A Multi-start VNS Algorithm for the TSP-D with Energy Constraints

Giovanni Campuzano(✉), Eduardo Lalla-Ruiz, and Martijn Mes

University of Twente, Drienerlolaan 5, 7500 AE Enschede, The Netherlands
{g.f.campuzanoarroyo,e.a.lalla,m.r.k.mes}@utwente.nl

Abstract. The Traveling Salesman Problem (TSP) is a well-known optimization problem with a wide range of extensions and applications in delivery systems. In this paper, we consider a recent extension of the TSP where a truck in collaboration with a single drone should visit a set of customers while minimizing the transportation times. We propose a Variable Neighbourhood Search (VNS) and a Multi-Start VNS (MS-VNS) algorithm, develop new neighbourhood structures, and compare the solutions against an existing mixed-integer linear programming (MILP) formulation. We take a set of instances based on existing benchmarks from the related literature. Results point out that the new neighbourhood structures substantially improve the performance of the VNS algorithms. Furthermore, results also show that the exact method is only able to find competitive solutions for small sets of instances, whereas our MS-VNS approach reaches better solution quality for large instances.

Keywords: Traveling salesman problem · Drones · UAV · Last-mile delivery · Multi start · Variable neighbourhood search

1 Introduction

Logistics plays an important role in today's economy. It controls the forward and reverse flows of goods from producers to consumers. In this context, the logistic sector is currently facing an era of unprecedented change with developments in digitization, autonomous vehicles, urbanization, increasing customers demands, and the rise of e-commerce. These changes have led companies to search for more efficient and sustainable management of urban freight distribution, in order to improve their service's quality, diminish greenhouse gas emissions, and reduce operational costs. Specifically in last-mile delivery, companies are seeing many opportunities to improve their business by incorporating autonomous vehicles into their logistic operations, giving rise to a wide range of new optimization problems in parcel distribution.

In this context, the Traveling Salesman Problem with Drone (TSP-D) is a recent extension of the TSP, in which a truck should work in collaboration with a single drone to serve a predefined group of customers while minimizing the transportation times or makespan. The TSP-D has shown distinct advantages

M. Mes et al. (Eds.): ICCL 2021, LNCS 13004, pp. 393–409, 2021.
https://doi.org/10.1007/978-3-030-87672-2_26

in last-mile operations, due to the drone's ability to transport parcels, food, medicine, and several other goods [8]. These autonomous vehicles have gained the attention of international companies, being involved in promising projects such as Zookal's project, Wing's project, and Parcelcopter, among others [20]. Additionally, drones have also shown strong applicability in other fields, such as energy, agriculture, forestry, environmental protection, and emergency [4].

The research community has also paid close attention to truck-and-drone problems. For example, [20] introduce the Flying Sidekick Traveling Salesman Problem (FSTSP) and the Parallel Drone Scheduling TSP (PDSTSP). They develop mathematical formulations and a two-phased heuristic approach to solve these problems. They show that the mathematical formulations are able to find efficient solutions for small-size instances, while the heuristic provides better solutions for large-size instances. [1] introduce the TSP-D and face this problem by developing a mathematical formulation and a heuristic approach. They also provide insights into the savings from the incorporation of a drone into the delivery system, showing the advantages of the truck-and-drone approach increase with faster drone speeds. [25] develop a compact formulation and several extensions for the TSP-D. They solve these problems by implementing a Branch-and-Price scheme, which is able to find solutions for instances of up to 39 customers. Branch-and-Cut procedures have been proposed in [6,9,26], and [2]. A Bender's Decomposition algorithm is developed in [28], reaching solutions for instances of up to 20 customers. In [21], authors study an extension of the FSTSP called the Multiple Flying Sidekicks Traveling Salesman Problem (mFSTSP). They develop a mathematical formulation and a three-phased heuristic, demonstrating that by incorporating more drones into the delivery system, it is possible to reduce the makespan for instances involving a large number of customers. The same authors study another extension, namely the mFSTSP with Variable Drone Speeds (FSTSP-VDS; [24]) in which the drone is able to fly at slower speed levels. They solve this problem by adapting the heuristic from [21]. They demonstrate that by allowing the drone to fly at slower speed levels, it is able to visit more customers due to a reduction in energy consumption. To provide a realistic representation of the delivery system, most of the works on truck-and-drone problems limit the drone flying range by using an energy consumption function, a maximum flying time, or a maximum flying distance ([6,11,12,16,19,27]). The reader is referred to [29], for a comparison of several energy consumption models used in truck-and-drone related problems.

In this paper, we propose two meta-heuristic approaches: a Variable Neighbourhood Search (VNS) and a Multi-Start Variable Neighbourhood Search (MS-VNS), which use 11 neighbourhood structures to explore the solution space. Those structures are a combination of neighbourhoods from the literature with some of our own. We take existing benchmarks from the literature to conduct the experiments and compare the results with the MILP formulation proposed in [25]. Furthermore, we provide insights into the heuristic performance improvements when incorporating the two new neighbourhood structures. Accordingly, we adapt to our algorithms the neighbourhood structures of the hybrid general VNS algorithm proposed in [13] and analyse the results. In addition, we

study these algorithms for the case where the drone flying range is infinite and when the drone has a limited energy endurance. Consequently, we show that the effectiveness of the MS-VNS is not only suitable for a theoretical version of the TSP-D, but also for a more realistic representation of the problem.

The remainder of this paper is organized as follows. Section 2 provides a detailed description of the TSP-D. In Sect. 3, the VNS and MS-VNS heuristics are presented. The experiments are conducted and analyzed in Sect. 4. Finally, Sect. 5 provides the main conclusions and suggests several directions for further research.

2 The Traveling Salesman Problem with Drone

The TSP-D is a problem that has gained increasing attention in last-mile delivery, due to the advantages that the incorporation of a drone provides. In this sense, the drone flies faster than the truck is able to drive and it avoids traffic congestion, however, its flying range is particularly restricted due to the battery endurance. Especially in urban areas, last-mile delivery faces crowded places with a high customers density, stimulating the collaboration between a truck and a drone. Figure 1 provides an illustrative example of a delivery schedule carried out by a truck and a drone. Here, solid arcs represent the truck arcs, dashed arcs represent the drone arcs, and double-solid arcs represent those that are traversed by the truck with the drone on board. Hence, in the example, there are customers that are served only by the truck (e.g., $\{8, 1\}$), customers that are visited by the drone (e.g., $\{4, 6\}$), and customers that are served by the truck with the drone on board (e.g., $\{3, 5, 7, 2\}$). We refer to those customers (or nodes) as truck nodes, drone nodes, and combined nodes, respectively.

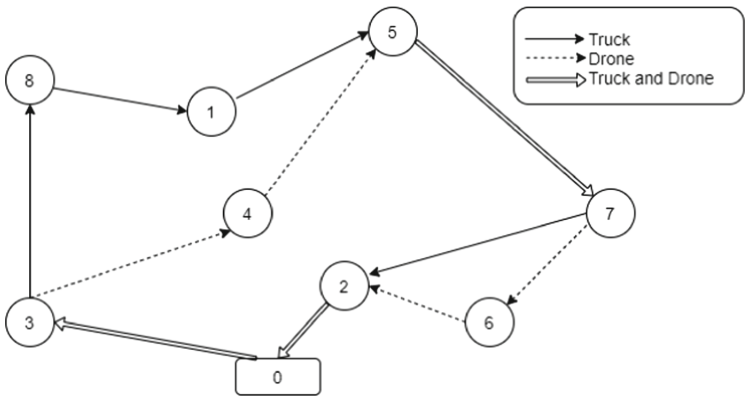

Fig. 1. Description of a feasible TSP-D route.

In real implementation scenarios, the TSP-D is a problem characterized by the restricted drone flying range. Therefore, when the drone is serving a given

customer, the energy consumed by the trip cannot exceed the energy capacity constraints. For this reason, it is important to properly coordinate the launching and landing times of the drone. Thus, we assume that every time the drone meets the truck in a combined node, the drone swaps the battery and loads a new package on board. Furthermore, to avoid feasibility issues, the vehicle that arrives first at a combined node should wait for the other vehicle before leaving to serve a new customer. To face the TSP-D described in [25], the following assumptions are made:

1. The truck capacity is large enough to carry all the packages and the drone batteries for the route.
2. The drone can only carry one package on board. For this reason, every time the drone is launched to serve a given customer, the drone should fly back after delivering the package, to meet the truck at a combined node.
3. The drone cannot take off and land in the same location. Therefore, once the drone is launched from a combined node, the truck cannot wait for the drone in the same place.
4. The drone has a fixed battery capacity and the drone's trips cannot exceed its battery's endurance. Furthermore, every time the drone meets the truck, it is assumed that the drone either is kept on the truck or swaps the battery and loads a new package on board.

3 Metaheuristic Algorithms

This section introduces two metaheuristic approaches to solve the TSP-D as presented in [25]. We first present two new neighbourhood structures and develop a Variable Neighbourhood Search algorithm in Sect. 3.1. Then, we extend the VNS heuristic to propose a Multi-Start Variable Neighbourhood Search algorithm in Sect. 3.2.

3.1 Variable Neighborhood Search Algorithm

VNS is a metaheuristic approach that systematically changes neighbourhood structures within a local search procedure to escape from local optima [15]. The first VNS algorithm was proposed in [14] and, ever since, a wide body of literature on VNS approaches has shown successful implementations in several routing problems, e.g., *the Asymmetric Traveling Salesman Problem* [3], *the Vehicle Routing Problem* [5], *the VRP with Multi-Depot* [17], *the VRP with Time Windows* [7], *the Flying Sidekick Traveling Salesman Problem* [11], and so forth. In this section, we present a VNS approach to solve the TSP-D, which uses eleven neighbourhoods to explore the solution space. We combine several neighbourhood structures from the literature with some of our own. The neighbourhoods *Make flying, Push left and Push right* are proposed in [1]. The neighbourhoods *Two optimality, Exchange 1.1, Exchange 2.1, Exchange 2.2, Reinsertion and Relocate customer* are proposed in [11]. The structures *Exchange 3.1 and Exchange 3.2* are developed in this research and are defined as follows:

- *Exchange 3.1*: This neighbourhood exchanges one node that is visited by the drone with another node that is visited by the truck.
- *Exchange 3.2*: This operator swaps two nodes that are visited by the drone.

A description for the VNS scheme developed in this research is provided in Algorithm 1. The input data of Algorithm 1 are the number of *iterations*, the number of shaking applications (*stopping*), the drone time matrix t^D, and the truck time matrix t^T. The output is the best TSP-D solution found in the search space S^* (incumbent). The objective function value of each solution is given by $f(\cdot)$, S_i stores the solution found when exploring the search space, and $S_i^{'}$ is used as a transition variable that stores the solution from the previous iteration. The index i represents the current iteration, l identifies the neighbourhood $\mathcal{N}_l(\cdot)$, p provides the number of shaking procedures applied during the search, and l_m gives the number of neighbourhoods.

Algorithm 1 consists of an *initialization phase* in which an initial TSP-D solution is built, and an *iterative phase* in which the solution space of the TSP-D is explored. An initial TSP solution is built in $SolveTSP(\cdot)$ by using the truck's time matrix t^T and applying a nearest neighbour search that starts from the depot (line 2). The TSP route is stored in S. Next, $MakeFlying(\cdot)$ builds a TSP-D solution by using the drone time matrix t^D and TSP route (line 3). This operation selects three consecutive nodes and builds a TSP-D solution in such a way that the truck visits the first and last node of the operation and the remaining node is visited by the drone. Therefore, given three selected nodes, the $MakeFlying$ movement creates a TSP-D operation leaving as a drone node the one that minimizes the costs the most. On the other hand, if the TSP-D operation does not reduce the transportation costs, the three selected nodes remain as a TSP section of the route within the TSP-D solution, i.e., visited by the truck and drone at the same time.

The iterative phase begins once an initial TSP-D solution has been constructed (line 6). This phase is repeated until either i or p reaches the maximum number of iterations or the maximum number of shaking procedures, respectively (line 7). The set of neighbourhoods is explored within the second while-loop until the current solution S_i is improved or all the neighbourhoods in $\mathcal{N}_l$ are checked (lines 9–11). To explore the solution space, the VNS algorithm applies the set of neighbourhoods $\mathcal{N}_l$ starting from the structure with the smallest search space to the neighbourhood with the largest one. Further, a given neighbourhood $\mathcal{N}_l$ is explored until its best solution is found. Once the algorithm finishes exploring the set of neighbourhoods, if S_i is better than $S_i^{'}$ and in addition the current solution is better than the incumbent ($S_i < S^*$), the incumbent is updated and p is set to zero (lines 12–15). Additionally, every time S_i is better than the previous solution $S_i^{'}$, l is set to 0 (lines 16). Otherwise, if the current solution S_i is worse than the previous solution $S_i^{'}$, it means the algorithm has found a local optimum, so the shaking procedure is applied and the solution is stored in S_i (lines 17–19). Finally, S_{i+1} and $S_{i+1}^{'}$ store the solution from the current iteration S_i (line 20).

Algorithm 1: Variable Neighbourhood Search

```
Data: (iterations, stopping, t^D, t^T)
Result: (S^*)
Initialization phase:
S ← SolveTSP(t^T)
S_i, S_i', S^* ← MakeFlying(S, t^D)
f(S^*) ← +∞
i, p ← 0
Iterative phase:
while i < iterations || p < stopping do
    l ← 0
    while f(S_i) ≥ f(S_i') and l ≤ l_m do
        S_i ← N_l(S_i)
        l ← l + 1
    if f(S_i) < f(S_i') then
        if f(S_i) < f(S^*) then
            S^* ← S_i
            p ← 0
        l ← 0
    else
        S_i ← Shaking(S_i)
        p ← p + 1
        if p then
            =
        =
    S_{i+1}', S_{i+1} ← S_i
    i ← i + 1
```

In a TSP-D solution, every time the drone visits a customer, the part of the route between the combined take-off and landing nodes is referred to as an *Operation*. In this context, the shaking procedure consists of firstly taking apart two operations to let the truck visit the customers and then applying three nested neighbourhoods, which are selected randomly. When applying a given random neighbourhood, the solution selected from this neighbourhood is chosen randomly without considering the objective function value.

3.2 Multi-Start Variable Neighborhood Search Algorithm

The VNS approaches have been broadly accepted by the research community and, over the years, several VNS schemes with additional mechanisms have been developed to improve their performances, for instance, *Descent Search*, *Reduced VNS*, *Basic VNS*, *General VNS*, *Skewed VNS*, *Decomposition Search*, *Parallel*, and *Primal-Dual*, among others [15]. In addition, multi-start mechanisms

are applied to add diversification to the heuristic optimization process by re-starting the search once a certain criterion is met [23]. Consequently, with the purpose of exploring deeply promising areas of the search space, we incorporate a multi-start mechanism into the VNS heuristic of Sect. 3.1. As such, we develop an effective multi-start VNS which, after a certain number of iterations without improving the solution, re-starts the search from those solutions that were chosen as *incumbents* throughout the optimization process. As a result, we re-start the exploration from previous local optima, which are adjusted by a *Shaking* procedure, to explore new regions of promising areas already identified. A description of the MS-VNS heuristic is provided in Algorithm 2. The input data of Algorithm 2 is the same as Algorithm 1 plus the number of consecutive shakings until a re-starting procedure is applied (*Restarting*). The output is the best TSP-D solution found in the search space S^*. The counter i represents the iteration number, p shows the number of re-starting procedures applied, and h displays the number of shakings.

As mentioned, Algorithm 2 is an adaptation of Algorithm 1, which incorporates a multi-start method. Note that different from the VNS, the MS-VNS randomizes the order in which the neighbourhoods $\mathcal{N}_l$ are explored in every iteration i (line 9). Therefore, in order to re-start the search, the MS-VNS uses *list()*, which is a list that stores each new incumbent found in the iterative phase (line 16). Every time the incumbent is updated, the number of shaking procedures h is set at zero (line 17). On the other hand, if after exploring the set of neighbourhoods $\mathcal{N}_l$ the solution from the previous iteration is not improved, one shaking procedure is applied and the counter h is increased by one (lines 19–21). Additionally, when h reaches the number of consecutive iterations without improving S_i (*Restarting*), the re-starting mechanism is activated (line 22). Consequently, S_i randomly takes one of the solutions stored in $list()$ and a shaking procedure is applied to this new solution in S_i (lines 23–24). Then, the counter of shakings h is set at zero and the counter of re-starting procedures p is increased by one (lines 25–26). Furthermore, it is worth mentioning that the MS-VNS heuristic also adjusts the *Shaking* procedure. The shaking procedure consists of firstly taking apart seven operations, if possible, and then three nested neighbourhoods, which are selected randomly, are applied. After that, the shaking uses the $MakeFlying()$ neighbourhood structure to create a new operation. Finally, similarly as done in the VNS heuristic, the solutions selected from the neighbourhoods in the shaking procedure are randomly chosen.

4 Numerical Experiments

In this section, we provide the computational experiments on the TSP-D. A detailed description of the computational settings, instances, heuristic tuning, and MILP model parameters are presented in Sect. 4.1. Computational results are presented in Sect. 4.2.

Algorithm 2: Multi-Start Variable Neighbourhood Search

Data: ($iterations$, $stopping$, t^D, t^T)
Result: (S^*)
Initialization phase:
$S \leftarrow SolveTSP(t^T)$
$S_i, S_i^{'}, S^* \leftarrow MakeFlying(S, t^D)$
$f(S^*) \leftarrow +\infty$
$i, p \leftarrow 0$
Iterative phase:
while $i < iterations \mid\mid p < stopping$ **do**
  $l \leftarrow 0$
  $\mathcal{N}_l() \leftarrow randomize()$
  while $f(S_i) \geq f(S_i^{'})$ *and* $l \leq l_m$ **do**
    $S_i \leftarrow \mathcal{N}_l(S_i)$
    $l \leftarrow l + 1$
  if $f(S_i) < f(S_i^{'})$ **then**
    if $f(S_i) < f(S^*)$ **then**
      $S^* \leftarrow S_i$
      $list(S^*)$
      $h \leftarrow 0$
    $l \leftarrow 0$
  else
    $S_i \leftarrow Shaking(S_i)$
    $h \leftarrow h + 1$
    if $h == Restarting$ **then**
      $S_i \leftarrow randomize(list())$
      $S_i \leftarrow Shaking(S_i)$
      $h = 0$
      $p \leftarrow p + 1$
  $S_{i+1}^{'}, S_{i+1} \leftarrow S_i$
  $i \leftarrow i + 1$

4.1 Experimental Settings

This section describes the parameters used by the heuristics from Sect. 3 as well as the computational settings. To conduct the experiments, we take the *single center* set of benchmark instances from [1], and execute the test on a computer equipped with a 2.29 GHz Intel(R) Xeon(R) Gold 5218 CPU with 64 GB of RAM. The algorithm was coded in C++ and solved using CPLEX 12.10. The truck and drone speeds considered are 7 and 15 [m/s], respectively.

Given two locations i and j, and their corresponding $x-y$ coordinates (x_i, y_i), the time it takes the drone to go from i to j is computed as $t_{ij}^D = t_i^{tk} + t_{ij}^h + t_j^l$, where t_{ij}^h is the horizontal flying time, t_i^{tk} the takeoff time, and t_{ij}^l the landing time. t_i^{tk} and t_i^l are constant parameter of values 60 an 30 [s], respectively. The

horizontal flying time is computed as $t^h_{ij} = \frac{\sqrt{(x_j-x_i)^2+(y_j-y_i)^2}}{v^D_{ij}} \forall (i,j) \in A$ (i.e., considering Euclidean distance). Likewise, when the truck is traversing the arc $(i,j) \in A$, at a given speed v^T_{ij}, the time it takes the truck to go from i to j is computed as $t^T_{ij} = \frac{\lfloor |x_j-x_i|+|y_j-y_i| \rfloor}{v^T_{ij}} \forall (i,j) \in A$ (i.e., considering Manhattan distance). We expand the $x-y$ coordinates by 100 to compute the transportation times.

We base the calculations of the energy consumption of the drone on the consumption model of [18], which depends on the selected drone speed, taking off speed, landing speed, and payload. The consumption model is described by Eqs. (1)–(3). Further, [18] propose an energy model $f(\cdot) = p^{tl} + p^c + p^h$, where p^{tl} is the induced power of the vertical flow (either take off or landing), p^c represents the profile power consumption during the horizontal cruise, and p^h is the energy consumed during hover. In this regard, when the drone is traversing the arc $(i,j) \in A$, transporting a load equal to weight w_j of customer $j \in N$, at a vertical drone speed (v_{ve}), and at a selected drone speed level v^D_{ij}, the time t^D_{ij} to traverse the arc (i,j) is given by: $t^D_{ij} = t^{tk}_i + t^h_{ij} + t^l_j$. Consequently, by making use of these different flying times of the drone, the battery consumption is set by Eqs. (1)–(2) as: $b_{ij}(w_j, v^D_{ij}) = t^{tk}_i \cdot p^{tl}_i(w_j, v_{ve}) + t^h_{ij} \cdot p^c_{ij}(w_j, v^D_{ij}) + t^l_j \cdot p^{tl}_j(w_j, v_{ve})$. Note, when traveling to meet the truck without a parcel on board, the same Eqs. (1)–(3) are used to compute b_{ij} with the payload w_j equal to 0.

$$p^{tl}_{ij} = k_1(W+w_j)g\left[\frac{v_{ve}}{2}\sqrt{\left(\frac{v_{ve}}{2}\right)^2 + \frac{(W+w_j)g}{k_2^2}}\right] + c_2\left((W+w_j)g\right)^{3/2} \qquad (i,j) \in A \quad (1)$$

$$p^c_{ij} = (c_1+c_2)\left(\left((W+w_j)g - c_5(v^D_{ij}\cos\theta)^2\right)^2 + \left(c_4(v^D_{ij})^2\right)^2\right)^{3/4} + c_4(v^D_{ij})^3 \qquad (i,j) \in A \quad (2)$$

$$p^h_{ij} = (c_1+c_2)\left((W+w_j)g\right)^{3/2} \qquad (i,j) \in A \quad (3)$$

In Eqs. (1)–(3), c_1, c_2, c_4 c_5, k_1, k_2, W, θ, g, and v_{ve} are model coefficients, whose values are derived from [18] and provided in Table 1. This way, c_1, c_2, c_4 c_5, k_1 and k_2 are model constant, W represents the drone frame weight, θ is the angle of attack, i.e., the vertical angle with which the drone faces the wind, and g is the gravitational constant. Note that the model of [18] assumes a fixed angle of attack for a certain range of payloads. Depending on the type of drone considered, other models could be applicable here.

Table 1. Coefficient values for the energy consumption models presented in [24] and [18].

Coefficient:	k_1	k_2	c_1	c_2	c_4	c_5	W	g	θ	$V_{ve}(takeoff)$	$V_{ve}(landing)$
Value:	0.8554	0.3051	2.8037	0.3177	0.0296	0.0279	1.5	9.8	10	5	10
Units:	$[unitless]$	$\sqrt{kg/m}$	$\sqrt{m/kg}$	$\sqrt{m/kg}$	kg/m	Ns/m	kg	m/s^2	Degrees	m/s	m/s

To properly select the parameter values of the algorithms presented in Sect. 3, the Friedman non-parametric statistical test is applied over the performance of

the heuristic approaches [22]. Table 2 shows the parameters assessed in the Friedman's test to state the most suitable settings, where *iterations* represents the number of iterations, *stopping* establish the maximum number of shaking procedures allowed for the VNS and the maximum number of re-starting procedures allowed for the MS-VNS, and *Restarting* is the maximum number of shaking procedures before the re-starting mechanism is activated for the MS-VNS. In this regard, the multiple parameters Friedman's test is performed in those cases in which the null hypothesis is rejected. A total of 5 representative instances for the combination of the parameters from Table 2 are solved by the VNS and MS-VNS algorithms. A significance level of $\alpha_{friedman} = 0.05$ is used for the objective functions to indicate a significant difference among the parameter assessed. After applying the Friedman non-parametric statistical test, we decided to use the parameter settings $Iterations = 9000$ and $Stopping = 70$ for the VNS and $Iterations = 21000$, $Stopping = 200$, and $Restarting = 90$ for the MS-VNS.

Table 2. Parameter values used to configure heuristics VNS and MS-VNS.

Parameter	VNS	MS-VNS
Iterations	{11000, 9000, 7000}	{42000, 21000, 7000}
Stopping	{90, 70, 50}	{4000, 300, 200}
Restarting	–	{150, 90, 30}

4.2 Computational Results

In this section, we conduct three sets of experiments with the purpose of measuring the effectiveness of our algorithms. The first set of experiments provides a comparison of the heuristic approaches with infinite energy capacity in the drone's battery. More precisely, we compare our VNS and MS-VNS algorithms using both the original 9 neighbourhoods (as used in the hybrid general VNS proposed in [10]) and the 11 neighbourhoods that include the 2 neighbourhoods proposed in this work. We denote the neighbourhood structures using 9 or 11 as subscript. The second set of experiments presents the results of the exact formulation and the metaheuristic from the first set of experiments with the best performance, for the TSP-D with infinite energy capacity. The third set shows the results of the TSP-D where the drone flying range is restricted by a limited energy capacity, using the same methods as used in the second set of experiments.

The tables presented below report the results of the optimization model and the metaheuristics VNS and MS-VNS, for instances of up to 75 nodes. The tables show the best solution found of 10 runs for each heuristic. The experiments are carried out with a time limit of 3600 (s) for the MILP model. The columns show the objective function value (Z), initial constructive TSP-D solution that the metaheuristic algorithms use to explore the solution space (Z_{init}), Gap (%), average computation time in seconds of the experiments for each instance, a

percentage difference of the heuristic objective function Z compared to the MILP formulation as: $\Delta Z(\%) = 100 \cdot \frac{Z_{MILP}-Z}{Z_{MILP}}$, and a percentage difference of the heuristic objective function compared to the initial constructive TSP-D solution as: $\Delta Z_{TSP-D}(\%) = 100 \cdot \frac{Z_{TSP-D}-Z}{Z_{TSP-D}}$.

Assessment of the Neighbourhood Structures for VNS Algorithms. Table 3 reports the results of the two new neighbourhood structures when studying the performance of the VNS and MS-VNS algorithms for the TSP-D with unlimited battery capacity. Best values are bold-faced. In addition, Fig. 2 provides a summary of the results, where for every heuristic algorithm the set of instances is averaged by size n.

Table 3. Comparison of the VNS and MS-VNS algorithms for the TSP-D.

Instance		VNS_9		VNS_{11}		MS-VNS_9		MS-VNS_{11}	
Name	n	Z	Time (s)	Z	Time (s)	Z	Time (s)	Z	Time (s)
51	10	4695	0.02	4118	0.03	3793	1.95	**3625**	2.12
52		6480	0.03	**5280**	0.02	**5280**	2.02	**5280**	1.94
53		4388	0.04	5629	0.03	8022	3.23	**3608**	1.73
54		6485	0.03	3883	0.03	4759	1.74	**3833**	1.54
55		6429	0.03	5023	0.02	4897	2.23	**4787**	2.09
61	20	7586	0.07	6345	0.12	6232	8.16	**4884**	7.57
62		7840	0.12	5087	0.07	5859	7.20	**4500**	6.98
63		11832	0.06	7603	0.08	6072	6.99	**5818**	7.09
64		7079	0.05	5112	0.07	5859	6.87	**4343**	6.24
65		7767	0.07	5592	0.06	4715	6.35	**4707**	6.41
71	50	9828	0.86	8344	1.66	7660	83.50	**7513**	88.25
72		12810	1.40	9799	1.12	8569	76.67	**8402**	83.21
73		8777	1.20	7271	1.87	6566	82.81	**6521**	87.66
74		13794	0.68	10502	1.15	9699	79.04	**9183**	83.41
75		14331	1.38	10430	1.43	12263	78.90	**9349**	82.95
81	75	19308	4.06	14226	12.32	14318	315.94	**12704**	323.11
82		16282	5.67	11776	10.87	10660	310.28	**10134**	337.81
83		16449	5.95	13050	7.20	12878	334.56	**11646**	335.94
84		18089	3.67	14068	6.89	12666	309.05	**12434**	333.00
85		15070	3.92	12413	12.05	12022	325.37	**10682**	336.90
Avg.		10765.95	1.47	8277.55	2.85	8139.45	102.14	7197.65	106.80

Results highlight the effectiveness of the multi-start mechanism in the performance of heuristic MS-VNS_{11}, although obviously at the expense of increasing computational time. As such, we can see that MS-VNS_{11} is the algorithm that finds the best solution for all the instances studied, where only for instance 52 the heuristics VNS_{11} and MS-VNS_9 were able to find the same result as MS-VNS_{11}.

Moreover, results also point out the effect of the two new neighbourhoods, which help the heuristics algorithms VNS and MS-VNS perform a better exploration of the solution space.

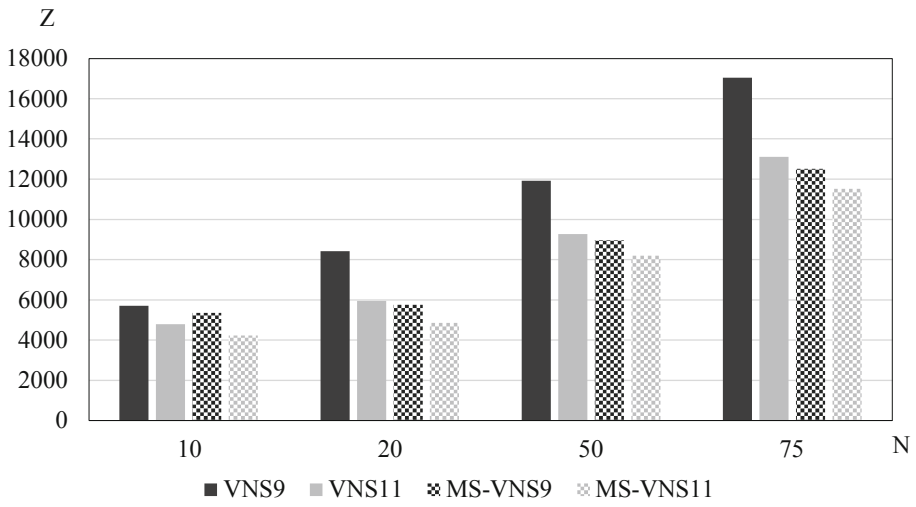

Fig. 2. Comparison of VNS_9, VNS_{11}, MS-VNS_9, and MS-VNS_{11} for unlimited drone flying range.

Assessment of Optimization Approaches for Unlimited Drone Flying Range. Table 4 reports the results of the MILP formulation and the MS-VNS_{11} algorithm for the TSP-D when considering unlimited battery capacity. Best values are bold-faced.

Results point out that the TSP-D formulation is able to find good quality solutions for the small instance set (10–20 nodes) in computational times under 50 s. The formulation reaches optimality for all the instances of 10 nodes and gaps under 17% for the instances of 20 nodes. With respect to the MS-VNS_{11} algorithm, the MS-VNS_{11} is able to reach optimal solutions for all the instances of 10 nodes and improve the performance of the MILP formulation for instances 61 and 64 by 2.3% and 0.84%, respectively. Further, the MILP formulation present better solutions than the MS-VNS_{11} for instances 62, 63, and 65 by 1.99%, 0.14%, and 0.84%, respectively. Additionally, the MS-VNS_{11} improves the initial constructive TSP-D solution that the algorithm uses to start the iterative search by at least 21%.

When conducting the experiments on the large set of instances, we see that the optimization model finds feasible solutions with gaps over 74% for all the instances. Nevertheless, MS-VNS_{11} presents better results by at least 64% for all the instances of the large set and improves the constructive initial solution by at least 35%. Furthermore, the initial constructive TSP-D solution shows better solutions than the final value of the MILP model for all the instances. This

Table 4. Comparison of the MILP formulation and the MS-VNS$_{11}$ algorithm for the TSP-D with unrestricted drone flying range.

Instance		MILP			Constructive	MS-VNS$_{11}$			
Name	n	Z	Gap (%)	Time (s)	Z_{TSP-D}	Z	Time (s)	ΔZ_{MILP} (%)	ΔZ_{TSP-D} (%)
51	10	**3625**	0.00	25.45	8653	**3625**	2.12	0.00	58.11
52		**5280**	0.00	17.80	6754	**5280**	1.94	0.00	21.82
53		**3608**	0.00	20.38	5901	**3608**	1.73	0.00	38.86
54		**3833**	0.00	18.19	6988	**3833**	1.54	0.00	45.15
55		**4787**	0.00	49.00	6913	**4787**	2.09	0.00	30.75
61	20	4999	17.12	3620.92	8504	**4884**	7.57	2.30	42.57
62		**4412**	3.38	3600.59	9603	4500	6.98	−1.99	53.14
63		**5810**	8.58	3632.08	9479	5818	7.09	−0.14	38.62
64		4380	9.34	3631.00	7819	**4343**	6.24	0.84	44.46
65		**4668**	4.20	3616.73	8913	4707	6.41	−0.84	47.19
71	50	33597	84.24	3613.98	12370	**7513**	88.25	77.64	39.26
72		57116	89.64	3656.48	17595	**8402**	83.21	85.29	52.25
73		18608	74.83	3670.56	12994	**6521**	87.66	64.96	49.82
74		52287	88.18	3615.36	14444	**9183**	83.41	82.44	36.42
75		45991	85.72	3618.97	20191	**9349**	82.95	79.67	53.70
81	75	87452	90.12	3601.52	22590	**12704**	323.11	85.47	43.76
82		76357	90.63	3600.64	17403	**10134**	337.81	86.73	41.77
83		75689	90.09	3601.08	20393	**11646**	335.94	84.61	42.89
84		84827	90.74	3600.59	19173	**12434**	333.00	85.34	35.15
85		70425	89.79	3602.06	16787	**10682**	336.90	84.83	36.37
Avg.		32387.55	45.83	2720.67	12673.35	7197.65	106.80	40.86	42.60

demonstrates that, for the large set, the initialization phase of the algorithm generates solutions that are more effective than the optimization model. According to this, we conclude that the MS-VNS$_{11}$ presents competitive behaviour for the small and large set of instances, where for the majority of the instances the VM-VNS$_{11}$ reaches the same or better solutions than the MILP formulation in smaller computational times.

Assessment of the Optimization Approaches for Limited Drone Flying Range. Table 5 reports the results of the MILP formulation and the MS-VNS$_{11}$, considering a battery capacity of 600 [KJ]. Best values are bold-faced.

When studying the TSP-D with a restricted drone energy capacity, results show that the MILP model is able to reach optimality for all the instances of 10 nodes under 49 s, and present gaps under 17% for instances of 20 nodes. In this regard, when studying the heuristic, we see that the MS-VNS$_{11}$ is able to reach the optimal solution for all instances of the small set and it finds better solutions than the MILP model for instances 62 and 64 by 3.97% and 5.21%, respectively. The MILP formulation reaches better solutions than the MS-VNS$_{11}$ for instances 61 and 65 by 0.48% and 0.45%. In addition, the MS-VNS$_{11}$ improves the initial TSP-D solution by at least 21% for the small set.

Table 5. Comparison of the MILP formulation and the MS-VNS algorithm for the TSP-D with restricted drone flying range.

Instance		MILP			Constructive	MS-VNS$_{11}$			
Name	n	Z	Gap (%)	Time (s)	Z_{TSP-D}	Z	Time (s)	ΔZ_{MILP} (%)	ΔZ_{TSP-D} (%)
51	10	**3793**	0.00	24.16	8653	**3793**	1.95	0.00	56.17
52		**5280**	0.00	10.02	6754	**5280**	2.02	0.00	21.82
53		**8022**	0.00	10.19	8760	**8022**	3.23	0.00	8.42
54		**4759**	0.00	12.53	9028	**4759**	1.74	0.00	47.29
55		**4897**	0.00	5.52	7144	**4897**	2.23	0.00	31.45
61	20	**6202**	21.94	3600.83	10556	6232	8.16	−0.48	40.96
62		6101	14.21	3682.98	9027	**5859**	7.20	3.97	35.09
63		**6072**	8.05	3607.05	10921	**6072**	6.99	0.00	44.40
64		6181	27.86	3600.63	7819	**5859**	6.87	5.21	25.07
65		**4694**	5.03	3600.30	10529	4715	6.35	−0.45	55.22
71	50	46679	88.53	3615.20	12939	**7660**	83.50	83.59	40.80
72		54923	89.37	3607.88	17905	**8569**	76.67	84.40	52.14
73		27755	83.11	3648.28	12994	**6566**	82.81	76.34	49.47
74		40209	84.46	3609.17	19496	**9699**	79.04	75.88	50.25
75		64628	88.13	3606.97	20191	**12263**	78.90	81.03	39.27
81	75	85509	89.15	3600.78	24700	**14318**	315.94	83.26	42.03
82		73198	90.25	3601.31	17403	**10660**	310.28	85.44	38.75
83		285228	97.19	3601.47	22404	**12878**	334.56	95.49	42.52
84		89872	91.18	3600.95	19173	**12666**	309.05	85.91	33.94
85		56462	86.94	3601.67	22083	**12022**	325.37	78.71	45.56
Avg.		44023.20	48.27	2712.39	13923.95	8139.45	102.14	41.91	40.03

On the other hand, we see that the MILP model provides feasible solutions with gaps over 83% for all the instances of the large set. In this respect, the MS-VNS$_{11}$ algorithms report better solutions than the MILP model, improving the initial TSP-D solution by at least 33%. Besides, results show that the initial TSP-D solutions are better than the solutions provided by the optimization model. Consequently, this demonstrates that the initialization phase of the VNS scheme reaches better solutions than the MILP model for these instances when studying the TSP-D with a restricted drone flying range. Similarly as for the previous set of experiments, we conclude that for the small and large set of instances the MS-VNS$_{11}$ presents competitive solutions for the TSP-D with a restricted drone flying range and small computational times. This shows that the effectiveness of the multi-start mechanism does not only apply to the unrestricted drone flying range, but also to the case in which limited energy capacity is considered. Consequently, the multi-start mechanism allows the MS-VNS to effectively explore the reduced solution space given the energy constraints, which provides a more realistic representation of the last-mile delivery operations.

5 Conclusions and Future Work

In this paper, we studied the Traveling Salesman Problem with Drone (TSP-D), with the objective of minimizing the makespan. Because of the NP-Hardness of the TSP-D, we proposed two heuristic approaches, a VNS and a MS-VNS, and compared their performance with a formulation from the literature [25]. Our heuristic schemes consist of two phases, where up to 11 neighbourhoods can be explored in every iteration of the search. We proposed two new neighbourhood structures and studied their effectiveness by comparing with the neighbourhoods from [13]. We showed that the MS-VNS algorithm presents competitive solutions for a small and large set of instances when comparing the solutions with the MILP formulation, with smaller computational times. Moreover, we demonstrate that the two neighbourhood structures substantially improve the performance of the heuristic approaches, finding better solutions in all the considered instances. In addition, we also demonstrate that the multi-start approach improves the performance of the VNS algorithm, finding better solutions for all the considered instances, but at the expense of higher computational times. Consequently, we conclude that the effectiveness of the two new neighbourhoods and the multi-start approach holds for the theoretical version of the TSP-D as well as for the more realistic representation of the problem including energy constraints.

With regards to future research, more emphasis should be placed on developing more efficient algorithms to reach better results in smaller computational times. Hence, algorithms that include learning, such as look ahead, arrangement of neighbourhoods based on their performance, and selection of a subgroup of neighbourhoods in different stages of the optimization process, seem particularly interesting as an object of study.

References

1. Agatz, N., Bouman, P., Schmidt, M.: Optimization approaches for the traveling salesman problem with drone. Transp. Sci. **52**(4), 965–981 (2018)
2. Boccia, M., Masone, A., Sforza, A., Sterle, C.: A column-and-row generation approach for the flying sidekick travelling salesman problem. Transp. Res. Part C Emerg. Technol. **124**, 102913 (2021)
3. Burke, E.K., Cowling, P.I., Keuthen, R.: Effective local and guided variable neighbourhood search methods for the asymmetric travelling salesman problem. In: Boers, E.J.W. (ed.) EvoWorkshops 2001. LNCS, vol. 2037, pp. 203–212. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45365-2_21
4. Carlsson, J.G., Song, S.: Coordinated logistics with a truck and a drone. Manage. Sci. **64**(9), 4052–4069 (2018)
5. Chen, P., Huang, H., Dong, X.: Variable neighborhood search algorithm for fleet size and mixed vehicle routing problem. J. Syst. Simul. **23**(9), 1945–1950 (2011)
6. Dell'Amico, M., Montemanni, R., Novellani, S.: Drone-assisted deliveries: new formulations for the flying sidekick traveling salesman problem. Optim. Lett. **15**(5), 1617–1648 (2019). https://doi.org/10.1007/s11590-019-01492-z
7. Dhahri, A., Mjirda, A., Zidi, K., Ghedira, K.: A VNS-based heuristic for solving the vehicle routing problem with time windows and vehicle preventive maintenance constraints. Procedia Comput. Sci. **80**, 1212–1222 (2016)

8. Dorling, K., Heinrichs, J., Messier, G.G., Magierowski, S.: Vehicle routing problems for drone delivery. IEEE Trans. Syst. Man Cybern. Syst. **47**(1), 70–85 (2016)
9. El-Adle, A.M., Ghoniem, A., Haouari, M.: Parcel delivery by vehicle and drone. J. Oper. Res. Soc. 1–19 (2019)
10. de Freitas, J.C., Penna, P.H.V.: A randomized variable neighborhood descent heuristic to solve the flying sidekick traveling salesman problem. Electron. Notes Discrete Math. **66**, 95–102 (2018)
11. de Freitas, J.C., Penna, P.H.V.: A variable neighborhood search for flying sidekick traveling salesman problem. Int. Trans. Oper. Res. **27**(1), 267–290 (2020)
12. Gonzalez-R, P.L., Canca, D., Andrade-Pineda, J.L., Calle, M., Leon-Blanco, J.M.: Truck-drone team logistics: a heuristic approach to multi-drop route planning. Transp. Res. Part C Emerg. Technol. **114**, 657–680 (2020)
13. Ha, Q.M., Deville, Y., Pham, Q.D., Hà, M.H.: On the min-cost traveling salesman problem with drone. Transp. Res. Part C Emerg. Technol. **86**, 597–621 (2018)
14. Hansen, P., Mladenović, N.: An introduction to variable neighborhood search. In: Voß S., Martello, S., Osman, I.H., Roucairol, C. (eds.) Meta-heuristics, pp. 433–458. Springer, Boston (1999). https://doi.org/10.1007/978-1-4615-5775-3_30
15. Hansen, P., Mladenović, N., Pérez, J.A.M.: Variable neighbourhood search: methods and applications. Ann. Oper. Res. **175**(1), 367–407 (2010)
16. Jeong, H.Y., Song, B.D., Lee, S.: Truck-drone hybrid delivery routing: Payload-energy dependency and no-fly zones. Int. J. Prod. Econ. **214**, 220–233 (2019)
17. Kocatürk, F., Tütüncü, G.Y., Salhi, S.: The multi-depot heterogeneous VRP with backhauls: formulation and a hybrid VNS with GRAMPS meta-heuristic approach. Ann. Oper. Res. 1–26 (2021). https://doi.org/10.1007/s10479-021-04137-6
18. Liu, Z., Sengupta, R., Kurzhanskiy, A.: A power consumption model for multi-rotor small unmanned aircraft systems. In: 2017 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 310–315. IEEE (2017)
19. Marinelli, M., Caggiani, L., Ottomanelli, M., Dell'Orco, M.: En route truck-drone parcel delivery for optimal vehicle routing strategies. IET Intell. Transport Syst. **12**(4), 253–261 (2017)
20. Murray, C.C., Chu, A.G.: The flying sidekick traveling salesman problem: optimization of drone-assisted parcel delivery. Transp. Res. Part C Emerg. Technol. **54**, 86–109 (2015)
21. Murray, C.C., Raj, R.: The multiple flying sidekicks traveling salesman problem: parcel delivery with multiple drones. Transp. Res. Part C Emerg. Technol. **110**, 368–398 (2020)
22. Oda, T., Liu, Y., Sakamoto, S., Elmazi, D., Barolli, L., Xhafa, F.: Analysis of mesh router placement in wireless mesh networks using Friedman test considering different meta-heuristics. Int. J. Commun. Netw. Distrib. Syst. **15**(1), 84–106 (2015)
23. Qi, X., Fu, Z., Xiong, J., Zha, W.: Multi-start heuristic approaches for one-to-one pickup and delivery problems with shortest-path transport along real-life paths. PloS One **15**(2), e0227702 (2020)
24. Raj, R., Murray, C.: The multiple flying sidekicks traveling salesman problem with variable drone speeds. Transp. Res. Part C Emerg. Technol. **120**, 102813 (2020)
25. Roberti, R., Ruthmair, M.: Exact methods for the traveling salesman problem with drone. Transp. Sci. **55**(2), 315–335 (2021)
26. Schermer, D., Moeini, M., Wendt, O.: A branch-and-cut approach and alternative formulations for the traveling salesman problem with drone. Networks **76**(2), 164–186 (2020)

27. Tu, P.A., Dat, N.T., Dung, P.Q.: Traveling salesman problem with multiple drones. In: Proceedings of the Ninth International Symposium on Information and Communication Technology, pp. 46–53 (2018)
28. Vásquez, S.A., Angulo, G., Klapp, M.A.: An exact solution method for the TSP with drone based on decomposition. Comput. Oper. Res. **127**, 105127 (2020)
29. Zhang, J., Campbell, J.F., Sweeney, D.C., II., Hupman, A.C.: Energy consumption models for delivery drones: a comparison and assessment. Transp. Res. Part D Transp. Environ. **90**, 102668 (2021)