Ban and Nguyen *Comput Soc Netw* (2021) 8:3
https://doi.org/10.1186/s40649-020-00084-7

Computational Social Networks

RESEARCH Open Access

A hybrid metaheuristic for solving asymmetric distance-constrained vehicle routing problem

Ha-Bang Ban* and Phuong Khanh Nguyen

*Correspondence:
BangBH@soict.hust.edu.vn
School of Information and Communication Technology, Hanoi University of Science and Technology, Hanoi, Vietnam

Abstract

The Asymmetric Distance-Constrained Vehicle Routing Problem (ADVRP) is NP-hard as it is a natural extension of the NP-hard Vehicle Routing Problem. In ADVRP problem, each customer is visited exactly once by a vehicle; every tour starts and ends at a depot; and the traveled distance by each vehicle is not allowed to exceed a predetermined limit. We propose a hybrid metaheuristic algorithm combining the Randomized Variable Neighborhood Search (RVNS) and the Tabu Search (TS) to solve the problem. The combination of multiple neighborhoods and tabu mechanism is used for their capacity to escape local optima while exploring the solution space. Furthermore, the intensification and diversification phases are also included to deliver optimized and diversified solutions. Extensive numerical experiments and comparisons with all the state-of-the-art algorithms show that the proposed method is highly competitive in terms of solution quality and computation time, providing new best solutions for a number of instances.

Keywords: DMPTW, ILS, TS, RVND

Introduction and literature review

The basic Vehicle Routing Problem (VRP) aims to design the least cost routes from a single depot to a set of geographically distributed customers such that each customer is visited exactly once by one vehicle. The VRP was first introduced by Dantzig and Ramser in 1959 [1], and was proved NP-hard by Lenstra and Kan in 1981 [2]. For 60 years, the VRP has been one of the most extensively studied problems in operations research with various variants deriving from real-world applications [3–6]. We study in this paper a variant of VRP named Distance-Constrained VRP (DVRP). An upper bound $D_{\max}$ is, thus, imposed on the length of any route. This restriction is relevant in many practical situations: school bus must go to pick up their students and come back to the school before its starting time, postal vans used to empty post boxes must return to the post office within a prescribed time [7], etc. When the distance from customer i to customer j differs from that of customer i to customer j, we call the problem the Asymmetric Distance Constrained VRP (ADVRP). Otherwise, it is defined as the Symmetric DVRP. Surprisingly, the literature is rich for Symmetric VRPs and poor for Asymmetric VRPs, although the Symmetric VRPs are considered as a

special case of Asymmetric VRP. In Asymmetric VRPs, the real distance will depend upon the specific location of the nodes in the territory and also on the structure of the road network that communicates them. Moreover, when considering oriented networks, real distances might not have to be symmetric.

To the best of our knowledge, the ADVRP has not been addressed well in the literature before. There are a few papers which proposed exact algorithms [8, 7] that could solve large instances. However, it could not always find a feasible solution due to the lack of memory. Moreover, when distance constraint is tight, solving the problem becomes harder, and these exact methods were stopped before it could find any feasible solution. For these reasons, developing metaheuristics to find a good feasible solution in a short computation time is a suitable approach. However, there exists only a metaheuristic [8] reported in the literature for ADVRP. The algorithm is based on the principles of Randomized Variable Neighborhood Search (RVNS). Unfortunately, it can be trapped into cycles as it returns to the points previously explored in the solution space. In this case, the algorithm gets stuck into local optima. Therefore, in this paper, we propose a hybrid metaheuristic combining RVNS and tabu search to overcome all these issues.

Besides ADVRP, we use algorithm to address also the Asymmetric Capacitated VRP (ACVRP) and Multiple Traveling Repairman Problem With Distance Constraints (MTRPD) to demonstrate its performance. Extensive numerical experiments on benchmark instances of ADVRP show that our algorithm could be comparable with the previous state-of-the-art algorithms in terms of solution quality and running time. Interestingly, in many cases, our proposed method is able to improve the best-known solutions available from the literature.

The remainder of the paper is structured as follows. We define the ADVRP in next section. Our proposed metaheuristic and its components are presented next, followed by the section dedicated to the experimental results, and the conclusion.

Problem description

The ADVRP is a generalization of the VRP, it is, thus, also a NP-hard problem. The ADVRP is defined on a complete graph K_n with the vertex set $V = \{v_1, v_2, ..., v_n\}$, an asymmetric distance matrix $C = \{c(v_i, v_j) \mid i, j = 1, 2, ..., n\}$, where $c(v_i, v_j)$ is the distance between two vertices v_i and v_j, $(c(v_i, v_j) \neq c(v_j, v_i))$. A fleet of k identical vehicles is based at the depot v_1. Suppose that the tour $T = \{R_1, ..., R_l, ..., R_k\}$ is a set of k routes obtained from these k vehicles, respectively. The route performed by the vehicle lth $(1 \leq l \leq k)$ is defined as $R_l = \{v_1, ..., v_h, ..., v_m, v_1\}, (1 < m \leq n)$, made up of a sequence of vertices, starting and ending at the depot v_1. The length of the route R_l is defined as the total traveled distances of the vehicle lth and its length cannot exceed the predetermined limit $D_{\max}$:

$$L(R_l) = \sum_{i=1}^{m-1} c(v_i, v_{i+1}) + c(v_m, v_1), \tag{1}$$

$$L(R_l) \leq D_{\max}. \tag{2}$$

The length of the tour T is the sum of its all k vehicles' total traveled distances:

$$L(T) = \sum_{l=1}^{k} L(R_l). \tag{3}$$

The ADVRP aims to determine the tour T with minimal length.

The proposed hybrid metaheuristic

The efficient hybrid metaheuristic we proposed brings together the components of Greedy Randomized Adaptive Search Procedures (GRASP) [9], Tabu search (TS) [10] and Randomized VNS [11]. It, thus, consists of two phases performed iteratively: (1) the first phase uses the GRASP combining with k-means [12] to generate initial solutions; (2) the second phase then enhances solutions consecutively by means of an RVNS with multiple neighborhoods combined with a short-term memory mechanism in the spirit of TS.

An outline of our proposed algorithm (HVT algorithm) is shown in Algorithm 1. In Step 1, the initial solution is generated using the cluster first–route second scheme. This initial solution is then enhanced in Step 2. We use the idea of RVNS based on the principle of systematically exploring several different neighborhoods. Moreover, to avoid cycling, we use tabu lists of the recent types of moves in the solution space to prohibit reversing these moves. Thus, at each iteration of the algorithm, one neighborhood is selected randomly, then the selected neighborhood is explored, and the best move is chosen. This move must not be tabu unless it improves the current best solution T^*. In Step 3, the new solution that has just been created from Step 2 is then added to a promising solution set P if its fitness value is not worse more than 10% to that of the T^*. Step 2 is executed for a number of times to find good promising solutions to insert to the set P. Step 4 uses elite solutions in P to exploiting the current solution space, thus enhances the solution quality. To explore more solution space, a diversification phase is added in Step 5. The search then restarts from the perturbed solution produced in Step 5. The search is stopped after *maxITER* number of iterations without improvement on the current best solution T^*.

Algorithm 1 The HVT algorithm

Input: v_1, V are the starting vertex and the set of vertices in K_n, respectively.
Output: the best solution T^*.

```
Promising solution set P ← ∅ ;
while stop criteria not met do
  {construction phase}
  Step 1 (Generate an initial solution):
  T ← Clustered-GRASP(v1, V);
  {Improvement phase}
  Step 2 (RVNS+TS):
  Initialize the Neighborhood List NL with 8 neighborhoods;
  Clear all tabu lists;
  while NL ≠ ∅ do
    Select a neighborhood N from NL at random;
    Find the best solution T' in the selected neighborhood N of T;
    if (T' is better than T and T' is not tabu) or (T' is better than T*) then
      T ← T';
      Update Tabu list;
      if (T' is better than T*) and (T' is a feasible solution) then
        T* ← T';
      end if
    else
      Remove N from the NL;
    end if
  end while
  Step 3 (Build a promising solution set P):
  if T is not worse than 1.1 times the best found solution T* then
    Add T to the set P;
  end if
  if (| P |< 5) then
    go to Step 2;
  end if
  Step 4 (Exploitation):
  T̄ ← the best solution in P
  while P ≠ ∅ do
    Select randomly a solution T (and remove it) from P;
    Perform Step 2 without using tabu restrictions on the solution T;
    if T is better than T̄ then
      T̄ ← T;
    end if
  end while
  Step 5 (Exploration):
  T ⟵ Shaking(T̄, num);
  go to Step 2;
end while
return T*;
```

In the following, the main components of our proposed algorithm (Algorithm 1) will be detailed. The search space is first presented in "Search space" section. The way to generate initial solution in Step 1 is then displayed in "The initial solution", "Neighborhoods of RVNS", "Tabu lists and tabu durations" section dedicate to explain the implementation of Step 2. They, thus,

Algorithm 2 Clustered-GRASP

Input: v_1, k, α are respectively the depot, the number of vehicles and the size of RCL.
Output: An initial solution T.

```
S = ∅; {S is the list of infeasible tours}
T = ∅;
{K_n^1, K_n^2, ..., K_n^k} = k-means(K_n, k);
STOP ← 0;
repeat
  for (l = 1; l <= k; l++) do
    R_l = {v_1}; {Every route of the tour T starts at the depot v_1}
  end for
  for each K_n^l do
    L = ∅; {L stores the list of visited vertices of K_n^l}
    while |L| < |K_n^l| do
      Create RCL with α vertices v_i ∈ K_n^l closest to v_t; {v_t is the last vertex in route R_l}
      FOUND ← 0;
      for (i = 1; i <= RCL; i++) do
        Select a vertex v = {v_i | v_i ∈ RCL and v_i ∉ R_l};
        Add v to the last position of the route R_l;
        if R_l is still feasible then
          FOUND ← 1;
          break;
        else
          Remove v from the route R_l;
        end if
      end for
      if not FOUND then
        Select randomly a vertex v ∈ RCL and v ∉ R_l;
        Add v to the last position of the route R_l;
      end if
      L = L ∪ v;
    end while
    T = T ∪ R_l; {Add the route R_l to the tour T}
  end for
  if T is a feasible solution then
    STOP ← 1;
  else
    Add solution T to the set S;
  end if
  if |S| = n then
    T ⟵ the solution with the minimum fitness L' in the set S;
    STOP ← 1;
  end if
until STOP
return T
```

describe the set of eight neighborhood structures used for RVNS and the tabu mechanism for each neighborhood, respectively. We detail in "Intensification and diversification" section the intensification (Step 4) and diversification (Step 5) phases.

Algorithm 3 Shaking(T, num)

Input: T, num are the tour and the number of swap, respectively.
Output: a new route T'.

```
Select randomly a route R_l in T;
Shaking-intra(R_l, num);
Select randomly two routes R_x and R_y in T;
Shaking-inter(R_x, R_y, num);
return T';
```

Algorithm 4 Shaking-intra(R_l, num)

Input: R_l, num are the l^{th} route and the number of exchange, respectively.
Output: a new route R_l.
while ($num > 0$) **do**
$p_1 = 1 + \text{rand}(|R_l| \ / \ 4)$;
$p_2 = p_1 + 1 + \text{rand}(|R_l| \ / \ 4)$;
$p_3 = p_2 + 1 + \text{rand}(|R_l| \ / \ 4)$;
$R_1 = R_l[1, ..., p_1]$;
$R_2 = R_l[p_3, ..., |R_l|]$;
$R_3 = R_l[p_2, ..., p_3]$;
$R_4 = R_l[p_1, ..., p_2]$;
$R_l = R_1 \cup R_2 \cup R_3 \cup R_4$; {Route R_l is the concatenation of 4 routes R_1, R_2, R_3, R_4}
$num = num - 1$;
end while
return R_l;

Algorithm 5 Shaking-inter(R_x, R_y, num)

Input: R_x, R_y, num are the x^{th}, y^{th} route, and the number of exchange, respectively.
Output: new routes R_x and R_y.
while ($num > 0$) **do**
$i = random(1, |R_x|)$; {Choose a position in the route R_x randomly}
$j = random(1, |R_y|)$; {Choose a position in the route R_y randomly}
Exchange vertices $R_x[i]$ and $R_y[j]$;
$num \longleftarrow num - 1$;
end while
return R_x and R_y;

Search space

For a given solution T, let $L(T)$ denote the total length of its routes and let $V(T)$ denote the total violation of vehicle length. The total vehicle-length violation $V(T)$ is computed on a route basis with respect to the value $D_{\max}$, thus is equal to $\sum_{R_l \in T} \max\{D_{\max} - L(R_l), 0\}$.

Solutions are then evaluated according to the weighted fitness function $L'(T) = L(T) + \rho * V(T)$, where ρ is the penalty parameter.

The initial solution

At Step 1 of Algorithm 1, we generate intial solution using the GRASP with clustering. The detail is described in Algorithm 2. The k-means [12] is first used to cluster the ADVRP problem to k smaller ADVRP problems. Specifically, the K_n is converted into k smaller complete graphs $K_n^1, K_n^2, ..., K_n^k$. Routing is then performed iteratively on each cluster $K_n^l (1 \le l \le k)$ to build a correspond route R_l. All routes are initialized with the main depot v_1. Each vertex of the K_n is then added to the routes by using its Restricted Candidate List (RCL). The RCL of a vertex includes a number of vertices which are closest to it. At an iteration, assuming vertex v_e be the current last vertex of the route R_l, an unvisited vertex v is picked randomly from its RCL to add to R_l such that this addition does not make the constraint violation. If there does not exist any such vertex v, we then accept the infeasibility. Thus, a not-yet-routed vertex v' is picked up randomly to add

to the last position of the route R_l. A solution is generated when all vertices of K_n are routed.

The procedure then returns the feasible solution if any. Otherwise, for added randomness in routing, it tries to generate n solutions, then the one with the minimum fitness value will be returned.

Neighborhoods of RVNS

The RVNS in Step 2 of Algorithm 1 exploits eight neighborhoods including different intra- and inter-route moves as following:

Five intra-route neighborhoods:

- Remove–insert move each vertex is shifted from its current location to the last position of the same route.
- Swap-adjacent move two adjacent vertices in the same route are exchanged.
- Swap move two vertices in the same route are exchanged.
- 2-opt move for each pair of vertices v_i, v_j in the same route, the edges emanating from them (v_i, v_i'), (v_j, v_j') are removed, two edges (v_i, v_j), (v_i', v_j') are added.
- 3-opt: for each three vertices v_i, v_j, v_k in the same route, the edges emanating from them (v_i, v_i'), (v_j, v_j'), (v_k, v_k') are removed, three adges (v_i, v_j'), (v_j, v_k'), (v_k, v_i') are added.

Three inter-route neighborhoods:

- Exchange-route move the two vertices of two different routes are exchanged.
- Insert-route move a vertex is shifted from its current position to another position of the other route.
- Cross-route move two segments $\{v_i, v_k\}$ and $\{v_j, v_l\}$ on two different routes are exchanged.

Tabu lists and tabu durations

The tabu lists for the eight moves described above are included in our algorithm. The solution elements receiving a tabu status following an intra-route move are:

- Remove–insert move the position of vertex v_i just moved at the end of the route cannot be changed by the same type of move while it is tabu;
- Swap-adjacent, swap move vertices v_i and v_j just swapped cannot be swapped again while they are tabu;
- 2-opt move a 2-opt move applied to vertices v_i and v_j cannot be applied again to the same vertices.
- 3-opt a 3-opt move applied to vertices v_i, v_j and v_k cannot be applied again to the same vertices.

while for inter-route moves:

- Exchange-route move vertices v_i in route R_l and v_j in route R_h cannot be swapped again by the same type of move while they are tabu.
- Insert-route move the position of vertex v_i in the route R_l just inserted to the position jth of the route R_h cannot be changed by the same type of move while it is tabu.
- Cross-route move two segments $\{v_i, v_k\}$ on R_l and $\{v_j, v_l\}$ on R_h cannot be swapped again by the same type of move while they are tabu.

A tabu status is assigned to an element for θ iterations, where θ is randomly selected from an uniform interval [13, 14]. At each iteration, a neighborhood is selected to explore. Its best neighbor T' is then accepted as the next starting solution if the corresponding move is non-tabu and T' is better than the current starting solution T. Otherwise, T' must be better than the current best solution T^*.

Intensification and diversification

We use a promising solution set P as a pool of high-quality solutions found during the algorithm. This set is initialized empty and limited in size. Each solution T produced by the RVNS+TS in Step 2 of Algorithm 1 is inserted into the set P if its fitness is not worse more than 10% to that of the current best solution T^*. Step 2 is thus executed for a number of times to find such solutions T for inserting them into the set P. Once the set P is full, the intensification phase (Step 4 of the Algorithm 1) will be triggered to enhance the quality of the solutions in P. Specifically, only the RVNS of Step 2 is applied to each solution in P. Without the tabu list restriction, the RVNS starts by applying a neighborhood selected randomly in the set NL of eight neighborhoods described in the "Neighborhoods of RVNS" section. The selected neighborhood is then searched on all possible moves, and the best neighbor is returned. The current solution will be modified if it is worse than its best neighbor. Otherwise, this selected neighborhood is removed from the set NL as it does not produce any improvement. The process is repeated until the set NL is empty. The best solution $\bar{T}$ produced by this intensification process will be served as the input for the diversification phase (Step 5 of the Algorithm 1).

Using only intensification, the search could get stuck in the local optima. Thus, diversification then proceeds to perturb the search that starts from the solution $\bar{T}$ by performing a combined shakings of the solution (see Algorithm 3). Creating a new working solution from the $\bar{T}$ helps to conserve the best solution characteristics encountered so far. On the other hand, implementing shaking process to perturb a new working solution provides a certain level of diversity to the search. In this work, there are two shaking mechanisms used to diversify the solution:

- Shaking intra-route: We use the shaking mechanism, called double-bridge, was originally developed by [15]. 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. The double-bridge shaking is described in the Algorithm 4.

– Shaking inter-route: We randomly choose two routes R_x and R_y in the solution, then exchange a number of vertices between them. The detailed description of the implementation is given in the Algorithm 5.

Evaluation

Our proposed algorithm is implemented in C++. Experiments are conducted on an Intel Pentium core i7 duo 2.10 Ghz CPU, 8 GB RAM. The performance of the proposed algorithm is evaluated through comparison with published results on the instances provided in the literature of ADVRP, ACVRP, MTRPD. Through preliminary experiments, we observed that the values $\alpha = 5$, num $= 5$, $\rho = 100$, $|E| = 5$, and $maxITER = 100$ resulted in a good trade-off between solution quality and running time.

Comparison with ADVRP's algorithms

The performance of our proposed algorithm (HVT) is evaluated through comparison with published results on the Group 1 instances of ADVRP provided in [16]. These instances are divided into three sets, denoted as $D_{\max(1)}$, $D_{\max(2)}$, $D_{\max(3)}$, in descending order of the upper bound value $D_{\max}$. The number of customers is in the range of [40, 1000]. All published results used for competing are taken from:

– M5SBB : The exact method (Multi Start Branch and Bound) of [16].
– VNS: The metaheuristic is developed on VNS framework of [16].

We report our best results over 10 runs. For concision sake, only aggregated results are provided in this section. A detailed comparison, instance by instance, may be found in Tables 4, 5, 6 of the Annex. Table 1 sums up comparison over all three sets. As the results of VNS for the $D_{\max(3)}$ set have not been published, we use the hyphen "-" to indicate its unavailable in Table 1. The first two rows respectively show for each competing algorithm, the percentage of times that algorithm could obtain the optimal solutions (%opt) and could not find feasible solutions (% of no feasible). Note that all current published optimal solutions and best known solutions (BKS) are produced by M5SBB [16]. The row Gap_1[%] indicates the average gaps of best solutions obtained by each algorithm with repect to the optimal solutions (OPT) for instances whose optimal solutions have been published. Similarly, the row Gap_2[%] displays the average gaps relative to BKS for instances whose the optimal solutions have not been found yet. More specifically, Gap_1[%] and Gap_2[%] in percentage on each instance are calculated as follows:

Table 1 Performance comparison for the ADVRP on 131 instances

	$D_{\max(1)}$		$D_{\max(2)}$		$D_{\max(3)}$	
	VNS	**HVT**	**VNS**	**HVT**	**VNS**	**HVT**
%opt	0	49.2	0	19.0	0	6.9
% of no feasible	0	0	14.29	0	–	0
Gap_1[%]	12.84	3.90	9.11	4.29	–	4.52
Gap_2[%]	0	3.1	0	23.8	0	17.24

$$\text{Gap}_1[\%] = \frac{\text{Best.Sol} - \text{OPT}}{\text{OPT}} \times 100\%, \tag{4}$$

$$\text{Gap}_2[\%] = \frac{\text{Best.Sol} - \text{BKS}}{\text{BKS}} \times 100\%. \tag{5}$$

One observes that only our algorithm could find feasible solutions in all cases. Indeed, for the easiest $D_{\max(1)}$ set with the largest value of upper bound $D_{\max}$, all algorithms could find feasible solutions. However, for $D_{\max}(2)$ set with smaller value of $D_{\max}$, the problem becomes harder to solve. As the result, the number of instances that the M5SBB could not find feasible solutions is three, and this value is even up to 19 (14.29%) for the VNS metaheuristic (see Table 5). On the hardest $D_{\max}(3)$ instances, the M5SBB could not find feasible solutions for three cases, while the results for the VNS has not been published due to its worse results on the $D_{\max(2)}$. Especially, there are 11 intances, though the M5SBB cannot reach the optimal solutions due to lack of memory, the HVT obtains better solutions (see Table 6).

Experimental results illustrated clearly the superior performance of our HVT algorithm compared to the VNS metaheuristic of [16]. Table 1 shows that the HVT provides high-quality solutions, with an average gap of 4.24% to the optimal solutions, compared to 10.97% of the VNS on the $D_{\max(1)}$, and $D_{\max(2)}$ dataset. Overall 131 instances, the HVT produces 124 better solutions than those of VNS, and 41 optimal solutions while the VNS could not find any. The VNS was executed on intel Core(TM) 2 CPU 6600 2.4GHz with 3.24 GB of RAM which has different characteristics than our machine; therefore, comparing the running time between two algorithms is evaluated relatively. As reported in [16], our running time is comparable with that of the VNS.

Comparison with ACVRP's algorithms

In this section, we present the performance comparison between our HVT algorithm and published results on the ACVRP instances provided by Pessoa et al. in [17]. The ACVRP is the Capacitated VRP in which the matrix cost is asymmetric. Table 2 displays the comparison between the best solutions obtained by our HVT algorithm and those obtained by two other algorithms:

Table 2 Experimental results for ACVRP instances

Instances	#vertices	#vehicles	OPT	AS	VL	HVT			
				Best.Sol	Best.Sol	Best.Sol	Aver.Sol	Gap_1	Time
A034-02f	34	2	1406	1406	1406	1406	1406.00	0.00	0.27
A036-03f	36	3	1644	1644	1644	1644	1644.00	0.00	0.31
A039-03f	39	3	1654	1654	1654	1654	1654.00	0.00	0.44
A045-03f	45	3	1740	1740	1740	1740	1740.00	0.00	0.52
A048-03f	48	3	1891	1891	1891	1891	1891.51	0.00	0.59
A056-03f	56	3	1739	1739	1739	1739	1739.00	0.00	0.63
A065-03f	65	3	1974	1974	1974	1976	1976.21	0.10	2.61
A071-03f	71	3	2054	2054	2121	2054	2054.51	0.00	2.80

- VL: a metaheuristic combines some heuristic concepts with compact mixed-integer linear programming (MILP) formulation, proposed by Valeria Leggieria and Mohamed Haouari in [18].
- AS: a column generation approach uses metaheuristic to generate columns and a sequence of set partitioning models for the Mixed-Integer Programming solver, proposed by A. Subramanian et al. in [19].

for each instance of the ACVRP.

The first three columns give the instance name, the number of vertices and vehicles, respectively. The column OPT indicates the optimal solutions reported by [20, 6]. We report for each algorithm the best solutions (*Best.Sol* column) found. The last three columns give our average results over 10 runs, the gap between our best found solution and the optimal solution, and our running time in seconds, respectively. One could observe that our HVT algorithm is comparable with the others in terms of solution quality and running time. It produces very near-optimal solutions, with an average gap of 0.01% to the optimal solutions.

Comparison with MTRPD's algorithms

In the MTRPD, a fleet of homogeneous vehicles is dispatched to serve a set of customers. Each customer is serviced exactly once by a vehicle. Each vehicle which starts from and ends at the depot is not allowed to travel a distance longer than a predetermined limit. The objective is to minimize the total waiting time of all customers after the vehicles leave the depot. Our algorithm runs on the MTRPD dataset inherited several instances in [20]. The optimal solutions for these instances are found using the exact algorithms in [20, 21]. The dataset includes six TSP instances from the TSPLIB such as brd14051, d15112, d18512, fnl4461, nrw1379, and pr1002. For each TSP instance, ten MTRPD instances are generated by randomly selecting ten subsets of n vertices, where $n = 30$, 40, 50, 60, 70, and 80. Let ($d_{\max}$) be the distance to the farthest vertices from the depot. The distance constraint gets the values $2 \times d_{\max}$, $2.5 \times d_{\max}$, and $3 \times d_{\max}$. The exact algorithms for these instances are extracted from [5, 15]. Therefore, in total, 900 MTRPD instances are used in our experiment.

For small instances of size 30–50 vertices, we compare our results to optimal solutions produced by [15]. However, for large instances with more than 50 vertices, there does not exist published optimal solutions. We thus compare our results to the lower bounds of the MTRPD which is the optimal solutions of the MTRP obtained in [12]. Table 3 displays our aggregated results over 900 instances in terms of CPU time (in seconds), gap to the optimal solution on small instances and gap to lower bound on large instances. Each row represents to a set of 10 instances. As shown in Table 3, our HVT algorithm is capable of finding the optimal solutions for all small instances in a reasonable amount of time (0.52 seconds on average). It is, thus, better than those obtained by GRASP+VNS in [13] which fails to find the optimal solutions for all instances with 50 vertices. Moreover, for most instances consisting of 60–80 vertices, our solutions fall into the range of 0.30–0.32% of the lower bound. Consequently, our HVT algorithm could produce much better results compared to that of [13] which ranges from 2.73 to 4.75%.

Table 3 The average experimental results for MTRPD instances

Instances	$MD = 2 \times d_{max}$		$MD = 2.5 \times d_{max}$		$MD = 3 \times d_{max}$	
	Gap_1	Time	Gap_1	Time	Gap_1	Time
pr1002_30_x	0.00	0.25	0.00	0.23	0.00	0.24
brd14051_30_x	0.00	0.25	0.00	0.24	0.00	0.25
fnl4461_30_x	0.00	0.22	0.00	0.22	0.00	0.24
d15112_30_x	0.00	0.25	0.00	0.23	0.00	0.21
nrw1379_30_x	0.00	0.25	0.00	0.25	0.00	0.24
pr1002_40_x	0.00	0.45	0.00	0.44	0.00	0.41
brd14051_40_x	0.00	0.44	0.00	0.42	0.00	0.41
fnl4461_40_x	0.00	0.44	0.00	0.44	0.00	0.44
d15112_40_x	0.00	0.44	0.00	0.41	0.00	0.44
nrw1379_40_x	0.00	0.43	0.00	0.42	0.00	0.42
pr1002_50_x	0.00	0.85	0.00	0.84	0.00	0.82
brd14051_50_x	0.00	0.81	0.00	0.82	0.00	0.81
fnl4461_50_x	0.00	0.83	0.00	0.83	0.00	0.83
d15112_50_x	0.00	0.83	0.00	0.83	0.00	0.85
nrw1379_50_x	0.00	0.84	0.00	0.84	0.00	0.82
pr1002_60_x	0.55	1.12	0.48	1.45	0.34	1.45
brd14051_60_x	0.67	1.13	0.46	1.45	0.54	1.45
fnl4461_60_x	0.48	1.12	0.39	1.45	0.29	1.45
d15112_60_x	0.57	1.11	0.39	1.46	0.31	1.46
nrw1379_60_x	0.64	1.12	0.71	1.45	0.76	1.45
pr1002_70_x	0.89	1.48	0.83	1.97	0.93	1.97
brd14051_70_x	0.51	1.49	0.54	1.84	0.51	1.84
fnl4461_70_x	0.63	1.46	0.60	1.94	0.67	1.94
d15112_70_x	0.70	1.51	0.74	1.85	0.76	1.85
nrw1379_70_x	0.65	1.44	0.73	1.95	0.62	1.95
pr1002_80_x	0.69	3.58	0.74	4.66	0.66	4.66
brd14051_80_x	0.58	3.56	0.63	4.62	0.57	4.62
fnl4461_80_x	0.74	3.59	0.80	4.63	0.73	4.63
d15112_80_x	0.41	3.57	0.39	4.65	0.43	4.65
nrw1379_80_x	0.96	3.53	0.86	4.67	0.82	4.67
Average	0.32	–	0.30	–	0.30	–

Conclusion

We have proposed a new effective metaheuristic algorithm for the ADVRP, which combines GRASP with clustering, tabu search and RVNS. Extensive computational experiments and comparisons with published algorithms on all benchmark instances show that the proposed algorithm is highly competitive. It could provide not only new best solutions but also first published feasible solutions for some instances with tight value of $D_{\max}$. Additionally, our algorithm provides better solutions than the state-of-the-art metaheuristic algorithm for 124 out of 131 cases. However, the efficiency and

running time for the large instances with up to 1000 vertices need to be improved. It is our aim for future research.

Acknowledgements
This work is implemented on the Machine from MSOLAB. We wishes to thank this lab for support. I also would like to thank EiC My T. Thai for her support.

Authors' contributions
The first author provided the ideas and implemented the algorithms. The second provided the ideas for the algorithms and verified their performance. All authors read and approved the final manuscript.

Funding
This paper was invited by EiC My T. Thai and will have fee waiver.

Availability of data and materials
The data is available from [8, 20, 17]. Authors are happy to support the data upon request.

Competing interests
The authors declare that they have no competing interests.

Annex

Tables 4, 5, 6 report the detailed comparisons, intance by intance, on the ADVRP problem between our HVT algorithm and two other algorithms:

- M5SBB : The exact method (Multi Start Branch and Bound) of [16].
- VNS: The metaheuristic is developed on VNS framework of [16].

Instances for experiments are provided in [16] where the customer range is [40, 1000]. All algorithms were run 10 times per instance. Our best results (Best.Sol column), average results (Aver.Sol column), gap between our best results and the optimal values (Gap_1 column), gap of our best results with respect to those of VNS (GAP Best-to-Best column), and computation time in seconds (Time column) are reported. Note that all optimal values (OPT column) are produced by M5SBB in [8, 16]. In cases M5SBB could not find optimal solutions, only best solutions produced by it are provided. We therefore put these values in parentheses in the OPT column. The value "inf" indicates that the algorithm could not find feasible solution.

Table 4 Experimental results for ADVRP instances with $D_{max(1)} = \infty$

# vertices	# vehicles	*p*	OPT (M5SBB)	VNS			HVT				
				Best.Sol	Gap_1	Time	Best.Sol	Aver.Sol	Gap_1	Gap Best-to-Best	Time
40	2	1	152	158	3.95	10	152	152	0.00	− 3.80	0.46
40	2	2	208	225	8.17	10	208	208	0.00	− 7.56	0.42
40	2	3	178	193	8.43	10	178	178	0.00	− 7.77	0.47
40	2	4	186	198	6.45	10	186	186	0.00	− 6.06	0.48
40	4	1	171	179	4.68	10.02	171	171	0.00	− 4.47	0.46
40	4	2	223	233	4.48	10	223	223	0.00	− 4.29	0.47
40	4	3	213	217	1.88	10	213	213	0.00	− 1.84	0.46
40	4	4	218	224	2.75	10	218	218	0.00	− 2.68	0.44
60	3	1	178	200	12.36	10.11	178	178	0.00	− 11.00	1.51
60	3	2	222	248	11.71	10	222	222	0.00	− 10.48	1.58
60	3	3	192	204	6.25	10.03	192	192	0.00	− 5.88	1.54
60	3	4	205	226	10.24	10.05	205	205	0.00	− 9.29	1.54
60	6	1	203	218	7.39	10.05	203	203	0.00	− 6.88	1.6
60	6	2	254	268	5.51	10.05	254	254	0.00	− 5.22	1.5
60	6	3	228	240	5.26	10.08	228	228	0.00	− 5.00	1.57
60	6	4	238	251	5.46	10.03	238	238	0.00	− 5.18	1.57
80	4	1	198	223	12.63	10.16	198	198	0.00	− 11.21	4.32
80	4	2	237	271	14.35	10.05	237	237	0.00	− 12.55	4.61
80	4	3	211	236	11.85	10.08	211	211	0.00	− 10.59	4.39
80	4	4	214	238	11.21	10.09	214	214	0.00	− 10.08	4.49
80	8	1	231	245	6.06	10.08	237	231	2.60	− 3.27	4.32
80	8	2	281	302	7.47	10	281	281	0.00	− 6.95	4.52
80	8	3	242	263	8.68	10.05	242	242	0.00	− 7.98	4.37
80	8	4	238	247	3.78	10.11	238	238	0.00	− 3.64	4.54
100	5	1	204	235	15.20	10.34	210	210	2.94	− 10.64	16.21

Table 4 (continued)

# vertices	# vehicles	*p*	OPT (M5SBB)	VNS			HVT				
				Best.Sol	Gap_1	Time	Best.Sol	Aver.Sol	Gap_1	Gap Best-to-Best	Time
100	5	3	228	268	17.54	10.08	238	238	4.39	− 11.19	16.01
100	5	4	225	252	12.00	10.22	247	247	9.78	− 1.98	17.55
100	10	1	232	251	8.19	10	232	232	0.00	− 7.57	17.63
100	10	2	309	336	8.74	10.02	311	311	0.65	− 7.44	17.74
100	10	3	270	295	9.26	10.14	270	270	0.00	− 8.47	16.17
100	10	4	255	279	9.41	10.13	255	255	0.00	− 8.60	16.8
120	6	1	232	270	16.38	10.02	253	254.3	9.05	− 6.30	72.1
120	12	1	269	304	13.01	10.09	278	278.3	3.35	− 8.55	71.21
120	12	2	329	367	11.55	10.05	329	329.5	0.00	− 10.35	71.01

Table 5 Experimental results for ADVRP instances with $D_{max(2)} = 0.9 \times LT(1)$

# vertices	# vehicles	*p*	OPT (M5SBB)	VNS			HVT				
				Best.Sol	Gap_1	Time	Best.Sol	Aver.Sol	Gap_1	Gap Best-to-Best	Time
40	2	1	inf	inf	inf	inf	157	157	inf	inf	0.45
40	2	2	inf	inf	inf	inf	214	214	inf	inf	0.44
40	2	3	(180)	188	4.44	100	176	176	2.22	− 6.38	0.47
40	2	4	187	202	8.02	100	187	187	0.00	− 7.43	0.47
40	4	1	(171)	177	3.51	100	169	169	1.17	− 4.52	0.43
40	4	2	224	228	1.79	100	224	224	0.00	− 1.75	0.45
40	4	4	inf	inf	inf	inf	223	223	inf	inf	0.46
60	3	2	222	241	8.56	100.05	230	230	3.60	− 4.56	1.54
60	3	3	192	213	10.94	100	198	198	3.13	− 7.04	1.46
60	3	4	205	218	6.34	100.03	205	205	0.00	− 5.96	1.46
60	6	1	203	217	6.9	100.02	215	215	5.91	− 0.92	1.53
60	6	2	(254)	265	4.33	100.06	252	252	0.79	− 4.91	1.57
60	6	3	(228)	234	2.63	100.05	221	221	3.07	− 5.56	1.6
60	6	4	238	254	6.72	100.02	238	238	0.00	− 6.30	1.45
80	4	1	198	228	15.15	100.05	219	219	10.61	− 3.95	4.56
80	4	2	237	260	9.7	100.02	249	249	5.06	− 4.23	4.59
80	4	3	211	233	10.43	100.03	211	211	0.00	− 9.44	4.45
80	4	4	215	250	16.28	100.03	215	215	0.00	− 14.00	4.29
80	8	1	231	252	9.09	100.03	239	239	3.46	− 5.16	4.35
80	8	2	(281)	302	7.47	100.05	276	276	1.78	− 8.61	4.66
80	8	3	242	255	5.37	100.2	261	261	7.85	2.35	4.32
80	8	4	238	251	5.46	100.19	238	238	0.00	− 5.18	4.62
100	5	2	256	287	12.11	100.17	279	279	8.98	− 2.79	17.6
100	10	1	232	259	11.64	100.08	238	238	2.59	− 8.11	16.36
100	10	2	309	331	7.12	100.08	315	315	1.94	− 4.83	16.53

Table 5 (continued)

# vertices	# vehicles	*p*	OPT (M5SBB)	VNS			HVT				
				Best.Sol	Gap_1	Time	Best.Sol	Aver.Sol	Gap_1	Gap Best-to-Best	Time
100	10	3	270	292	8.15	100.23	271	271	0.37	− 7.19	16.29
100	10	4	255	276	8.24	100.22	279	279	9.41	1.09	16.27
120	12	1	269	292	8.55	100.56	282	282	4.83	− 3.42	73.66
120	12	2	329	inf	inf	inf	331	331	0.61	inf	73.93
120	12	3	277	309	11.55	100.06	291	291	5.05	− 5.83	71.98
120	12	4	308	339	10.06	100.02	339	339	10.06	0.00	70.8
140	14	1	280	317	13.21	100.38	288	288.6	2.86	− 9.15	84.4
140	14	2	334	360	7.78	100.11	342	342.8	2.40	− 5.00	88.34
140	14	3	283	324	14.49	100.17	307	307.3	8.48	− 5.25	85.73
140	14	4	326	346	6.13	100.31	346	346.7	6.13	0.00	83.88
160	16	1	309	354	14.56	100.23	347	348.9	12.30	− 1.98	97.72
160	16	2	(365)	inf	inf	inf	346	347.6	5.21	inf	97.76
160	16	3	310	345	11.29	100.19	324	324.9	4.52	− 6.09	99.32
180	18	2	(392)	inf	inf	inf	369	374.6	5.87	inf	110.53
180	18	4	347	386	11.24	100.27	386	368.6	11.24	0.00	110.82
200	10	2	322	376	16.77	101.25	376	376.8	16.77	0.00	121.22
200	20	3	358	401	12.01	100.41	401	409.8	12.01	0.00	116.4

Table 6 Experimental results for ADVRP instances with $D_{max(3)} = 0.9 \times LT(2)$

# vertices	# vehicles	*p*	MS5BB (OPT)	HVT			
				Best.Sol	Aver.Sol	Gap_1	Time
40	4	1	171	171	171	0	0.55
60	3	2	(223)	225	225	0.9	1.65
60	3	3	(193)	195	195	1.04	1.54
60	6	1	(203)	206	206	1.48	1.54
60	6	2	254	244	244	0	1.58
60	6	4	(239)	229	229	0	1.43
80	8	1	231	236	236	2.16	4.78
80	8	2	(282)	271	271	0	4.98
80	8	3	(243)	243	243	0	4.88
80	8	4	238	261	261	9.66	4.77
100	5	1	inf	270	270	inf	17.41
100	5	2	256	Inf	inf	inf	17.98
100	10	1	232	253	253	9.05	17.12
100	10	2	309	299	299	0	17.23
100	10	3	(271)	276	276	1.85	17.54
100	10	4	(inf)	280	280	inf	17.12
120	12	1	269	303	303	12.64	74.32
120	12	2	(330)	333	333	0.91	73.93
120	12	4	308	346	346	12.34	71.32
140	14	1	280	310	312	10.71	84.65
140	14	2	334	347	348	3.89	89.65
140	14	4	326	354	356	8.59	84.65
160	16	1	309	346	348	11.97	95.65
160	16	2	365	377	379	3.29	94.68
160	16	4	(328)	365	367	11.28	105.65
180	18	2	392	408	408	4.08	98.65
180	18	3	(inf)	395	398	inf	114.65
180	18	4	347	394	395	13.54	99.65
200	20	2	394	424	427	7.61	114.65
200	20	4	364	414	418	13.74	165.23

Received: 24 June 2020 Accepted: 21 December 2020
Published online: 22 January 2021

References

1. Dantzig GB, Ramser JH. The truck dispatching problem. Managem Sci. 1959;6(1):80–91.
2. Lenstra JK, Rinnooy Kan AHG. Complexity of vehicle routing and scheduling problems. Networks. 1981;11(2):221–7.
3. Baldacci R, Toth P, Vigo D. Recent advances in vehicle routing exact algorithms. J Or. 2007;5(4):269–98.
4. Baldacci R, Mingozzi A, Roberti R. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. J Operat Res. 2012;218(1):1–6.
5. Laporte G. The vehicle routing problem: an overview of exact and approximate algorithms. J Operat Res. 1992;59(3):345–58.
6. Toth P, Vigo D. The Vehicle Routing Problem J. SIAM monographs on discrete mathematics and applications. Philadelphia: Society for Industrial and Applied Mathematics; 2002.
7. Laporte G, Nobert S, Taillefer S. A branch and bound algorithm for the asymmetrical distance-constrained vehicle routing problem. J Mathe Modell. 1987;9(12):857–68.
8. Almoustafa S, Hanafi S, Mladenovic N. New exact method for large asymmetric distance-constrained vehicle routing problem. J Operat Res. 2013;226(3):386–94.
9. Feo TA, Resende MGC. Greedy randomized adaptive search procedures. J Global Opt. 1995;6(2):109–33.
10. Glover F. Tabu Search. J Informs. 1990;2(1):4–32.
11. Mladenovic N, Hansen P. Variable neighborhood search. J Operat Res. 1997;24(11):1097–100.

12. Olafsson S, Lia X, Wua S. Operations research and data-mining. J Eur Operat Res. 2008;187(3):1429–48.
13. Ban HB. A GRASP+VNS algorithm for the multiple traveling repairman problem with distance constraints. J Jcsc. 2017;33(3):272–88.
14. D. S. Johnson, and L. A. McGeoch, "The traveling salesman problem: A Case Study in Local Optimization in Local Search in Combinatorial Optimization", E. Aarts and J. K. Lenstra, eds., pp. 215-310.
15. Martin O, Otto SW, Felten EW. Large-Step markov chains for the traveling salesman problem. J Complex Syst. 1991;5(3):299–32626.
16. Almoustafa S. Distance-constrained vehicle routing problem: exact and approximate olution (Mathematical programming). School of Information Systems, Computing and Mathematics: Brunel University; 2013 phD thesis.
17. Pessoa A, Uchoa E, Aragao PD. A robust branch-cut-price algorithm for the Heterogeneous fleet vehicle routing problem. J Netw. 2009;54(4):167–77.
18. Leggieri V, Haouari M. A matheuristic for the asymmetric capacitated vehicle routing problem. J Discrete Appl Mathe. 2018;234:139–50.
19. Subramanian A, Uchoa E, Ochi LS. Hybrid algorithm for a class of vehicle routing problems. J Comput Operat Res. 2013;40(10):2519–31.
20. Luo Z, Qin H, Lim A. Branch-and-Price-and-Cut for the multiple traveling repairman problem with distance constraints. J Operat Res. 2013;234(1):49–60.
21. Nucamendi-Guillen S, Martinez-Salazar I, Angel-Bello F, Moreno-Vega JM. A mixed integer formulation and an efficient metaheuristic procedure for the K-Travelling repairmen problem. J. Jors. 2016;67(8):1121–34.

Publisher's Note