02	3820.02	2.13
Pn70k10	2097.17	3414.5	2137.3	2332.3804	1.40	2137.3	2332.3804	1.40	2137.3	2332.3804	1.40

Table 7. Results for P-instances

Instances	Init.Sol	$MD = 2\times d_{max}$			$MD = 2.5\times d_{max}$			$MD = 3\times d_{max}$		
		Best.Sol	Aver.Sol	T	Best.Sol	Aver.Sol	T	Best.Sol	Aver.Sol	T
tai75a	6840.64	4840.69	5020.63	2.06	4803.02	5023.08	2.06	4803.02	5023.08	2.06
tai75b	5575.85	3782.95	3931.11	2.14	3793.62	3941.72	2.14	3793.62	3941.72	2.14
tai75c	7110.74	3838.18	4008.01	2.19	3848.39	4008.11	2.25	3848.39	4008.11	2.25
tai75d	6501.43	4762.54	5091.00	2.18	4762.54	5091.00	2.22	4762.54	5091.00	2.22
tai100a	11398.60	7798.82	8167.83	6.35	7772.71	8042.25	6.37	7733.07	8150.46	6.37
tai100b	9775.59	7002.04	7510.64	6.42	6968.49	7452.22	6.34	6859.95	7398.34	6.34
tai100c	7895.75	4773.88	4945.76	6.35	4773.88	4945.76	6.35	4773.88	4945.76	6.35
tai100d	9051.64	5411.22	5695.49	6.38	5384.50	5627.70	6.36	5357.70	5646.42	6.36
tai150a	16188.04	14048.22	14595.99	67.08	14052.37	14594.23	67.77	14075.20	14559.93	67.77
tai150b	14018.07	11225.23	11802.84	67.15	11225.23	11802.84	68.54	11303.73	11804.18	68.54
tai150c	13293.35	9756.99	10177.14	66.69	9763.81	10151.19	68.12	9789.65	10210.75	68.12
tai150d	13001.42	9843.82	10201.89	67.46	9843.82	10201.89	69.01	9846.68	10199.94	69.01

Table 8. Results for Tai-instances

GRASP + VND. On the other hand, the results from the different values of MD in Table 9 indicate that the distance constraint also affects the quality of solutions.

In the CMT and Kelly instances with the maximum total distance traveled in a route (as in Tables 11 and 12), as expected, our proposed algorithm shows a significant improvement comparing to the initial solutions, with average $Improv$ of 24.01 % to 26.31 %.

Table 12 shows the evolution of the average deviation to the initial solutions during the iterations in some instances. The deviations are 26.07 %, 28.05 %, 28.34 %, 28.61 %, 28.86 %, and 28.86 % for the first local optimum, obtained by one, ten, twenty, thirty, fifty and one-hundred calls VNS, respectively. A major part of the descent obtained is about 0.87 % by fifty to three-hundred calls VNS. We can observe that additional iterations give a minor improvement with the big running time. Hence, the first way to reduce the long running time is to use no more than fifty calls to VNS, and the improvement of our algorithm is about 28.61 %. A much faster option is to run the initial construction phase, then improve it by using a single call to VNS. As as a result, we can obtain an average deviation of 26.07 % and average

Instances	$MD = 2 \times d_{max}$			$MD = 2.5 \times d_{max}$			$MD = 3 \times d_{max}$		
	Gap_2	*Impro*	T	Gap_2	*Impro*	T	Gap_2	*Impro*	T
pr1002_60_x	0.53	21.30	1.12	0.48	21.35	1.45	0.34	21.45	1.45
brd14051_60_x	0.66	25.63	1.13	0.46	25.77	1.45	0.53	25.73	1.45
fnl4461_60_x	0.48	20.26	1.12	0.39	20.33	1.45	0.29	20.41	1.45
d15112_60_x	0.56	23.94	1.11	0.39	24.07	1.46	0.31	24.12	1.46
nrw1379_60_x	0.75	25.65	1.12	0.70	25.69	1.45	0.63	25.74	1.45
pr1002_70_x	0.92	24.55	1.48	0.88	24.58	1.97	0.82	24.62	1.97
brd14051_70_x	0.53	21.62	1.49	0.50	21.65	1.84	0.50	21.66	1.84
fnl4461_70_x	0.66	23.53	1.46	0.62	23.56	1.94	0.59	23.58	1.94
d15112_70_x	0.75	23.70	1.51	0.73	23.73	1.85	0.69	23.70	1.85
nrw1379_70_x	0.72	24.17	1.44	0.64	24.23	1.95	0.61	24.25	1.95
pr1002_80_x	0.73	22.23	3.58	0.68	22.26	4.66	0.65	22.28	4.66
brd14051_80_x	0.62	17.85	3.56	0.57	17.89	4.62	0.56	17.89	4.62
fnl4461_80_x	0.79	20.29	3.59	0.73	20.34	4.63	0.72	20.35	4.63
d15112_80_x	0.43	21.59	3.57	0.41	21.61	4.65	0.39	21.62	4.65
nrw1379_80_x	0.95	25.60	3.53	0.85	25.67	4.67	0.81	25.70	4.67
E-instances	5.29	32.69	1.39	5.29	32.69	1.42	5.29	32.69	1.37
P-instances	3.14	30.18	1.48	3.14	30.18	1.45	3.14	30.18	1.48
Tai-instances	-	29.77	32.79	-	32.42	33.20	-	29.83	33.31
Aver	1.09	24.14	3.69	1.03	24.33	4.23	0.99	24.21	4.23

Table 9. Average results for dataset type 2

Instances	$MD = 2 \times d_{max}$			$MD = 2.5 \times d_{max}$			$MD = 3 \times d_{max}$		
	Gap_1	Gap_2	T	Gap_1	Gap_2	T	Gap_1	Gap_2	T
pr1002_60_x	0.53	21.30	1.12	0.48	21.35	1.45	0.34	21.45	1.45
brd14051_60_x	0.66	20.43	1.13	0.46	24.86	1.45	0.53	24.80	1.45
fnl4461_60_x	0.48	21.78	1.12	0.48	21.78	1.45	0.48	21.78	1.45
d15112_60_x	0.56	21.18	1.11	0.56	21.18	1.46	0.56	21.18	1.46
nrw1379_60_x	0.63	20.98	1.12	0.63	20.98	1.45	0.63	20.98	1.45
pr1002_70_x	0.88	24.58	1.14	0.88	24.58	1.52	0.88	24.58	1.52
brd14051_70_x	0.50	21.65	1.15	0.50	21.65	1.41	0.50	21.66	1.41
fnl4461_70_x	0.62	23.56	1.12	0.62	23.56	1.49	0.62	23.56	1.49
d15112_70_x	0.69	23.70	1.16	0.69	23.70	1.13	0.69	23.70	1.42
nrw1379_70_x	0.64	24.23	1.11	0.64	24.23	1.11	0.61	24.25	1.50
pr1002_80_x	0.68	22.26	3.58	0.73	22.23	3.58	0.65	22.28	3.58
brd14051_80_x	0.57	17.89	3.56	0.62	17.85	3.55	0.56	17.89	3.55
fnl4461_80_x	0.73	20.34	3.59	0.79	20.29	3.56	0.72	20.35	3.56
d15112_80_x	0.41	21.61	3.57	0.39	21.62	3.57	0.39	21.62	3.57
nrw1379_80_x	0.95	25.60	3.53	0.85	25.67	3.59	0.81	25.70	3.59
E-instances	2.28	32.78	1.07	2.14	32.68	1.09	2.14	32.68	1.06
P-instances	3.14	30.18	1.14	3.14	30.18	1.11	3.14	30.18	1.14
Tai-instances	-	29.83	25.20	-	29.91	25.63	-	29.98	25.63
Aver	0.88	23.55	3.14	0.86	23.79	3.31	0.84	23.81	3.35

Table 10. Average results for dataset type 2

Instances	*n*	*k*	*MD*	*Init.Sol*	*Best.Sol*	*Aver.Sol*	*Impro*	*T*
kelly01	240	9	650	72143.84	56121.25	57116.60	22.21	177.04
kelly02	320	10	700	152816.69	104163.33	106346.89	31.84	349.84
kelly04	480	10	1600	353374.40	271826.46	278169.07	23.08	566.42
kelly05	200	5	1800	153187.07	116050.81	118758.33	24.24	179.39
kelly06	280	7	1500	157506.03	127733.90	129785.16	18.90	404.47
kelly07	360	8	1300	218214.03	166321.67	170351.03	23.78	615.73
Aver							24.01	382.15

Table 11. Results for Kelly-instances

Dataset	1 iteration		20 iterations		30 iterations		50 iterations		100 iterations		200 iterations	
	$\overline{impro}$	$\bar{T}$	$\overline{impro}$	$\bar{T}$	$\overline{impro}$	$\bar{T}$	$\overline{impro}$	$\bar{T}$	$\overline{impro}$	$\bar{T}$	$\overline{impro}$	$\bar{T}$
E-instances	27.23	0.16	31.62	0.45	32.24	0.64	32.71	1.07	32.71	2.24	32.71	4.15
P-instances	28.01	0.17	29.80	0.48	30.04	0.68	30.18	1.13	30.18	2.26	30.18	4.72
Tai-instances	28.29	1.83	29.56	12.74	29.84	16.38	29.91	25.49	29.91	52.80	29.91	105.59
CMT	24.98	17.45	25.87	23.92	26.31	27.10	26.31	35.06	26.31	61.03	26.31	141.56
Kelly	22.01	188.22	23.55	259.52	23.88	293.74	24.01	382.15	24.01	667.34	24.01	1544.29
Aver	26.10	41.56	28.08	59.42	28.46	67.71	28.62	88.98	28.62	157.13	28.62	360.06

Table 12. Evolution of average *Impro* deviation to *Init.Sol*

time of 41.46 seconds, even for the instances which are up to 560 customers.

Most previous algorithms are proposed for specific variants; hence, they do not apply for the other variants. However, our proposed algorithm is applicable to multiple variants of MTRPD, although it was not designed for solving them. In comparison with the state-of-the-art algorithms in [15, 23, 28, 29, 30, 31], our TS + VNS algorithm's solutions are at least as good as already the existing CCVRP algorithm in [23, 28, 30], MTRP algorithm in [15, 29], TRP algorithm in [31]. Specifically, for CCVRP problem, our algorithm obtains the better solutions for CMT1, CMT2, CMT3, CMT4 or at least similar solutions for the others in Table 14. For the MTRP problem, as shown in Table 13, the quality of our solutions is much better than the algorithm of Ezzine et al. in [15] and comparable with the algorithm of Nucamendi-Guillén et al. Tables 15 to 18 show the experimental results for the TRP problem. Our algorithm outperforms that of Silva et al. [31] for four instances, and has similar performance for the most of instances in TRP-100-Rx. In TRP-200-Rx, although the VNS + TS metaheuristic cannot find any new best solution, our average solution quality is slightly improved. In addition, our algorithm can find the optimal solutions for the problems with 50 to 100 vertices in several seconds, as shown in Tables 15 and 18 (note that the optimal solutions for the instances are extracted from Abeledo et al. [2]).

Our metaheuristic performs well because of two reasons:

1. The algorithm uses more neighborhoods than the others. Therefore, the explored part of the solution space is more substantial. Hence, the chances of finding even better solutions are higher. The extension of explored part is not time-

Instances	IOE		SNG		Our Algorithm		
	Best.Sol	*T*	*Best.Sol*	*T*	*Best.Sol*	*T*	*cTime*
E-n51-k5	3320	2.25	2209.64	0.70	2209.64	1.31	3.25
E-n76-k10	4094	1.48	2310.09	4.20	2310.09	2.67	6.62
E-n76-k14	3762	0.50	2005.40	3.40	2005.40	2.77	6.87
E-n76-k15	3822	0.09	1962.47	2.81	1962.47	2.74	6.80
E-n101-k8	6383	89.4	-	-	4051.47	6.40	15.87
E-n101-k14	5048	5.43	-	-	3288.53	6.74	16.72
P-n50-k7	-	-	1547.89	0.70	1547.89	1.26	3.12
P-n55-k7	-	-	1766.56	1.01	1766.56	1.39	3.45
P-n60-k10	-	-	1676.35	1.42	1676.35	1.54	3.82
P-n76-k4	-	-	4686.92	3.40	4686.92	2.83	7.02
P-n76-k5	-	-	3820.02	3.63	3820.02	2.78	6.89
CMT1	-	-	2209.64	0.70	2209.64	1.40	3.47
CMT2	-	-	2310.09	4.19	2310.09	2.78	6.89
CMT3	-	-	4002.90	14.94	4002.90	6.40	15.87
tai100a	-	-	7809.43	13.63	7733.07	6.37	15.54
tai100b	-	-	7038.60	12.83	6859.95	6.34	15.47
tai100a	-	-	4868.61	14.12	4786.94	6.35	15.49
tai100b	-	-	5422.63	14.23	5357.70	6.36	15.52

Table 13. Comparision with state-of-the-art metaheuristic algorithm for MTRP ($MD = 0$)

Instances	**MA1**		**ALNS**		**L. Ke's algorithm**		**Our Algorithm**		
	Best.Sol	*T*	*Best.Sol*	*T*	*Best.Sol*	*T*	*Best.Sol*	*T*	*cTime*
CMT1	2230.35	10.63	2230.35	30.29	2230.35	17.64	2209.64	1.40	3.47
CMT2	2421.90	27.78	2391.63	60.77	2391.63	22.48	2310.09	2.78	6.89
CMT3	4073.12	97.91	4045.42	172.4	4045.42	60.96	4002.90	6.40	15.8
CMT4	4987.52	449.4	4987.52	235.1	4987.52	92.68	4953.94	63.0	156.
CMT5	5810.12	1035.	5838.32	277.3	5809.59	135.36	5809.59	11.2	92.7
CMT12	3558.92	53.72	3558.92	38.20	3558.92	152.74	3564.24	9.42	23.3

Table 14. Comparision with state-of-the-art metaheuristic algorithm for CCVRP ($MD = 0$)

consuming because of a constant time operation for calculating the latency cost of each neighboring solution.

2. In some cases, while their algorithms get trapped in cycles, our algorithm overcomes the issue and obtains the better solutions.

In Tables 13–14, the average scaled running time of the VNS + TS algorithm is better than those of Ban et al., Ngueveu et al., and Ribeiro et al., and as well as the algorithm of Ke et al. Besides, it grows quite moderately with the algorithm of Nucamendi-Guillén et al.

Instances $k = 1$ *MD*=0	*OPT*	**Our Algorithm**			
		Best.Sol	*Aver.Sol*	*T*	*cTime*
TRP-50-R1	12198	**12198**	**12198**	0.49	1.20
TRP-50-R2	11621	**11621**	**11674**	0.57	1.41
TRP-50-R3	12139	**12139**	**12139**	0.61	1.50
TRP-50-R4	13071	**13071**	**13071**	0.61	1.49
TRP-50-R5	12126	**12126**	**12284**	0.58	1.44
TRP-50-R6	12684	**12684**	**12684**	0.55	1.36
TRP-50-R7	11176	**11176**	**11176**	0.59	1.45
TRP-50-R8	12910	**12910**	**12945**	0.61	1.50
TRP-50-R9	13149	**13149**	**13149**	0.62	1.53
TRP-50-R10	12892	**12892**	**12892**	0.62	1.53
TRP-50-R11	12103	**12103**	**12181**	0.61	1.44
TRP-50-R12	10633	**10633**	**10633**	0.61	1.47
TRP-50-R13	12115	**12115**	**12115**	0.56	1.47
TRP-50-R14	13117	**13117**	**13117**	0.58	1.47
TRP-50-R15	11986	**11986**	**11986**	0.61	1.47
TRP-50-R16	12138	**12138**	**12138**	0.58	1.47
TRP-50-R17	12176	**12176**	**12176**	0.49	1.48
TRP-50-R18	13357	**13357**	**13357**	0.57	1.48
TRP-50-R19	11430	**11430**	**11430**	0.61	1.48
TRP-50-R20	11935	**11935**	**11935**	0.58	1.48
Aver				0.58	1.47

Table 15. Comparision with state-of-the-art metaheuristic algorithm for TRP (TPR-50-Rx)

Instances $k = 1$ *MD*=0	**MS**	Our Algorithm			
	Best.Sol	*Best.Sol*	*Aver.Sol*	*T*	*cTime*
TRP-100-R1	32779	32680	32680	5.67	14.00
TRP-100-R2	33435	31598	31598	5.77	14.25
TRP-100-R3	32390	32390	32390	5.67	14.00
TRP-100-R4	34733	35208	35208	5.98	14.76
TRP-100-R5	32598	32598	32598	5.87	14.50
TRP-100-R6	34159	34159	34159	5.46	13.49
TRP-100-R7	33375	33375	33375	5.05	12.47
TRP-100-R8	31780	32479	32479	5.36	13.23
TRP-100-R9	34167	34167	34167	5.98	14.76
TRP-100-R10	31605	31605	31289	5.26	12.98
TRP-100-R11	34188	34188	34188	5.26	13.84
TRP-100-R12	32146	32146	30487	5.46	13.83
TRP-100-R13	32604	31930	31930	5.05	13.78
TRP-100-R14	32433	32433	32433	5.67	13.76
TRP-100-R15	32574	32574	32574	5.87	13.67
TRP-100-R16	33566	33566	33275	5.26	13.58
TRP-100-R17	34198	34198	34198	5.87	13.59
TRP-100-R18	31929	31929	31929	5.46	13.70
TRP-100-R19	33463	33463	33463	6.08	13.75
TRP-100-R20	33632	33363	33363	5.36	13.65
Aver				5.57	13.72

Table 16. Comparision with state-of-the-art metaheuristic algorithm for TRP (TRP-100-Rx)

Instances $k=1$ MD=0	**MS**		Our Algorithm			
	Best.Sol	*Aver.Sol*	*Best.Sol*	*Aver.Sol*	*T*	*cTime*
TRP-200-R1	88787	88794.60	88789	88794.25	89.51	218.40
TRP-200-R2	91977	92013.10	91977	91989.30	91.77	223.93
TRP-200-R3	92568	92631.20	92570	92570.90	96.31	234.98
TRP-200-R4	93174	93192.30	93174	93178.60	101.97	248.81
TRP-200-R5	88737	88841.20	88737	88740.65	94.04	229.46
TRP-200-R6	91589	91601.90	91591	91590.50	99.70	243.28
TRP-200-R7	92754	92763.20	92754	92759.85	92.91	226.69
TRP-200-R8	89048	89049.00	89048	89051.25	94.04	229.46
TRP-200-R9	86326	86326.00	86326	86327.70	90.64	221.16
TRP-200-R10	91554	91596.50	91554	91555.51	91.77	223.93
TRP-200-R11	92655	92700.60	92658	92658.22	94.04	229.46
TRP-200-R12	91457	91504.10	91457	91458.43	90.64	221.16
TRP-200-R13	86155	86181.40	86159	86178.31	97.44	237.75
TRP-200-R14	91882	91929.10	91882	91890.95	95.17	232.22
TRP-200-R15	88914	88912.40	88914	88928.95	90.64	221.16
TRP-200-R16	89313	89364.70	89313	89316.21	96.31	234.98
TRP-200-R17	89089	89118.30	89089	89092.93	96.31	234.98
TRP-200-R18	93619	93676.60	93619	93632.77	99.70	243.28
TRP-200-R19	93369	93401.60	93369	93371.85	94.04	229.46
TRP-200-R20	86294	86292.00	86294	86294.35	95.17	232.22
Aver					94.60	230.82

Table 17. Comparision with state-of-the-art metaheuristic algorithm for TRP (TRP-200-Rx)

Instances $k=1$ MD=0	n	*OPT*	*UB*	Our algorithm			
				Best.Sol	*Aver.Sol*	*T*	*cTime*
dantzig42	42	12528	12650	**12528**	12528	0.56	1.50
att48	48	209320	25315	**209320**	209320	1.45	3.87
eil51	51	10178	10593	**10178**	10178	1.56	4.17
berlin52	52	143721	15209	**143721**	143721	1.51	4.03
st70	70	20557	25809	**20557**	20557	2.43	6.49
KroA100	100	983128	10912	**983128**	983128	8.25	22.03
KroB100	100	986008	10212	**986008**	986008	8.12	21.68
KroC100	100	961324	11013	**961324**	961324	8.28	22.11
KroD100	100	976965	10253	**976965**	976965	8.19	21.87
Aver						4.48	11.97

Table 18. Comparision with state-of-the-art metaheuristic algorithm for TRP (TSPLIB)

5 CONCLUSIONS

In this paper, we study the global structure of the MTRPD solution space. We have proposed a new effective meta-heuristic algorithm for MTRPD, which combines Insertion Heuristic (IH), Tabu Search (TS), and Variable Neighborhood Search (VNS). Our algorithm has been suitable for the global structure of the solution space. Moreover, we introduce the novel neighborhoods' structure as well as the constant time operation for efficient calculation of the latency cost for each neighboring solution.

The extensive computational experiments on benchmark instances show that the proposed algorithm is able to find the optimal solutions for all instances with up to 50 vertices in a fraction of seconds. Moreover, almost all the found solutions for instances from 60 to 80 vertices fall into the range of 0.9 %–1.1 % of the lower bounds of the optimal solutions at a reasonable amount of time. For the larger number of vertices, our algorithm obtains good-quality solutions fast. Additionally, our algorithm can find better solutions than the state-of-the-art ones.

REFERENCES

[1] Archer, A.—Levin, A.—Williamson, D. P.: A Faster, Better Approximation Algorithm for the Minimum Latency Problem. SIAM Journal on Computing, Vol. 37, 2008, No. 5, pp. 1472–1498, doi: 10.1137/07068151X.

[2] Abeledo, H.—Fukasawa, R.—Pessoa, A.—Uchoa, E.: The Time Dependent Traveling Salesman Problem: Polyhedra and Algorithm. Mathematical Programming Computation, Vol. 5, 2013, No. 1, pp. 27–55, doi: 10.1007/s12532-012-0047-y.

[3] Afrati, F.—Cosmadakis, S.—Papadimitriou, C. H.—Papageorgiou, G.—Papakostantinou, N.: The Complexity of the Travelling Repairman Problem. Informatique Théorique et Applications, Vol. 20, 1986, No. 1, pp. 79–87.

[4] Arora, S.—Karakostas, G.: Approximation Schemes for Minimum Latency Problems. Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing (STOC '99), 1999, pp. 688–693, doi: 10.1145/301250.301432.

[5] Ausiello, G.—Leonardi, S.—Marchetti-Spaccamela, A.: On Salesmen, Repairmen, Spiders and Other Traveling Agents. In: Bongiovanni, G., Petreschi, R., Gambosi, G. (Eds.): Algorithms and Complexity (CIAC 2000). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 1767, 2000, pp. 1–16, doi: 10.1007/3-540-46521-9_1

[6] Ban, H. B.—Nguyen, D. N.: Improved Genetic Algorithm for Minimum Latency Problem. Proceedings of the 2010 Symposium on Information and Communication Technology (SoICT '10), 2010, pp. 9–15, doi: 10.1145/1852611.1852614.

[7] Ban, H. B.—Nguyen, K.—Ngo, M. C.—Nguyen, D. N.: An Efficient Exact Algorithm for Minimum Latency Problem. Progress in Informatics, No. 10, 2013, pp. 167–174, doi: 10.2201/NiiPi.2013.10.10.

[8] Ban, H. B.—Nguyen, D. N.: A Meta-Heuristic Algorithm Combining Between Tabu and Variable Neighborhood Search for the Minimum Latency Problem. The 2013 RIVF International Conference on Computing and Communication Technologies – Research, Innovation, and Vision for Future (RIVF), 2013, pp. 192–197, doi: 10.1109/RIVF.2013.6719892.

[9] Ban, H. B.: A GRASP + VND Algorithm for the Multiple Traveling Repairman Problem with Distance Constraints. Journal of Computer Science and Cybernetics, Vol. 33, 2017, No. 3, pp. 272–288, doi: 10.15625/1813-9663/33/3/10511.

[10] Blum, A.—Chalasani, P.—Coppersmith, D.—Pulleyblank, B.—Raghavan, P.—Sudan, M.: The Minimum Latency Problem. Proceedings

of the Twenty-Sixth Annual ACM Symposium on Theory of Computing (STOC '94), 1994, pp. 163–171, doi: 10.1145/195058.195125.

[11] Chaudhuri, K.—Godfrey, B.—Rao, S.—Talwar, K.: Paths, Trees and Minimum Latency Tours. Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS '03), 2003, pp. 36–45.

[12] Li, C.-L.—Simchi-Levi, D.—Desrochers, M.: On the Distance Constrained Vehicle Routing Problem. Operations Research, Vol. 40, 1992, No. 4, pp. 790–799, doi: 10.1287/opre.40.4.790.

[13] Erera, A. L.—Morales, J. C.—Savelsbergh, M.: The Vehicle Routing Problem with Stochastic Demand and Duration Constraints. Transportation Science, Vol. 44, 2010, No. 4, pp. 474–492, doi: 10.1287/trsc.1100.0324.

[14] Dongarra, J. J.: Performance of Various Computers Using Standard Linear Equations Software. Linpack Benchmark Report, University of Tennessee, Computer Science Technical Report, CS-89-85, 2013.

[15] Ezzine, I. O.—Elloumi, S.: Polynomial Formulation and Heuristic Based Approach for the k-Travelling Repairman Problem. International Journal of Mathematics in Operational Research, Vol. 4, 2012, No. 5, pp. 503–514, doi: 10.1504/IJMOR.2012.048928.

[16] Feo, T. A.—Resende, M. G. C.: Greedy Randomized Adaptive Search Procedures. Journal of Global Optimization, Vol. 6, 1995, No. 2, pp. 109–133, doi: 10.1007/BF01096763.

[17] Fischetti, M.—Laporte, G.—Martello, S.: The Delivery Man Problem and Cumulative Matroids. Operations Research, Vol. 41, 1993, No. 6, pp. 1055–1064, doi: 10.1287/opre.41.6.1055.

[18] Goemans, M.—Kleinberg, J.: An Improved Approximation Ratio for the Minimum Latency Problem. Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '96), 1996, pp. 152–158.

[19] Glover, F.: Tabu Search – Part II. INFORMS Journal on Computing, Vol. 2, 1990, No. 1, pp. 4–32, doi: 10.1287/ijoc.2.1.4.

[20] Fakcharoenphol, J.—Harrelson, C.—Rao, S.: The k-Traveling Repairman Problem. Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '03), 2003, pp. 655–664.

[21] Johnson, D. S.—McGeoch, L. A.: The Traveling Salesman Problem: A Case Study in Local Optimization in Local Search. In: Aarts, E. H. L., Lenstra, J. K. (Eds.): Combinatorial Optimization. John Wiley and Sons, New York, 1997, pp. 215–310.

[22] Jothi, R.—Raghavachari, B.: Minimum Latency Tours and the k-Traveling Repairmen Problem. In: Farach-Colton, M. (Ed.): LATIN 2004: Theoretical Informatics. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 2976, 2004, pp. 423–433, doi: 10.1007/978-3-540-24698-5_46.

[23] Ke, L.—Feng, Z.: A Two-Phase Metaheuristic for the Cumulative Capacitated Vehicle Routing Problem. Computers and Operations Research, Vol. 40, 2013, No. 2, pp. 633–638, doi: 10.1016/j.cor.2012.08.020.

[24] Laporte, G.—Nobert, Y.—Desrochers, M.: Optimal Routing under Capacity and Distance Restrictions. Operations Research, Vol. 33, 1985, No. 5, pp. 1050–1073,

doi: 10.1287/opre.33.5.1050.

[25] Luo, Z.—Qin, H.—Lim, A.: Branch-and-Price-and-Cut for the Multiple Traveling Repairman Problem with Distance Constraints. European Journal of Operational Research, Vol. 234, 2014, No. 1, pp. 49–60, doi: 10.1016/j.ejor.2013.09.014.

[26] Martin, O.—Otto, S. W.—Felten, E. W.: Large-Step Markov Chains for the Traveling Salesman Problem. Complex Systems, Vol. 5, 1991, No. 3, pp. 299–326.

[27] Mladenović, N.—Hansen, P.: Variable Neighborhood Search. Computers and Operations Research, Vol. 24, 1997, No. 11, pp. 1097–1100, doi: 10.1016/S0305-0548(97)00031-2.

[28] Ngueveu, S. U.—Prins, C.—Wolfler Calvo, R.: An Effective Memetic Algorithm for the Cumulative Capacitated Vehicle Routing Problem. Computers and Operations Research, Vol. 37, 2010, No. 11, pp. 1877–1885, doi: 10.1016/j.cor.2009.06.014.

[29] Nucamendi-Guillén, S.—Martínez-Salazar, I.—Angel-Bello, F.—Moreno-Vega, J. M.: A Mixed Integer Formulation and an Efficient Metaheuristic Procedure for the k-Travelling Repairmen Problem. Journal of the Operational Research Society, Vol. 67, 2016, No. 8, pp. 1121–1134, doi: 10.1057/jors.2015.113.

[30] Ribeiro, G.—Laporte, G.: An Adaptive Large Variable Neighborhood Search Heuristic for the Cumulative Capacitated Vehicle Routing Problem. Computers and Operations Research, Vol. 39, 2012, No. 3, pp. 728–735, doi: 10.1016/j.cor.2011.05.005.

[31] Silva, M. M.—Subramanian, A.—Vidal, T.—Ochi, L. S.: A Simple and Effective Metaheuristic for the Minimum Latency Problem. European Journal of Operational Research, Vol. 221, 2012, No. 3, pp. 513–520, doi: 10.1016/j.ejor.2012.03.044.

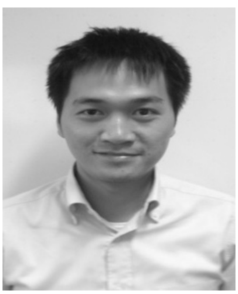
[32] Simchi-Levi, D.—Berman, O.: Minimizing the Total Flow Time of N Jobs on a Network. IIE Transactions, Vol. 23, 1991, No. 3, pp. 236–244, doi: 10.1080/07408179108963858.

[33] Wu, B. Y.—Huang, Z.-N.—Zhan, F.-J.: Exact Algorithms for the Minimum Latency Problem. Information Processing Letters, Vol. 92, 2004, No. 6, pp. 303–309, doi: 10.1016/j.ipl.2004.09.009.

[34] NEO: Capacitated VRP Instances. `http://neo.lcc.uma.es/vrp/vrp-instances/capacitated-vrp-instances/`, 2013.

Ha-Bang Ban received his B.E. in information technology in 2006 and the Ph.D. in computer science in 2015, both from the Hanoi University of Science and Technology (HUST), Vietnam. He is currently Lecturer at the School of Information and Communication Technology (SOICT), HUST, Vietnam. His research interests include algorithms, graphs, optimization, logistics, etc. He has published many publications in international peer-reviewed journals and conferences.

Duc-Nghia Nguyen is Associate Professor of computer science at the School of Information and Communication Technology, Hanoi University of Science and Technology (HUST), Vietnam. He received his Ph.D. in computer science in 1988 from Belarusian State University. His current research interests include algorithms and optimization, high performance computing, data science.



Kien Nguyen received his B.E. in electronics and telecommunications from Hanoi University of Science and Technology (HUST), Vietnam, in 2004, and the Ph.D. in informatics from the Graduate University for Advanced Studies, Japan, in 2012. He is currently Assistant Professor at the Graduate School of Engineering, Chiba University, Japan. Before joining Chiba University, he was a researcher at the National Institute of Information and Communications Technology (NICT), Japan, during 2014–2018. His research interests include communication networks, the Internet, and the Internet of Things (IoT). He has published 70+ publications in international peer-reviewed journals and conferences. Besides, he has co-authored submitted patents and Internet Engineering Task Force (IETF) Internet drafts. He is a member of IEICE and a senior member of IEEE.