Computing and Informatics, Vol. 42, 2023, 1184–1212, doi: 10.31577/cai_2023_5_1184

METAHEURISTIC FOR SOLVING THE DELIVERY MAN PROBLEM WITH DRONE

Ha-Bang Ban*, Hai-Dang Pham

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

Abstract. Delivery Man Problem with Drone (DMPD) is a variant of Delivery Man Problem (DMP). The objective of DMP is to minimize the sum of customers' waiting times. In DMP, there is only a truck to deliver materials to customers while the delivery is completed by collaboration between truck and drone in DMPD. Using a drone is useful when a truck cannot reach some customers in particular circumstances such as narrow roads or natural disasters. For NP-hard problems, metaheuristic is a natural approach to solve medium to large-sized instances. In this paper, a metaheuristic algorithm is proposed. Initially, a solution without drone is created. Then, it is an input of split procedure to convert DMP-solution into DMPD-solution. After that, it is improved by the combination of Variable Neighborhood Search (VNS) and Tabu Search (TS). To explore a new solution space, diversification is applied. The proposed algorithm balances diversification and intensification to prevent the search from local optima. The experimental simulations show that the proposed algorithm reaches good solutions fast, even for large instances.

Keywords: DMPD, metaheuristic, VNS

1 INTRODUCTION

Delivery Man Problem with Drone (DMPD) is a variant of Delivery Man Problem (DMP) that is seen as a "customer-centric" routing problem. However, in DMP, there is only a truck to deliver materials to customers. In the case of narrow or

* Corresponding author

congested roads and disasters, when a truck cannot reach some customers, the drone is used. Informally, in DMPD, there is a truck and drone at the main depot s, and n customers. The goal is to find a tour with a combination of truck and drone that minimizes the overall customers' waiting times while ensuring that all customers are served.

Using drone to deliver parcels to customers is not new in the literature. Many companies use drones for parcel delivery [1] because drone brings many advantages:

1. It can be operated on itself;
2. It is beneficial in the case of congestion;
3. It can move faster than trucks [2].

However, a drone also has some drawbacks:

1. The drone can carry parcels of 2 kilograms. It cannot carry packages that exceed its weight [3]. Therefore, it needs to return to the depot after each delivery;
2. Its range is limited because of the battery-powered engines. On the other hand, the truck has a long range and brings many materials but is rather heavy and slow.

Therefore, the collaboration between trucks and drones brings many advantages. The idea of the collaboration is that the delivery truck and drone collaboratively serve all customers. Figure 1 describes an example in which 8 customers need to be visited. We see that the drone visits two customers instead of the truck. The truck is not waiting and can serve another customer instead. The truck or drone that first reaches the reconnection node has to wait for each other. By the parallelization of delivery tasks, the total waiting time can be reduced.

To the best of our knowledge, though various works were proposed to solve DMP and its variants [4, 5, 6, 7, 8, 9, 10, 11, 12, 13], there is no work for DMPD in the literature. Due to the characteristics of the problem, adapting these metaheuristics for DMP to solve DMPD is not easy. We aim to develop an effective metaheuristic to solve the problem. The success of any metaheuristic approach depends on the right balance between intensification and diversification. The main contributions of this work can be summarized as follows:

- From the algorithmic perspective, the proposed metaheuristic consists of five steps. We introduce a split heuristic that builds a DMPD solution from a DMP solution. In the local search step, the VNS with new neighborhoods adapted from the traditional ones is proposed for DMPD. This step aims to exploit the explored solution space. Lastly, a diversification step is applied to lead our search to a new region. Moreover, Tabu Search is incorporated into our algorithm to prevent the search from local optima.
- From the computational perspective, extensive numerical experiments on benchmark instances show that our algorithm reaches good solutions fast, even for

large instances. In addition, the proposed algorithm is adapted to solve DMPD. The outcome shows that the proposed algorithm obtains good solutions fast. Its solutions are compared with the state-of-the-art metaheuristics' ones.

The rest of this paper is organized as follows: Sections 2, and 3 present the literature and the problem definition, respectively. Section 4 describes the proposed algorithm. Computational evaluations are reported in Section 5. Sections 6 and 7 discuss and conclude the paper, respectively.

2 LITERATURE

DMPD is NP-hard because it is a generalization case of DMP. Despite the similarities between DMP and DMPD, solving DMPD is more challenging due to drone's presence. Currently, there are three main methods to solve an NP-hard problem, specifically:

1. exact algorithms,
2. approximation algorithms, and
3. heuristic (or metaheuristic) algorithms.

The exact algorithms obtain the optimal solution, but they consume more time. Therefore, it can only solve the problems with small sizes. Meanwhile, for an α-approximation algorithm, the value of α is often large, and they are complex to implement. Metaheuristic, on the other hand, is naturally suitable to solve large-sized instances in a short time.

2.1 DMP

Though we found no work for DMPD, several classes of problems closely related to the problem were introduced in the literature: the Delivery Man Problem (DMP), DMP with Profits (DMPP), and DMP in post-disaster.

- Delivery Man Problem (DMP) is a particular case where DMPD has no drone. Numerous works for DMP can be found in [6, 7, 8, 9, 10, 11, 13]. Ban et al. [7] also present an exact algorithm to solve instances with up to 40 vertices. Several approximation algorithms [10, 11, 13] are proposed to solve DMP with the best ratio of 3.59. In the metaheuristic algorithms, these algorithms [6, 8, 9] obtain good solutions fast for instances with up to 1000 vertices.
- DMP with Profits (DMPP) aims to find a travel plan for a server that maximizes the total revenue. However, contrary to DMP, in DMPP, not all the customers need to be visited. Metaheuristic algorithms in [5, 14] can solve well the problem with up to 500 vertices. After that, Dewilde et al. [12] have introduced a stochastic variant of DMPP under uncertain travel times. The difference between DMPP and stochastic DMPP is that the travel times in stochastic DMPP

are uncertain. The stochastic DMPP is heuristically solved using a beam search heuristic.

- In the minimizing latency in the post-disaster road clearance operations (ML-RCP) [4], we find a tour to minimize latency sum in the post-disaster. Their metaheuristic algorithm reaches the optimal or near-optimal solutions on Istanbul data within seconds.

These algorithms are the state-of-the-art algorithms for DMP. However, there is no drone presence, and they cannot be adapted directly to DMPD. That means that we cannot use the above algorithms to solve DMPD.

2.2 Routing with Drone

The UAV for surveillance purposes has been used for a long time. However, in recent times, research for drone delivery has been improved by logistics companies. We describe the works related to routing with drone.

- Murray et al. [15] first introduced the TSP with drone (TSPD) in which there are a single truck and a single drone. In this paper, two constraints are considered. In the first constraint, the drone can deliver only one parcel at a time because of the capacity limit. The second one shows that a subset of the customers can be visited by a drone. They proposed a mixed integer programming formulation and a heuristic to minimize the completion time for all vehicles.
- Agatz et al. [16, 17] proposed TSPD with different constraints in comparison with Murray et al. in which the drone is allowed to launch and return to the same location (the constraint is forbidden in [15]). They developed a new dynamic programming approach to solve exactly the problem with up to 10 vertices.
- Freitas and Penna [18] proposed a metaheuristic with the min-time objective function. Firstly, a mixed-integer program (MIP) is used to obtain an initial solution. Then it is modified by removing some truck vertices and adding some drone ones. Finally, several neighborhoods are applied to find better solutions. The experiment shows that the delivery time is decreased up to 68 %.
- Poikoene et al. [19] proposed a branch-and-bound to solve TSPD in the case of allowing drone to revisit customers. They tested the proposed algorithm on small instances. The experimental results indicate that the problem can be solved optimally for the problem with 10 customers.
- Poikoene et al. [1] have extended TSPD with more constraints. Specifically, a drone can serve multiple customers in a row. Moreover, the capacity limit of a drone is also mentioned. They proposed a flexible heuristic that obtains good solutions fast. It showed that metaheuristic is a suitable approach for DMPD.
- Ha et al. [20] proposed a metaheuristic algorithm to minimize the total operational cost (the min-cost TSPD). They proposed the greedy randomized adaptive search procedure (GRASP) to minimize the total operational cost (the min-cost

TSPD) [20]. The results are quite good in terms of solution quality and running time. They then presented a hybrid genetic algorithm (GA) [21] with some new strategies to control diversity. The results are provided for instances with up to 50 and 100 customers.

- TSPD with more than one drone is also interested by many researchers in literature [22, 23, 24, 25, 26, 27].

The above works are the state-of-the-art algorithms to solve TSPD. DMPD in this paper is different from TSPD. Our objective function is to minimize the total waiting time of customers (customer-oriented routing), while their works aim to minimize the minimum travel cost (server-oriented routing). The difference between the two objective functions leads to a big difference in their solutions. In [28], Salehipour et al. showed that the good algorithms for the TSP cannot be adapted for DMP. Therefore, developing an efficient algorithm for DMPD is necessary.

3 PROBLEM DEFINITION

We are given a complete graph $K_n = (V, E)$ with $V = V' \cup \{v_1\}$ where v_1 is a depot and $V' = V^t \cup V^d \cup V^c$ in which V^t, V^d, V^c are the set of vertices visited by only truck, only drone, and both of them, respectively. Let $C = \{c_{ij} \mid i, j = 1, \ldots, n\}$ be the distance matrix between all vertices. The travel time between a pair of vertices (v_i, v_j) is the distance between v_i and v_j (c_{ij}). Let $\rho = \frac{\beta}{\alpha}$ be the ratio between the drone's and truck's travel time per unit distance. Every customer can be served by either a truck or drone. A drone can only deliver one parcel at a time. We make the following assumptions:

- A truck can dispatch and pick up a drone only at the depot or at a customer location.
- Customers can only be served once, either by a drone or a truck.
- DMPD allows the parallelization of delivery tasks. That means the drone departs from the truck at the customer i. It then drops off a parcel at the second customer j and goes to the connection customer k. The drone cannot serve more than one customer between nodes i and j.
- The vehicle (truck or drone) that first arrives at the reconnection node has to wait for the other one.
- When returning to the truck to take another parcel, the time required to prepare the drone launch is negligible.

Our objective is to minimize the total waiting time of all customers.

Because of the presence of the truck and the drone, a DMPD solution $T = (TR, DR)$ is represented by two components:

- A truck tour, denoted as TR, is a sequence of vertices: $TR = \{v_1, v_2, \ldots, v_k, \ldots\}$.

- A drone tour, denoted as DR, is a sequence of tuple $< i, j, k >$. That means drone launches from v_i, delivers to v_j, and rejoins truck at v_k.

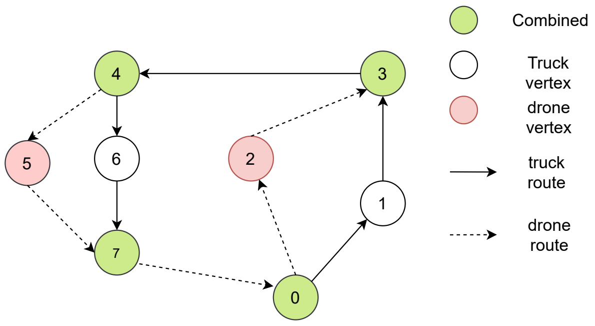

Figure 1. The simple example of DMPD

After that, we describe a simple example of DMPD. There are three types of vertices. A truck vertex is a vertex that is delivered by truck. Similarly, a drone vertex is a vertex delivered by a drone. A combined vertex is a vertex visited by both the truck and drone. For example, there are 7 vertices in the graph in which vertex 1 and 6 are truck vertices, while vertex 2 and 5 are drone vertices. Last, vertex 3, 4, and 7 are combined vertices. The truck moves from the depot to vertex 1 while the drone is flown from the depot to vertex 2. After that, truck travels from vertex 1 to 3, and drone moves from vertex 2 to 3. The truck and drone are rejoined at vertex 3. The truck has to wait for the drone if it goes to vertex 3 before the drone. On the other hand, the drone has to wait for the truck if it goes to vertex 3 before the truck. The truck and drone travel together to vertex 4. The drone is launched toward vertex 5 while the truck goes to vertices 4 to 6. The truck and drone are rejoined at vertex 7. The waiting time of vertices is calculated as follows:

$$
\begin{aligned}
w(v_0) &= 0,\\
w(v_1) &= \alpha \times c(v_0, v_1),\\
w(v_2) &= \beta \times c(v_0, v_2),\\
w(v_3) &= w(v_1) + \alpha \times c(v_1, v_3),\\
w(v_4) &= \max\{w(v_1) + \alpha \times c(v_1, v_3), w(v_2) + \beta \times c(v_2, v_3)\} + \alpha \times c(v_3, v_4),\\
w(v_5) &= w(v_4) + \beta \times c(v_4, v_5),\\
w(v_6) &= w(v_4) + \alpha \times c(v_4, v_6),
\end{aligned}
$$

$$w(v_7) = w(v_6) + \alpha \times c(v_6, v_7),$$

$$W(T) = \sum_{i=0}^{7} w_i.$$

If the drone first arrives at v_3, v_7, it has to wait for the truck because the drone cannot serve more than one customer between nodes i and j. It means that customers at v_3, v_7 must be served by the truck. The waiting time of customer at v_4 is calculated more complex than the others because of rendezvous time.

The objective function in this example is different from the one of TSPD when it optimizes the following function cost: $\max\{\alpha \times (c(v_0, v_1) + c(v_1, v_3)), \beta \times (c(v_0, v_2) + c(v_2, v_3))\} + \alpha \times c(v_3, v_4) + \max\{\alpha \times (c(v_4, v_6) + c(v_6, v_7)), \beta \times (c(v_4, v_5) + c(v_5, v_7))\} + \alpha \times c(v_7, v_0)$.

4 A METAHEURISTIC ALGORITHM

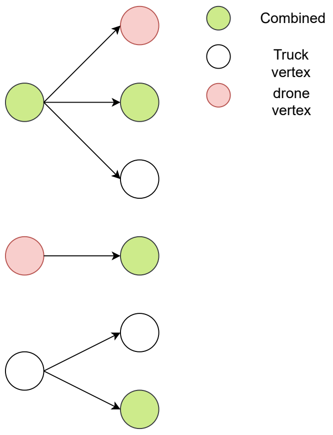

Figure 2. Label assignment for a vertex

The algorithm is divided into five steps: In the first step, a solution without the drone is created. Then, it is an input of splitting procedure to convert DMP-solution into DMPD-solution in the second. After that, it is improved by the combination of Variable Neighborhood Search (VNS) and Tabu Search (TS) in the local search. When the proposed algorithm gets stuck into local optima, diversification is used. It leads the search to unexplored solution space. Algorithm 1 depicts the whole process. In line 2, we build a giant DMP tour T_{TR} visiting the depot and all the customers. For that aim, we apply some heuristics for construction. As a result, an initial solution T_{TR}, where all vertices are visited by truck and no vertex is assigned to drone. Line 5 decomposes T_{TR} in two partitions: the first assigned to

Algorithm 1 General scheme of algorithm

```
Input: $v_1, V, N_i(T) (i = 1, \dots, 6), level$ are a starting vertex, the set of vertices in
    $K_n$, the set of neighborhoods, the parameter to control the strength of the
    perturbation procedure, respectively.
Output: the best solution $T^*$.
1: {Step 1: Construction-DMP}
2: $T_{TR} \leftarrow$ construction-DMP($v_1, V, N_i(T), K_n$); {$T_{TR}$ is DMP tour}
3: {Step 2: Split phase}
4: $T = (T_t, T_d) \leftarrow$ split($T_{TR}$); {Assign vertices into "drone", "truck", or "combined"
   label}
5: $T^* \leftarrow T$;
6: while The stop condition is not satisfied do
7:    $T^{'} \leftarrow T$;
8:    {Step 3: Local search}
9:    $k = 1$;
10:   repeat
11:      Find the best neighborhood $T^{''}$ of $T \in N_k(T^{'})$;
12:      if $((W(T^{''}) < W(T^{'})$ and ($T^{'}$ is not Tabu)) $\|$ $(W(T^{''}) < W(T^*))$ then
13:         $T^{'} = T^{''}$;
14:         Update Tabu list;
15:         if $((W(T^{''}) < W(T^*))$ then
16:            $T^* = T^{''}$;{Update the best solution}
17:         $k = 1$;
18:      else
19:         $k = k + 1$; {switch to another neighborhood}
20:   until $k < k_{max}$;
21:   {Step 4: Intensification}
22:   if $W(T^{''}) < W(T)$ then
23:      $\overline{T}$ = Perform VNS without using tabu on the solution $T^{''}$;
24:      if $(W(T^{''}) < W(\overline{T}))$ or $(W(T^*) < W(\overline{T}))$ then
25:         $T^{''} \leftarrow \overline{T}$;
26:         if $(W(T^*) < W(\overline{T}))$ then
27:            $T^* \leftarrow \overline{T}$; {Update the best solution}
28:   $T \leftarrow T^{''}$;
29:   {Step 5: Diversification}
30:   $T \longleftarrow$ Perturbation($T, level$);
31: return $T^*$;
```

Algorithm 2 GRASP + VNS

```
Input: $v_1, V, \gamma$ are a starting vertex, the set of vertices in $K_n$, and the size of $RCL$,
   respectively.
Output: the initial solution $T$.
  $T \leftarrow v_1$;
  while $|x| < n$ do
    Create RCL with $\gamma$ vertices $v_i \in V$ closest to $v_l$; // $v_l$ is the last vertex in $T$;
    Select randomly vertex $v = \{v_i | v_i \in RCL$ and $v_i \notin T\}$;
    $T \leftarrow T \in \{v\}$;
  repeat
    $T^{'} \leftarrow$ Shaking($T$);
    $T^{''} \leftarrow$ arg min $N_k(T^{'})$; {local search}
    if $(L(T^{''} < L(T))$ then
      $T \leftarrow T^{''}$ {update solution}
    else
      $k = k + 1$; {switch to another neighborhood}
  until $k = k_{max}$
  return $T$;
```

truck (T_t) and the second assigned to drone (T_d). Lines 9–22 optimize the tour T by using six neighborhoods in local search. In lines 23–33, the VNS step is implemented without Tabu in the intensification step. Line 15 implements Perturbation to maintain diversity. The algorithm is repeated until the stop condition is satisfied.

4.1 Construction-DMP

In the first step, we generate a DMP solution from some heuristics. In this paper, three heuristics and an exact algorithm are used as follows:

- k-nearest neighbor [29]: It is inspired by the well-known nearest-neighbor algorithm. It begins from the root and repeatedly travels each vertex that is chosen among k closest vertices randomly.
- Insertion heuristic [29]: It works similarly to the above heuristic, but it picks a random vertex among all unvisited nodes iteratively.
- GRASP with VNS: The version of GRASP construction phase [30] is used to create an initial solution, and then it is improved by the VNS [31, 30]. Initially, at each vertex, a Restricted Candidate List including its nearest vertices is built. At a constructive step, a vertex is selected randomly from RCL, and added into the solution. The GRASP stops when all vertices are visited. After that, the GRASP's solution is shaken by swapping vertices randomly before the VNS step is used to improve the solution. The shaking aims to maintain the diversity of our search. The VNS is based on a simple principle that systematically switches between different neighborhoods. In the VNS, we use some popular

Algorithm 3 Greedy approach for splitting

```
Input: $v_1, V$ are a starting vertex and graph, respectively.
Output: the best solution $T^*$.
1: $v_1.label$ = "combined";
2: $TR = TR \cup \{v_1\}$;
3: $i = 0$;
4: while $\exists v$ is unvisited do
5:    $v = T[i]$;
6:    if ($v.label$ == "truck") then
7:       $rd$ =random(2);
8:       if ($rd$ ==1) then
9:          $v'.label$ = "truck";
10:         $TR = TR \cup \{v'\}$;
11:      else
12:         $v.label$ = "drone";
13:         $TD = TD \cup \{v_1\}$;
14:   else if ($v.label$ == "drone") then
15:      for $k \leftarrow i + 1$ to $n$ do
16:         find a $v' = \{v_j | v_j$ is unvisited and $T(i-1, j, k) = \min_{j=i+1}^{k-1} T(i, j, k)\}$;
17:      $v.label$ = "combined";
18:   else
19:      $rd$ =random(2);
20:      if ($rd$ ==1) then
21:         $v.label$ = "truck";
22:      else
23:         $v.label$ = "drone";
24:   $T = T \cup \{v_1\}$;
25: return $T^*$;
```

neighborhoods such as remove-insert (N_1), swap (N_2), and 2-opt (N_3) [30]. The combination between the GRASP and VNS demonstrated the efficiency in [8]. The GRASP + VNS scheme is described in Algorithm 2.

- Exact algorithm: We can use an exact algorithm in [7] to obtain the optimal solution for DMP. However, the exact algorithm can solve the problem with small sizes (less than 40 vertices).

4.2 Split Scheme

In this step, two approaches are used: Greedy heuristic and exact partitioning algorithm based on dynamic programming. All approaches remain in the sequence of vertices visited.

Algorithm 4 Dynamic-Programming for splitting(K_n, C)

```
Input: $K_n, C$ are the graph, and the distance matrix, respectively.
Output: A matrix $M$, a list of drone vertices $DM$.
1: for $i \leftarrow 1$ to $n$ do
2:    for $j \leftarrow 1$ to $n$ do
3:       $M[i,j] \leftarrow 0$;
4:       $DM[i,j] \leftarrow 0$;
5: for $i \leftarrow 1$ to $n$ do
6:    for $j \leftarrow 1$ to $n$ do
7:       $TT \leftarrow -Inf$;
8:       for $k \leftarrow i$ to $j$ do
9:          $TTd = \beta \times (c(v_i, v_k) + c(v_k, v_j))$.
10:         $TTt = \alpha \times (\sum_{h=i}^{k-2} c(v_h, v_{h+1}) + \sum_{h=k+1}^{j-1} c(v_h, v_{h+1}))$.
11:         if $(TT \leq \max\{TTd, TTt\})$ then
12:            $TT = \max\{TTd, TTt\})$
13:            $DM[i,j] = k$; {drone position is stored}
14:      $M[i,j] = TT$;
15: $MT[1] \leftarrow 0$;
16: $DM = \phi$;
17: for each $v_i$ in V do
18:    $w$=Inf;
19:    for $k \leftarrow 1$ to $i$ do
20:       if $(MT[k] + M[k,i] < w)$ then
21:          $w = MT[k] + M[k,i]$;
22:          $DP = k$;
23:    $DM = DM \cup DP$; {$DM$ is a list of drone positions}
24:    $MT[i] = w$;
25:    Assign a vertex at $DP$ position as “drone” label.
26: return $DM$;
```

4.2.1 Greedy Approach

We propose a greedy heuristic to split an initial T_{TR} into two subtours:

1. a subtour for the drone;
2. and a subtour for the truck.

Each vertex $v \in V$ is assigned by a label. The “truck” label indicates the truck, while the “drone” label indicates the drone. Moreover, a vertex that both the truck and drone can travel is called “combined”. Our greedy approach includes two phases. An initial solution consisting of the truck and drone is generated in the first phase, while “drone” label assignment optimization is done in the second. In this first stage, the label of all vertices is “unvisited”. In every heuristic step, a vertex is assigned to the “truck” or “drone” label. The algorithm stops when there is no

```
Algorithm 5 Perturbation($T$, $level$)
Input: $T, k$ are the tour, and the number of swap, respectively.
Output: a new tour $T$.
 1: $TR$, $TD$ = get($T$);
 2: while ($level > 0$) do
 3:     {Perturbation for truck}
 4:     $i_{TR}$ = rand($TR$);
 5:     $j_{TR}$ = rand($TR$);
 6:     $TR$=swap($i_{TR}, j_{TR}, TR$);
 7:     {Perturbation for drone}
 8:     $i_T D$ = rand($TD$);
 9:     $j_T D$ = rand($TD$);
10:     $TD$ = swap($i_{TD}, j_{TD}, TD$);
11:     $level = level - 1$;
12: $T = (TR, TD)$;
13: return $T$;
```

"unvisited" vertex. The detail of the Split Algorithm is described in Algorithm 3. To assign a label for a current vertex v, we need to consider its previous one:

- If a previous vertex is "combined" one, there are three label candidates ("truck", "drone", and "combined") for the current vertex v (see Figure 2 a)). Which label brings the best total cost for the tour will be selected.
- If a previous vertex is the truck vertex, there are two options ("truck", and "combined") for the current vertex v (see in Figure 2 b)). We choose a label such that the tour receives the best cost.
- If a previous vertex is the drone vertex, the label of the current vertex is "combined" (see Figure 2 c)). This is the only option in this case.

After the first step, we optimize the "drone" label assignment in the second one. An operation is defined by a triplet (i, j, k), which starts at v_i, delivers at v_k by drone, and rejoins at v_j. We try to enumerate the different cases resulting from changing a set of v_k candidates. After that, a v_k is picked so that our solution reaches the best cost. For example, in Figure 3, the drone and truck can rejoin at v_3, v_4, and v_n. Vertex v_4 is chosen because rejoining at this vertex makes our solution's cost decrease. The time complexity of the greedy heuristic is $O(n^2)$.

4.2.2 Dynamic Approach

We now describe an algorithm to partition an initial solution $T = \{v_1, v_2, \ldots, v_k, \ldots, v_n\}$ into a truck tour TR and a drone tour TD based on dynamic programming. The order of vertices in T remains unchanged. For each vertex, the algorithm decides to visit it by the truck or drone. A move is defined by a triplet (i, j, k), which starts at v_i, delivers at v_k by the drone, and rejoins at v_j. If a move does not contain the drone, we set k to -1. We also define $T(i, j, k)$ by the maximum amount of time to complete (i, j, k):

$$T(i,j,k) = \max\left\{\beta \times (c(v_i, v_k) + c(v_k, v_j)), \alpha \times \left(\sum_{h=i}^{k-2} c(v_h, v_{h+1})\right)\right.$$
$$\left. + \sum_{h=k+1}^{j-1} c(v_h, v_{h+1}) + c(v_{k-1}, v_{k+1})\right\}.$$

For each subsequent move $(v_i, v_{i+1}, \dots, v_j)$ in the tour, the minimum time is calculated as follows:

$$T(i,j) = \min_{k=i+1}^{j-1} T(i,j,k).$$

The recursive formulation for computing the minimum arrival time for the truck to reach a vertex is:

$$W(0) = 0,$$
$$W(v_i) = \min_{k=0}^{i-2}(W(v_k) + T(k, i-1) + \alpha \times c(v_{i-1}, v_i)).$$

The $\arg\min_{k=0}^{i-2}(W(v_k) + T(k, i-1) + \alpha \times c(v_{i-1}, v_i))$ is used to determine which of vertex is to be visited by the drone or truck.

4.3 Local Search

In this step, we apply the VNS [32, 33] for a full solution $T = (TR, TD)$. Neighbor solutions are evaluated, and the best feasible neighboring solution is accepted if it is non-tabu, improving, or tabu but globally improving. Six of our neighborhoods are inspired by traditional moves. We then describe these neighborhoods:

- Exchange role (N_1) exchanges the drone vertex to the truck vertex and vice versa. An example is shown in Figure 3.
- Swap (N_2) tries to swap the positions of each pair of vertices in T. An example is shown in Figure 4.
- Reverse-Swap (N_4) removes each pair of vertices from T and reverses the sub-tour between them. An example is shown in Figure 5.
- 2-opt (N_4) removes each pair of vertices from T and reverses the sub-tour between them. An example is shown in Figure 6.
- Add-drone (N_6) selects a truck location and replaces its delivery with drone delivery. An example is shown in Figure 7.

- Remove-drone (N_5) selects a drone location and replaces its delivery with a truck delivery. An example is shown in Figure 8.

After the local search step, its cost value is less than the current best value the new best value is updated. Moreover, all moves are updated in Tabu list.

4.4 Intensification and Diversification

When we find a good solution space, we try to exploit it in an intensification step. To exploit good solution space, we implement Step 2 without any tabu move. After that, the algorithm goes to the perturbation step to maintain diversification. A mechanism design plays an important role in obtaining success. If too small moves are applied, the search may get stuck into the previously visited solution space. On the other hand, excessive moves may drive the search to unexpected regions. In this work, we use two different perturbation methods for truck and drone. A perturbation method is simple to exchange some vertices visited by truck and drone for truck and drone, respectively. The detail of the step is described in Algorithm 5.

4.5 Tabu Lists

A tabu list for the moves is included in the proposed algorithm. The tabu status is assigned to an element for θ iterations, where θ is randomly selected. We describe the tabu list for each move as follows:

Exchange role: A position of exchanged vertices cannot be implemented by the same type of move if it is tabu.

Swap move: Two vertices swapped cannot be swapped again if they are tabu.

2-opt and reverse moves: The moves applied to two vertices cannot be applied again to the same positions.

Remove-drone: A "drone" vertex remains a "drone" label if it is tabu.

Add-drone: A "truck" vertex remains a "truck" label if it is tabu.

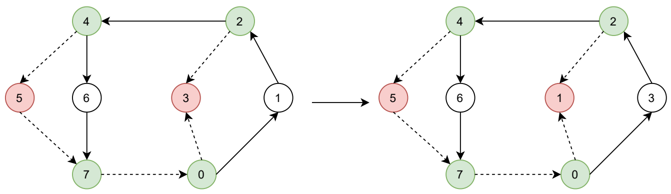

Figure 3. The exchange role

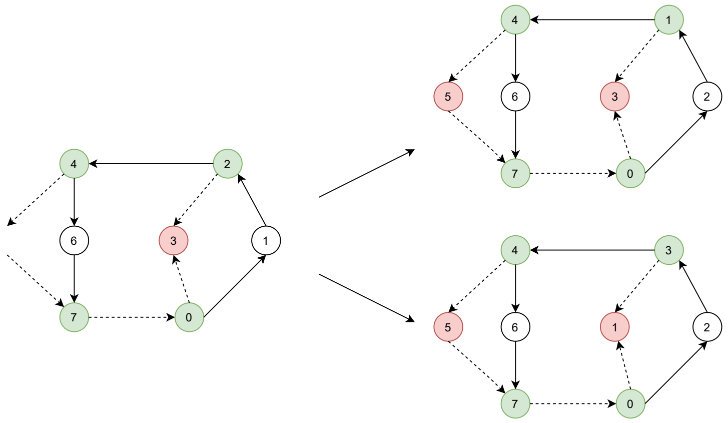

Figure 4. The swap

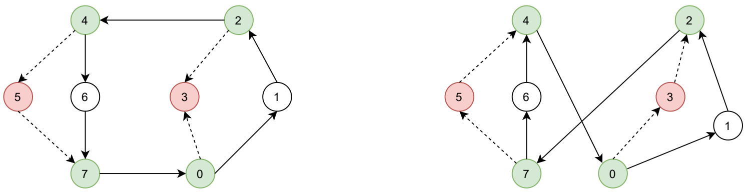

Figure 5. The reverse

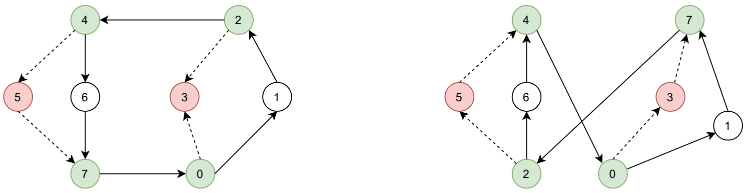

Figure 6. The 2-opt

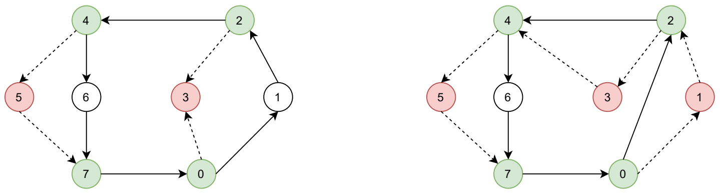

Figure 7. The drone insertion

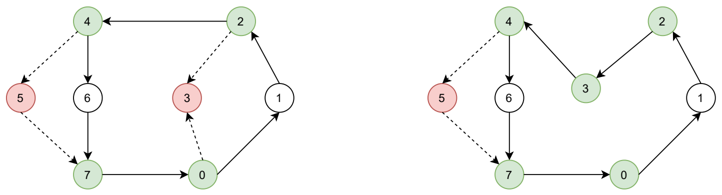

Figure 8. The drone removal

5 EVALUATIONS

Our algorithm is run on a Pentium 4 core i7 3.40 GHz 8 GB RAM processor. For all experiments, the parameters γ, *level*, and θ are set to 10, 5, and 5, respectively. These parameters are selected using empiric experiments and, with them, the algorithm might provide good solutions.

5.1 Instances

The proposed algorithm is tested on 900 instances in two datasets as follows:

- The first dataset is proposed by Poikonen et al. which are available on [19, 1]. This dataset includes 100 instances with up to 39 vertices. Customer locations in each instance are generated randomly from 50-by-50 grid. Moreover, the truck traveling time to move from customer i to j is calculated by $\lfloor |x_i - x_j| + |y_i - y_j| \rfloor$ while drone travel time is calculated by $\left\lfloor \frac{\sqrt{((x_i-x_j)^2+(y_i-y_j)^2)}}{\rho} \right\rfloor$. The speed of drone is ρ times faster than truck speed. To evaluate the efficiency of the proposed algorithm with different speeds of truck and drone, our algorithm is tested with ρ values $=$ 1, 2, and 3.
- The second dataset is released by Bouman [34]. This dataset consists of three types. In the first type (the uniform instances), the x and y coordinates are generated uniformly from $[0, 100]$. For the second type (the 1-center), an angle ϕ and distance r is created uniformly. The x and y coordinates are computed as $r \times \cos(\phi)$, and $r \times \sin(\phi)$. By doing this, locations close to the center $(0, 0)$ with higher probability than in the previous case. Finally, we have the 2-center, which is generated in the same way, but every location is transformed into 200 distance units over the x coordinate with probability 0.5. These instances have a greater probability of closing to two centers at $(0, 0)$ and $(200, 0)$. In all cases, the drone is ρ times as fast as the truck.

5.2 Results

The improvement of the proposed algorithm is defined considering *Best.Sol* (*Best.Sol* is the best solution found by our algorithm) in comparison with the other algorithms as follows:

$$Gap_1[\%] = \left|\frac{Best.Sol - BKS}{BKS}\right| \times 100\,\%, \tag{1}$$

$$Gap_2[\%] = \left|\frac{Best.Sol - DMPS}{Best.Sol}\right| \times 100\,\%, \tag{2}$$

$$Improv[\%] = \left|\frac{Best.Sol - Init.Sol}{Init.Sol}\right| \times 100\,\%. \tag{3}$$

In all tables, *Init.Sol*, *Best.Sol*, *Aver.Sol*, T correspond to the initial, best, average solution, and average time in seconds of ten executions obtained by the proposed algorithm, respectively, while *BKS* is the best-known solution in the literature. DMPS corresponds to DMP solution in the first step. The experiment results can be found in Tables 1 to 18 in [35]. The values in Tables 1, 2 and 3 in this paper are the average values calculated from Tables 1 to 18 in [35].

Currently, no metaheuristic for this problem can be found in the literature to compare directly. Therefore, it is difficult to evaluate the efficiency of the proposed algorithm exactly. To overcome the issue, several state-of-the-art metaheuristic algorithms for TSPD are chosen to compare to our algorithm. Fortunately, Bouman et al. [36], and Murray et al. [15] support their code for solving TSPD. However, their algorithms are developed for TSPD rather than DMPD. We adapt their objective function to solve DMPD. Therefore, the proposed algorithm and their algorithms can be compared directly. Moreover, our solutions are compared to some algorithms [36, 15, 37] in the case of TSPD. All algorithms are run on the same instances.

Different experiments have been carried out to evaluate the efficiency of the proposed algorithms as well as analyze the impact of parameters: investigate the performance of different construction heuristics for DMP, explore the performance of different heuristics to convert from DMP to DMPD; compare DMPD solutions with solutions in the case of an only truck, analyze the impact of drone in developing the quality of solution, consider DMPD with some constraints, compare our solution quality with the previous algorithms in the case of TSPD.

5.3 The Impact of Different DMP's Construction Heuristics for DMPD

In this experiment, we evaluate the performance of different construction heuristics for DMP's solution. The experimental results can be found in Table 1.

Overall, all heuristics work with stable results. The GRASP with VNS provided the best solution quality in comparison with the other heuristics. Specifically, in Table 1, the average gap of k-nearest neighbor and Insertion heuristic are $-21.93\,\%$ and $-21.44\,\%$, respectively, while the average one of GRASP + VNS is $-25.71\,\%$.

It is understandable because GRASP with VNS balances diversification and intensification while the others only maintain intensification. In addition, we carried out some additional tests and found that, in general, using the optimal DMP tours does not obtain the better solution quality for DMPD while it consumes more time to solve. From these analyses, we decided to use GRASP with VNS to generate DMP tours in the next experiments.

α, β	Instance	NN		IH		GRASP	
		Gap_2	T	Gap_2	T	Gap_2	T
1, 1	Uniform	−12.77	2.09	−13.26	2.56	−18.58	2.83
	SingleCenter	−14.56	2.19	−13.44	2.53	−23.21	2.53
	DoubleCenter	−15.85	2.07	−15.37	2.61	−18.37	2.63
1, 2	Uniform	−22.09	2.12	−22.01	2.60	−23.29	2.64
	SingleCenter	−24.51	2.17	−22.56	2.53	−24.68	2.50
	DoubleCenter	−24.97	2.05	−23.98	2.71	−19.66	2.69
1, 3	Uniform	−27.06	2.12	−26.96	2.72	−36.19	2.71
	SingleCenter	−27.08	2.17	−27.10	2.54	−34.53	2.52
	DoubleCenter	−28.49	2.05	−28.30	2.81	−32.89	2.81
aver		−21.93	2.11	−21.44	2.62	−25.71	2.65

Table 1. Experiment results with three construction methods

[

α, β	Instance	diff [%]	Split	DP
			Time	Time
1, 1	Uniform	0.00	2.83	6.83
	SingleCenter	−0.85	2.53	5.53
	DoubleCenter	−0.77	2.63	5.63
1, 2	Uniform	−0.53	2.64	6.64
	SingleCenter	−0.41	2.50	6.50
	DoubleCenter	−0.73	2.69	7.01
1, 3	Uniform	0.00	2.71	6.71
	SingleCenter	−0.17	2.52	5.52
	DoubleCenter	−0.30	2.81	6.81
aver		−0.42	2.65	6.35

Table 2. Experiment results between Exact and GRASP Construction Methods

5.4 The Impact of the Method to Assign Drone Vertices

In this experiment, we evaluate the performance of two methods to assign drone vertices. The experiment results can be found in Table 2.

For using two methods, using the dynamic programming approach helps the proposed algorithm to obtain a slightly better solution than using the greedy approach when the average difference of diff [%] between them is only below 0.42 %

Instances	$\alpha = 1, \beta = 1$			$\alpha = 1, \beta = 2$			$\alpha = 1, \beta = 3$		
	Gap_2	*Improv*	T	Gap_2	*Improv*	T	Gap_2	*Improv*	T
Uniform	−18.58	−4.96	2.09	−23.29	−10.24	2.12	−36.19	−23.91	2.12
SingleCenter	−23.21	−2.58	2.19	−24.68	−4.46	2.17	−34.53	−14.47	2.17
DoubleCenter	−18.37	−4.27	2.07	−19.66	−5.78	2.05	−32.89	−19.55	2.05
aver	−20.05	−3.94	2.12	−22.54	−6.83	2.11	−34.54	−19.31	2.11

Table 3. Experiment results with various the values of α and β

α, β	Instances	Murray et al.			Agatz et al.		
		better	equal	worse	better	equal	worse
1, 1	Uniform	81	10	9	65	12	22
	SingleCenter	83	3	14	67	10	23
	DoubleCenter	85	5	10	60	24	26
1, 2	Uniform	73	0	27	26	42	32
	SingleCenter	79	0	21	51	14	35
	DoubleCenter	66	0	34	45	9	44
1, 3	Uniform	72	0	28	18	64	18
	SingleCenter	79	0	21	21	40	39
	DoubleCenter	77	1	22	16	70	14
Sum		695	19	186	369	286	253

Table 4. Comparisons with two Murray et al.'s and Agatz et al.'s algorithms

(see Table 2). It shows that the greedy approach is also good for assigning drone vertices. More interestingly, while using a dynamic program consumes more time for large instances, the greedy approach spends less time on these cases. Thus, to balance solution quality and running time, using the greedy approach is a suitable choice, especially for large instances.

5.5 The Impact of the Improvement Phase

In this experiment, we evaluate the performance of the improvement phase in developing our solution quality. The experiment results can be found in Table 3.

It is shown that the difference in the average gap between the construction and improvement phases is from −3.94 % to −19.31 %. The average gap is rather large. It indicates the good efficiency of the improvement phase. However, for the larger instances with up to 100 vertices, our search is quite time-consuming. Hence, to reduce the running time, we can only run the construction phase with a loss of −10.03 % solution quality on the overall average.

5.6 Comparison of DMPD Solutions with DMP Solutions

In this experiment, we compare the results of DMP with the drone and DMP without the drone. The experiment results can be seen in Table 3.

Instances	Best.Sol	Aver.Sol	Gap_1	T
uniform-51-maxradius-60	1 322.29	1 335.04	16.74	0.28
uniform51-maxradius100	1 322.29	1 322.29	16.74	0.08
uniform51-maxradius150	1 322.29	1 325.83	16.74	0.07
uniform52-maxradius60	1 135.77	1 186.17	48.17	0.08
uniform52-maxradius100	1 172.43	1 256.52	52.96	0.06
uniform52-maxradius150	1 172.43	1 215.11	52.96	0.05
uniform53-maxradius100	952.87	1 051.09	18.44	0.08
uniform53-maxradius150	973.23	1 046.33	20.97	0.05
uniform54-maxradius60	1 450.59	1 457.04	56.78	0.07
uniform54-maxradius100	1 433.15	1 447.19	54.90	0.16
uniform54-maxradius150	1 433.15	1 449.20	54.90	0.06
uniform55-maxradius60	1 481.55	1 481.55	33.22	0.06
uniform55-maxradius100	1 444.72	1 513.20	29.90	0.06
uniform55-maxradius150	1 444.72	1 499.50	29.90	0.08
uniform56-maxradius60	1 530.57	1 530.57	50.58	0.14
uniform56-maxradius100	1 512.88	1 524.67	48.84	0.09
uniform56-maxradius150	1 530.57	1 530.57	50.58	0.11
uniform57-maxradius100	885.07	885.07	19.23	0.08
uniform58-maxradius1000	1 291.83	1 318.55	27.42	0.08
uniform58-maxradius150	1 291.83	1 291.83	27.42	0.06
uniform60-maxradius40	1 169.87	1 169.87	26.26	0.05
uniform60-maxradius60	1 162.91	1 162.91	25.51	0.07
uniform60-maxradius100	1 162.91	1 175.47	25.51	0.08
uniform60-maxradius150	1 162.91	1 162.91	25.51	0.06
aver			34.59	34.59

Table 5. Experimental results for DMPD with the range constraint

By using the drone, the proposed algorithm reaches better solutions than by using the truck only. Specifically, the average gap between DMPD's and DMP's solution quality is from $-18.58\,\%$ to $-34.54\,\%$. The large difference shows the important role of the drone in improving the solution quality. In all cases, the waiting time of customers is improved though the speed of the drone and truck is equal. Intuitively, when two vehicles deliver simultaneously, the waiting time of clients certainly is improved.

5.7 Comparison of DMPD with Different Speeds of the Drone

In this experiment, we evaluate the impact of different drone speeds on the results of DMPD. The experiment results can be seen in Table 3.

To evaluate the impact of different drone speeds, the ratio $\rho = \frac{\beta}{\alpha}$ is changed from one to three. For example, the ratio 2 means the drone travels at twice the speed of as truck. The results show that all of the proposed algorithms obtain better solutions

Instances	Best.Sol	Aver.Sol	Gap_1	T
uniform_51_novisit_20_rep_1	1 322.29	1 330.55	53.02	0.25
uniform_51_novisit_20_rep_2	1 135.77	1 206.62	31.44	0.18
uniform_51_novisit_20_rep_3	952.87	1 013.97	10.27	0.52
uniform_51_novisit_20_rep_4	1 433.15	1 440.84	65.85	0.10
uniform_51_novisit_20_rep_5	1 444.72	1 490.37	67.19	0.05
uniform_51_novisit_20_rep_6	1 512.88	1 519.95	75.08	0.12
uniform_51_novisit_20_rep_7	885.07	885.07	2.42	0.06
uniform_51_novisit_20_rep_8	1 345.26	1 345.26	55.68	0.05
uniform_51_novisit_20_rep_9	1 162.91	1 186.46	34.58	0.04
uniform_51_novisit_30_rep_10	1 322.29	1 330.55	53.02	0.10
uniform_51_novisit_30_rep_11	1 135.77	1 222.79	31.44	0.10
uniform_51_novisit_30_rep_12	952.87	1 012.38	10.27	0.20
uniform_51_novisit_30_rep_13	1 433.15	1 453.21	65.85	0.12
uniform_51_novisit_30_rep_14	1 444.72	1 478.96	67.19	0.14
uniform_51_novisit_30_rep_15	1 512.88	1 524.67	75.08	0.06
uniform_51_novisit_30_rep_16	885.07	885.07	2.42	0.05
uniform_51_novisit_30_rep_17	1 162.91	1 170.76	34.58	0.06
uniform_51_novisit_40_rep_18	1 322.29	1 334.32	53.02	0.08
uniform_51_novisit_40_rep_19	1 135.77	1 236.68	31.44	0.10
uniform_51_novisit_40_rep_20	1 038.13	1 068.81	20.14	0.12
uniform_51_novisit_40_rep_21	1 359.89	1 411.81	57.37	0.16
uniform_51_novisit_40_rep_22	1 444.72	1 513.20	67.19	0.07
uniform_51_novisit_40_rep_23	1 512.88	1 518.77	75.08	0.08
uniform_51_novisit_40_rep_24	885.07	885.07	2.42	0.05
uniform_51_novisit_40_rep_25	1 291.83	1 305.19	49.50	0.05
uniform_51_novisit_40_rep_26	1 162.91	1 173.38	34.58	0.05
aver			43.31	0.11

Table 6. Experimental results for DMPD with the INCOMP constraint

when the speed of drone increases. Specifically, the drone with double speed further reduces completion times by at least 2 % points, while the drone 3 times as fast as the truck adds at least another 12 % points improvement on average. Note, that in practice, we expect the drone to reach speeds that are more than the speed of the truck[1]. This means that for realistic drone speeds, our heuristics provide very good solutions.

5.8 Comparison with the Other Algorithms for DMPD

In this experiment, we compare the results of the proposed algorithm with the others [36, 15]. The experiment results can be seen in Table 4. In Table 4, each column includes three subcolumns that have "better", "equal", and "worse" labels.

[1] `https://www.droneomega.com/how-fast-do-drones-fly/`

Instances	P. Bouman et al.	Murray et al.	Best.Sol
uniform-1-n5	158.65	227.72	158.65
uniform-2-n5	199.07	199.07	199.07
uniform-3-n5	159.65	174.60	159.65
uniform-4-n5	136.10	181.84	136.10
uniform-10-n5	181.50	181.50	181.50
uniform-11-n6	140.55	171.70	140.55
uniform-67-n20	284.46	368.68	340.04
uniform-77-n50	500.55	677.10	564.70
singlecenter-28-n7	133.05	247.25	216.53
singlecenter-33-n8	141.47	166.90	153.21
singlecenter-93-n100	970.58	1 220.91	970.58
singlecenter-94-n100	985.60	1 497.03	985.60
doublecenter-65-n20	504.41	646.16	579.13
doublecenter-86-n75	895.32	1 310.30	895.32
doublecenter-87-n75	982.08	1 342.53	982.08
doublecenter-93-n100	1 057.31	1 479.56	1 057.31
doublecenter-94-n100	1 100.50	1 532.66	1 100.50
doublecenter-95-n100	1 189.19	1 819.39	1 189.19
doublecenter-96-n100	1 163.53	1 705.61	1 163.53

Table 7. Comparisons of the proposed algorithm with the others for TSPD $(\alpha, \beta = 1, 1)$

The "better", "equal", and "worse" labels mean that our solutions are better, equal, and worse in comparison with the other algorithms.

The experimental results for DMPD in Table 4 show that in 900 instances, our algorithm finds 674 better and 207 worse solutions than Murray et al.'s algorithm [15] while it reaches 369 better and 253 worse solutions than Agatz et al.'s algorithm [34]. We also use the Wilcoxon test to evaluate the hypothesis that the proposed method generates statistically better results than the other methods. The results indicate that the proposed algorithm shows a significant improvement over Murray et al.'s algorithm with a level of significance $\alpha = 0.05$. However, since the p-value was larger than the significance level (0.05), it indicates that no statistically significant improvements are detected between our best solutions with Agatz et al.'s ones. Nevertheless, 655 out of 900 better or the same solutions found in comparison with Agatz et al.'s algorithm is still beneficial.

5.9 Results for Variants of DMPD

In this paper, we consider the problem with additional constraints that are:

- Original DMPD: DMPD without any constraints.
- DMPD with INCOMP: A subset of customers must not be visited by the drone. This constraint can be found in Ha et al. [20, 21].

Instances	P. Bouman et al.	Murray et al.	Best.Sol
uniform-1-n5	228.57	227.72	158.65
uniform-14-n6	170.97	162.55	140.02
uniform-44-n9	239.15	222.10	217.63
uniform-52-n10	209.31	229.01	205.55
singlecenter-14-n6	238.41	104.31	99.82
singlecenter-33-n8	249.88	166.90	153.21
singlecenter-92-n100	912.50	1 312.59	1 161.43
singlecenter-99-n100	695.77	1 059.70	979.62
doublecenter-42-n9	483.48	468.60	480.16
doublecenter-68-n20	558.53	698.35	675.46
doublecenter-69-n20	626.51	730.97	712.96
doublecenter-70-n20	531.49	715.15	703.48
doublecenter-71-n50	921.26	1 053.45	1 062.32
doublecenter-72-n50	776.98	1 169.53	1 105.39
doublecenter-73-n50	697.26	990.38	990.96
doublecenter-74-n50	754.47	1 112.06	754.47
doublecenter-75-n50	876.80	1 164.56	1 209.75
doublecenter-76-n50	916.30	1 107.57	916.30
doublecenter-77-n50	754.65	1 065.48	754.65
doublecenter79-n50	794.50	1 068.85	968.70
doublecenter80-n50	879.11	1 245.45	879.11
doublecenter81-n75	969.61	1 265.04	1 305.23

Table 8. Comparisons of the proposed algorithm with the others for TSPD ($\alpha, \beta = 1, 2$)

α, β	Instances	Our algorithm	
		% Gap_1	T
1, 1	poi-10-x	−12.77	2.09
	poi-20-x	−14.56	2.19
	poi-30-x	−15.85	2.07
	poi-40-x	−22.09	2.12
	aver	−24.51	2.17
1, 2	poi-10-x	−12.77	2.09
	poi-20-x	−14.56	2.19
	poi-30-x	−15.85	2.07
	poi-40-x	−22.09	2.12
	aver	−24.51	2.17
1, 3	poi-10-x	−12.77	2.09
	poi-20-x	−14.56	2.19
	poi-30-x	−15.85	2.07
	poi-40-x	−22.09	2.12
	aver	−24.51	2.17

Table 9. Comparisons with Roberti and Ruthmair's algorithm for small instances

- DMPD with Range: The flying range of the drone is limited. This constraint is mentioned in Ha et al. [20, 21].

The constraints are popular when there are many drone-incompatible customers or the limit of drone's range. In this experiment, the common method is to add penalty values to the cost function for each infeasible solution to solve the problems with additional constraints. Therefore, the proposed algorithm is adjusted to adapt to these constraints. In the construction phase, we try to find a feasible solution. If it finds any feasible one, the solution is an input for the improvement phase. Otherwise, the penalty value is added to the original objective function. The advantage of the penalty technique is a simple implementation. With a tour T, let $V(T)$ be the violation. The violation value $V(T)$ is computed as follows:

- For DMPD with INCOMP:

$$V(T) = \max\{d(T), 0\}.$$

- For DMPD with Range:

$$V(T) = \sum^{<i,j,k>\in DR} \max\{c_{ij} + c_{jk} - 2 \times Rd, 0\}.$$

$d(T)$, Rd are the number of vertices that must not be served by the drone but receiving the drone delivery (violation) and the operation range of the drone, respectively. Solutions are then evaluated according to the weighted fitness function $L^{'}(T) = L(T) + \rho * V(T)$, where ρ is the penalty parameter.

Tables 5 and 6 show the results of DMPD with Range, and DMPD with INCOMP. As we know, the constraints make the problem more difficult. In some cases, finding a feasible solution is a challenge. However, the proposed algorithm finds feasible solutions fast for many cases. The differences in the average gap between DMPD's solution and DMPD with Range and INCOMP are 34.59 %, and 43.51 %, respectively. It indicates that the constraints strongly impact the results.

5.10 Results for TSPD

From Tables 7, 8 and 9, we tested the proposed algorithm for TSPD. Our algorithm still runs well for TSPD, although it was not designed to solve it. In comparison with the metaheuristic algorithms of Murray et al. [15] and Agatz et al. [16], our results are better than Murray et al.'s ones and are comparable with Agatz et al.'s ones in many cases. In addition, we run the proposed algorithm with small instances in [37]. Specifically, the proposed algorithm can find near-optimal solutions for the instances with 40 vertices. In many cases, the exact algorithm in [37] fails to find the optimal solution because of memory and time limits. On the other hand, the proposed algorithm reaches better solutions in these cases.

The algorithms in[16, 15] are executed on the same configuration. Therefore, it is convenient for us to compare the running time exactly. From the experimental results, the running time of the proposed algorithm is comparable with the other algorithms.

6 DISCUSSIONS

In this paper, we propose the hybrid approach between VNS with Tabu and Shaking, as follows. Firstly, the VNS maintains intensification by generating many good locally optimal solutions around the optimal global solution while the Shaking tries to maintain diversification. The combination balances diversification and intensification. Secondly, TS prevents the search from getting trapped into cycles. It helps the search to drive to the optimal global solution.

Considering the experiments, the proposed algorithm is tested with various scenarios: investigate the performance of different construction heuristics, explore the performance of different split heuristics, analyze the impact of drone on solution quality, and compare our solution quality with the previous algorithms. From the experimental results, we conclude:

- DMP's solution quality is relatively important to generate an initial solution for DMPD. In some methods, the metaheuristic (GRASP with VNS) obtains better solutions than the others. In small instances, the exact algorithm can be used to reach the optimal solutions for DMP. However, in most cases, the exact solutions for DMP cannot help the proposed algorithm find better solutions for DMPD while it consumes more time. The GRASP with VNS is the best choice in terms of solution quality and running time.
- The split step takes an important role in the final result. Using dynamic programming can receive a very slightly better than the greedy approach but it consumes much time for large instances. On the other hand, the greedy approach balances between solution quality and running time well. Therefore, the first strategy for decreasing the running time is using the greedy method with a loss of $-10.03\,\%$ solution quality on average.
- The results demonstrate that drone supports us in improving solution quality in comparison with truck only. The higher the speed of drone is, the more the total waiting time of clients decreases. In addition, the proposed algorithm finds feasible solutions for DMPD with constraints. The experimental results show that the constraints strongly affect the solution quality. It implies that the problem becomes much more complex when the constraints are involved.
- The algorithm finds better solutions than Murray et al. [15], and Agatz et al. [16] in the case of DMPD. The proposed algorithm is also tested with TSPD's instances. The results show that the proposed algorithm is comparable with the other algorithms [36, 15]. In addition, the proposed algorithm can find nearly

optimal solutions for the instances with 40 vertices. In many cases, the proposed algorithm finds better solutions in these cases, while Roberti and Ruthmair's algorithm [37] fails because of memory or time limits. Therefore, the proposed algorithm can be applied to TSPD well, though, it is not designed to solve it.

7 CONCLUSIONS

In this paper, we introduce DMPD. The main contribution is to propose a metaheuristic algorithm that balances intensification and diversification for solving DMPD. In the first step, a solution without the drone is generated by some heuristics. Then, it is an input of split procedure to convert it into DMPD-solution. After that, it is improved by the combination of Variable Neighborhood Search (VNS) and Tabu Search (TS). We implemented the algorithm on the benchmark dataset to compare it to several state-of-the-art metaheuristics. The proposed algorithm obtains better solutions in comparison with the other algorithms in many cases. Our algorithm is also tested with TSPD. The results show that our solutions can be compared with the previous algorithms, and the proposed algorithm can find near-optimal solutions for instances with up to 40 vertices in a short time. However, the running time of the algorithm can be improved to meet practical situations. In addition, applying to some different neighborhoods will be investigated in future research. Moreover, some other aspects will be considered to make the problem a more practical situation. In the first aspect, truck uses gasoline to move. It can run a long distance. Therefore, we do not consider the energy consumption of truck in this paper. Otherwise, the drone needs to return to the truck after each delivery to recharge energy and its range is limited because of battery-powered engines. Nevertheless, we assume that the drone always recharges energy when needed. In the second aspect, multiple drones and trucks deliver packets at the same time. Considering these aspects will be our aim in future work.

Acknowledgements

This research was supported by the Hanoi University of Science and Technology under grant number T2021-PC-021.

REFERENCES

[1] Poikonen, S.—Golden, B.—Wasil, E. A.: A Branch-and-Bound Approach to the Traveling Salesman Problem with a Drone. INFORMS Journal on Computing, Vol. 31, 2019, No. 2, pp. 335–346, doi: 10.1287/ijoc.2018.0826.

[2] Burgess, M.: DHL's Parcelcopter Drone Can Make Drops Quicker Than a Car. WIRED UK, 2016, `https://www.wired.co.uk/article/dhl-drone-delivery-germany`.

[3] Popper, B.: Hydrogen Fuel Cells Promise to Keep Drones Flying for Hours. The Verge, 2015, `https://www.theverge.com/2015/12/15/10220456/intelligent-energy-hydrogen-fuel-cell-drone`.

[4] Ajam, M.—Akbari, V.—Salman, F. S.: Minimizing Latency in Post-Disaster Road Clearance Operations. European Journal of Operational Research, Vol. 277, 2019, No. 3, pp. 1098–1112, doi: 10.1016/j.ejor.2019.03.024.

[5] Avci, M.—Avci, M. G.: A GRASP with Iterated Local Search for the Traveling Repairman Problem with Profits. Computers and Industrial Engineering, Vol. 113, 2017, pp. 323–332, doi: 10.1016/j.cie.2017.09.032.

[6] Ban, H. B.—Nguyen, D. N.: Improved Genetic Algorithm for Minimum Latency Problem. Proceedings of the 1st Symposium on Information and Communication Technology (SoICT '10), ACM, 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 the Minimum Latency Problem. Progress in Informatics, Vol. 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. Fundamenta Informaticae, Vol. 156, 2017, No. 1, pp. 21–41, doi: 10.3233/FI-2017-1596.

[9] Ban, H. B.: A Metaheuristic for the Delivery Man Problem with Time Windows. Journal of Combinatorial Optimization, Vol. 41, 2021, No. 4, pp. 794–816, doi: 10.1007/s10878-021-00716-2.

[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. 44th Annual IEEE Symposium on Foundations of Computer Science, 2003, pp. 36–45, doi: 10.1109/SFCS.2003.1238179.

[12] Dewilde, T.—Cattrysse, D.—Coene, S.—Spieksma, F. C. R.—Vansteenwegen, P.: Heuristics for the Traveling Repairman Problem with Profits. Computers and Operations Research, Vol. 40, 2013, No. 7, pp. 1700–1707, doi: 10.1016/j.cor.2013.01.003.

[13] Goemans, M.—Kleinberg, J.: An Improved Approximation Ratio for the Minimum Latency Problem. Mathematical Programming, Vol. 82, 1998, No. 1-2, pp. 111–124, doi: 10.1007/BF01585867.

[14] Beraldi, P.—Bruni, M. E.—Laganà, D.—Musmanno, R.: The Risk-Averse Traveling Repairman Problem with Profits. Soft Computing, Vol. 23, 2019, No. 9, pp. 2979–2993, doi: 10.1007/s00500-018-3660-5.

[15] Murray, C. C.—Chu, A. G.: The Flying Sidekick Traveling Salesman Problem: Optimization of Drone-Assisted Parcel Delivery. Transportation Research Part C: Emerging Technologies, Vol. 54, 2015, pp. 86–109, doi: 10.1016/j.trc.2015.03.005.

[16] Agatz, N.—Bouman, P.—Schmidt, M.: Optimization Approaches for the Traveling Salesman Problem with Drone. Transportation Science, Vol. 52, 2018, No. 4, pp. 965–981, doi: 10.1287/trsc.2017.0791.

[17] Bouman, P.: Code for the Traveling Salesman Problem with Drone. 2018, `https://github.com/pcbouman-eur/Drones-TSP`.

[18] de Freitas, J. C.—Penna, P. H. V.: A Variable Neighborhood Search for Flying Sidekick Traveling Salesman Problem. International Transactions in Operational Research, Vol. 27, 2019, No. 1, pp. 267–290, doi: 10.1111/itor.12671.

[19] Poikonen, S.—Wang, X.—Golden, B.: The Vehicle Routing Problem with Drones: Extended Models and Connections. Networks, Vol. 70, 2017, No. 1, pp. 34–43, doi: 10.1002/net.21746.

[20] Ha, Q. M.—Deville, Y.—Pham, Q. D.—Hà, M. H.: On the Min-Cost Traveling Salesman Problem with Drone. Transportation Research Part C: Emerging Technologies, Vol. 86, 2018, pp. 597–621, doi: 10.1016/j.trc.2017.11.015.

[21] Ha, Q. M.—Deville, Y.—Pham, Q. D.—Hà, M. H.: A Hybrid Genetic Algorithm for the Traveling Salesman Problem with Drone. Journal of Heuristics, Vol. 26, 2020, No. 2, pp. 219–247, doi: 10.1007/s10732-019-09431-y.

[22] Daknama, R.—Kraus, E.: Vehicle Routing with Drones. CoRR, 2017, doi: 10.48550/arXiv.1705.06431.

[23] Dorling, K.—Heinrichs, J.—Messier, G. G.—Magierowski, S.: Vehicle Routing Problems for Drone Delivery. IEEE Transactions on Systems, Man, and Cybernetics: Systems, Vol. 47, 2017, No. 1, pp. 70–85, doi: 10.1109/TSMC.2016.2582745.

[24] Kim, S.—Moon, I.: Traveling Salesman Problem with a Drone Station. IEEE Transactions on Systems, Man, and Cybernetics: Systems, Vol. 49, 2019, No. 1, pp. 42–52, doi: 10.1109/TSMC.2018.2867496.

[25] Schermer, D.—Moeini, M.—Wendt, O.: The Traveling Salesman Drone Station Location Problem. In: Le Thi, H. A., Le, H. M., Pham Dinh, T. (Eds.): Optimization of Complex Systems: Theory, Models, Algorithms and Applications (WCGO 2019). Springer, Cham, Advances in Intelligent Systems and Computing, Vol. 991, 2020, pp. 1129–1138, doi: 10.1007/978-3-030-21803-4_111.

[26] Mbiadou Saleu, R. G.—Deroussi, L.—Feillet, D.—Grangeon, N.—Quilliot, A.: An Iterative Two-Step Heuristic for the Parallel Drone Scheduling Traveling Salesman Problem. Networks, Vol. 72, 2018, No. 4, pp. 459–474, doi: 10.1002/net.21846.

[27] Wang, X.—Poikonen, S.—Golden, B.: The Vehicle Routing Problem with Drones: Several Worst-Case Results. Optimization Letters, Vol. 11, 2017, No. 4, pp. 679–697, doi: 10.1007/s11590-016-1035-3.

[28] Salehipour, A.—Sörensen, K.—Goos, P.—Bräysy, O.: Efficient GRASP + VND and GRASP+VNS Metaheuristics for the Traveling Repairman Problem. 4OR, Vol. 9, 2011, No. 2, pp. 189–209, doi: 10.1007/s10288-011-0153-0.

[29] Johnson, D. S.—McGeoch, L. A.: The Traveling Salesman Problem: A Case Study in Local Optimization. Chapter 8. In: Aarts, E. H. L., Lenstra, J. K. (Eds.): Local Search in Combinatorial Optimization. John Wiley & Sons Ltd., 1997, pp. 215–310.

[30] 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.

[31] Belošević, I.—Ivić, M.: Variable Neighborhood Search for Multistage Train Classification at Strategic Planning Level. Computer-Aided Civil and Infrastructure Engineering, Vol. 33, 2018, No. 3, pp. 220–242, doi: 10.1111/mice.12304.

[32] Hansen, P.—Mladenović, N.—Todosijević, R.—Hanafi, S.: Variable Neighborhood Search: Basics and Variants. EURO Journal on Computational Optimization, Vol. 5, 2017, No. 3, pp. 423–454, doi: 10.1007/s13675-016-0075-x.

[33] 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.

[34] Bouman, P.: TSP-D-Instances. 2015, `https://github.com/pcbouman-eur/TSP-D-Instances`.

[35] Appendix-DMPD.pdf. `https://drive.google.com/file/d/16SmLpCUs_nekhO5_yaCkWoiPWgpBaZB5/view?usp=sharing`.

[36] Bouman, P.—Agatz, N.—Schmidt, M.: Dynamic Programming Approaches for the Traveling Salesman Problem with Drone. Networks, Vol. 72, 2018, pp. 528–542, doi: 10.1002/net.21864.

[37] Roberti, R.—Ruthmair, M.: Exact Methods for the Traveling Salesman Problem with Drone. Transportation Science, Vol. 55, 2021, No. 2, pp. 315–335, doi: 10.1287/trsc.2020.1017.

Ha-Bang Ban received his Ph.D. in computer science at the Hanoi University of Science and Technology (HUST), Vietnam in 2015. He is currently the 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 peer-reviewed international journals and conferences.

Hai-Dang Pham received his engineering diploma in information technology from the Hanoi University of Science and Technology (HUST), Vietnam, in 1995 and his Ph.D. in computer science from École Pratique des Hautes Études (EPHE), France, in 2011. He is currently the Senior Lecturer at the School of Information and Communication Technology (SoICT), HUST, Vietnam. His current research interests include algorithms, parallel and distributed simulation, multi-agent based simulation and high performance computing.