drones

MDPI

Article

UAVTrajectory Optimization for Data Pickupand Delivery with Time Windows

Ines Khoufi [1], Anis Laouiti [1,*], Cedric Adjih [2] and Mohamed Hadded [3]

[1] SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris, 9 rue Charles Fourier, 91011 Evry, France; ines.khoufi@telecom-sudparis.eu
[2] Inria, Saclay Ile-de-France Research Centre, 91120 Palaiseau, France; cedric.adjih@inria.fr
[3] REVECOM Team, Institute VEDECOM, 23 bis allee des Marronniers, 78000 Versailles, France; mohamed.elhadad@vedecom.fr
* Correspondence: anis.laouiti@telecom-sudparis.eu

Abstract: Unmanned Aerial Vehicles (UAVs), also known as drones, are a class of aircraft without the presence of pilots on board. UAVs have the ability to reduce the time and cost of deliveries and to respond to emergency situations. Currently, UAVs are extensively used for data delivery and/or collection to/from dangerous or inaccessible sites. However, trajectory planning is one of the major UAV issues that needs to be solved. To address this question, we focus in this paper on determining the optimized routes to be followed by the drones for data pickup and delivery with a time window with an intermittent connectivity network, while also having the possibility to recharge the drones' batteries on the way to their destinations. To do so, we formulated the problem as a multi-objective optimization problem, and we showed how to use the Non-dominated Sorting Genetic Algorithm II (NSGA-II) to solve this problem. Several experiments were conducted to validate the proposed algorithm by considering different scenarios.

Keywords: UAVs; pickup and delivery; time window; multi-objective optimization; intermittent connectivity; MOEA; NSGA-II

Citation: Khoufi, I.; Laouiti, A.; Adjih, C.; Hadded, M. UAVTrajectory Optimization for Data Pickupand Delivery with Time Windows. *Drones* **2021**, *5*, 27. https://doi.org/10.3390/drones5020027

Academic Editor: Diego González-Aguilera

Received: 25 February 2021
Accepted: 13 April 2021
Published: 16 April 2021

1. Introduction, Context, and Motivation

In recent years, Unmanned Aerial Vehicles (UAVs) have been proposed for various applications, including inventive delivery methods. In this article, we focus on the specific problem of data delivery with UAVs when radio communication is used and the connectivity is intermittent.

In such networks, UAVs act as data mules in an environment where the communication range is limited, and thus, connectivity is intermittent. UAVs are in charge of collecting data (e.g., messages, videos, images, medical messages, high-resolution photographs [1]) from a source site and carrying them to a destination site. The use of UAVs is ideally adapted to such scenarios as they provide considerable time savings compared to other types of transport. Nevertheless, these flying vehicles have to operate with limited energy due to their modest size and constrained weight, and they also have limited storage capacity. In addition, for the applications under consideration, data exchanges are subject to time constraints: the UAVs have a time limit for delivering data to destinations. All these constraints have to be taken into account when computing optimized trajectories for UAVs that act as data mules in a collaborative fashion.

The literature is rich with several methods designed to optimize the itineraries of mobile vehicles for predefined delivery missions operating under a set of constraints. Among them, the Pickup and Delivery optimization Problem (PDP) [2] models the scenario in which a fleet of vehicles must collaboratively achieve a set of transportation tasks. In this model, customers advertise their transportation needs by sending their requests to a headquarter specifying the pickup and/or delivery locations. The aim is then to

find a routing solution where vehicles service all requests, satisfying the time windows and vehicle capacity constraints while optimizing a certain objective function such as total distance traveled. Another model is the Pickup and Delivery Problem with a Time Window (PDPTW), which is a generalization of the well-known Vehicle Routing Problem (VRP) [3] and was originally designed for terrestrial vehicles. The PDPTW optimization problem has inspired much research and has been well studied in the literature in the case of terrestrial vehicles. PDP and PDPTW, as most routing problems, are NP-hard problems [4]. These classes of problems can only be optimally solved for small instances, and for larger instance, various heuristics are commonly applied. For example, the authors in [5] proposed a tabu search heuristic to solve the PDP with a time window where vehicles have to transport goods from the origin position to the destination while satisfying load capacity and time constraints. A similar study was proposed in [6] where the authors considered multiple depots and adopted a genetic algorithm to solve the PDPTW. The work carried out in [7] also considered multiple depots and proposed three exact solutions to solve the PDPTW optimization problem. A new variant of the PDP called VRP with Simultaneous Delivery and Pickup and Time Windows (VRPSDPTW) was proposed in [8]. In this variant, the authors introduced five objectives and designed two meta-heuristics, namely Multi-Objective Local Search (MOLS) and the Multi-Objective Memetic Algorithm (MOMA), to solve the proposed optimization problem. Unlike PDPTW for terrestrial vehicles, only a few papers have focused on the PDP optimization problem for flying aerial vehicles. For example, the authors in [9] focused on data delivery using UAVs. They proposed a SMARTheuristic based on the cluster-first, route-second algorithm to determine UAV routes while minimizing the time needed to satisfy all requests (i.e., pickup data from a source and deliver it to a target). The study in [10] proposed a pickup and delivery service using both terrestrial and aerial vehicles. The authors developed two algorithms, the vehicle-driven and drone-driven routing heuristics, in order to optimize the planning of the vehicle-drone delivery service. For more details on the extended variants of routing optimization problems for UAV path optimization, the reader can refer to the survey in [11].

In our study, we focus on the Multi-Objective Evolutionary Algorithm (MOEA) technique to solve the PDPTW using multiple UAVs. We propose to use the elitist Non-dominated Sorting Genetic Algorithm II (NSGA-II) [12] to find a set of UAV routes minimizing three objective functions described as follows:

- Distance traveled: the sum of the distances traveled by all UAVs deployed.
- Schedule duration: the total time of all UAVs' tours. This time starts when a given UAV leaves the depot and ends when it returns. The schedule duration includes the UAV's flying time to visit the different sites of its tour, the service time at each site, and the refueling time (time to recharge the UAV's battery), if needed.
- The number of vehicles: the total number of UAVs used to perform all tasks.

The solutions sought include not only solutions that minimize one of the objective functions, but also solutions that dominate others over a combination of objectives (e.g., in the sense of Pareto optimality, according to the NSGA-II design). Genetic algorithms, in general, including the NSGA-II, have been applied to several variants of routing problems, such as the Multiple Objective Dial a Ride Problem (MO-DRP) [13], the bi-objective pickup and delivery problem [14], or the Pickup and Delivery Problem with Time Windows and Demands (PDPTW-D) [15].

In [15], the PDPTW-D optimization problem was proposed for terrestrial vehicles and was formulated as a multi-objective optimization problem. The solutions were then found by an application and adaption of the NSGA-II. To illustrate the efficiency of the proposed NSGA-II algorithm, the authors conducted simulation experiments based on PDPTW test instances created by [16]. The instances' definitions and best-known solutions of the PDPTW benchmark problems are available in [17]. In [13], the problem was the Dial A Ride Problem (DARP) motivated by the real example of tourist transportation companies in Portugal: a set of drivers are picking up and dropping off customers with minibus vehicles. The multi-objective cost functions that should be minimized are respectively

the total distance traveled by the vehicles, the wage difference of the drivers, and the total number of empty seats. The individuals in their NSGA-II variant are vectors that map every customer to a vehicle index. In [14], the problem was a PDP with just one vehicle (helicopter) and with two objective functions: a total cost that is equivalent to the total distance traveled and the weighted sum of arrival times, where weights give more importance to more urgent requests (unlike our case, which has time windows). The NSGA-II was used as well, and an individual was a single ordered list of sites, corresponding to the pickup site and the delivery site of each task.

The contributions of our study are as follows:

- A new formulation of the PDPTW adapted to the UAV context. The new optimization problem is called the PDPTW-UAV.
- A new solution of the PDPTW-UAV adapting and applying the NSGA-II algorithm. Our solution is different from the ones proposed in the literature [13–15] because the problem is different. For instance, compared to [15], our study considers UAVs' constraints (PDPTW-UAV), whereas their variant was PDPTW-D for vehicles.
- A validation of our NSGA-II solution based on the test instances with 100 nodes provided in [16].
- A performance evaluation of our PDPTW-UAV with UAVs' constraints (such as refueling).
- Provide benchmark results for the new PDPTW problem with UAV constraints.

The remainder of our article is organized as follows: In Section 2, we introduce and precisely define the PDPTW-UAV. In Section 3, we formalize it as a corresponding mathematical optimization problem. In Section 4, we describe our adaptation and application of the NSGA-II algorithm, a multi-objective version of a genetic algorithm, to solve the PDPTW-UAV problem. In Section 5, we first validate the improved NSGA-II, and then, we provide new benchmark PDPTW results for UAVs, useful for researchers in the field. In Section 6, we discuss some of the sources of uncertainty from the real-world context that can affect our problem solving and how to deal with these issues. Finally, we conclude in Section 7.

2. Use Case: Data Exchange Using UAVs

During natural disasters such as earthquakes, floods, landslides, or hurricanes, road and telecommunications infrastructures can suffer significant damage that renders them unusable. For instance, communication and network connectivity may be lost, or roads may become impassable and prevent emergency vehicles from reaching damaged areas, which we call critical sites. These sites therefore need the rapid deployment of a temporary solution to communicate with the other critical sites and with the emergency headquarters management center, also known as the central entity. In such a scenario, flying nodes such as UAVs can be used to provide intermittent communication between critical sites and the central entity, as shown in Figure 1. It illustrates a scenario in which several UAVs leave a truck, operating as a depot, to pick up data at given sites, deliver them to other sites while satisfying data latency, UAV load capacity, and UAV autonomy (i.e., UAV energy), and finally, fly back to the depot.

2.1. PDPTW-UAV: Definitions and Terminology

The Pickup and Delivery optimization Problem with a Time Window for UAVs (PDPTW-UAV) is an extended variant of the PDPTW originally designed for terrestrial vehicles. The PDPTW-UAV takes into account energy consumption and the limit of available energy and allows UAVs to recharge their batteries at any site if necessary during their services.

To better describe our optimization problem, we propose below the different terminologies used in the PDPTW-UAV.

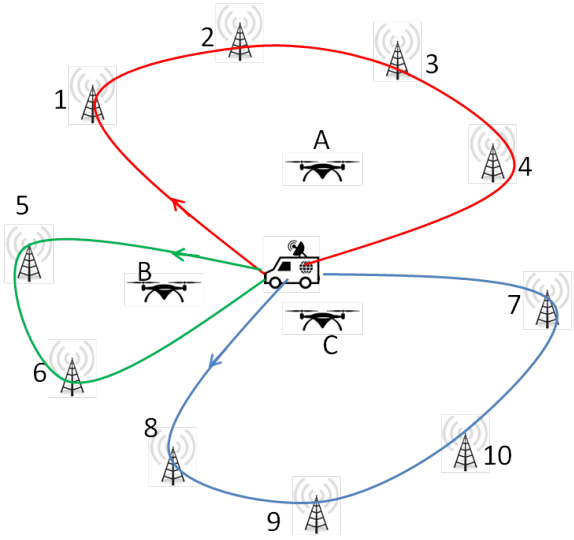

Figure 1. Illustrated example: UAVs tours.

- Data: Information to be carried from one site to another. These data can be of different types, such as text, photos, or videos. This information does not have a latency constraint (the data are carried by the use of the store-carry-forward mechanism). The size of the data is variable and can range from small to large.
- Task: It can be a pickup or delivery task. When one site has data to transmit to another, the UAV has to satisfy both types of tasks. It has to visit the source site to pick up data, then deliver them to the destination site. As described in [16], each task contains the following fields:
  - Task ID,
  - Site position (x,y),
  - Demand: data size,
  - Earliest pickup or delivery time,
  - Latest pickup or delivery time,
  - Service time,
  - Pickup task ID: if it is equal to zero, then this task is for data pickup,
  - Delivery task ID: if it is equal to zero, then this task is for data delivery.
- Central entity, also called the depot: This represents the point of departure and arrival of each UAV route. It is in charge of computing the next UAV tour based on the task list.
- Unmanned Aerial Vehicles (UAVs), also called drones: In our study, UAVs serve as data mules to provide temporary connectivity. They pick up data from sites and deliver them to other sites. UAVs have limited energy and storage capacity.
- Site: This is an access point that acts as a drop box. It contains data to be sent to other sites.

Objective, Constraints, and Assumptions

The PDPTW-UAV is a multi-objective optimization problem designed to minimize three objectives under a set of constraints, described as follows.

- Objectives:
  - Minimize the flying distance.
  - Minimize the schedule duration (including the refueling time if it is needed).
  - Minimize the number of UAVs used.
- Constraints:
  - Every UAV's tour starts and ends at the depot.
  - Every task is served exactly once: data are picked up or delivered once.

  - The pickup task is completed before the associated delivery task.
  - Pickup and delivery tasks for the same data are served by the same UAV. The tasks are then denoted paired tasks (or associated or corresponding).
  - The service time window is satisfied. To start service at any site, a UAV has to reach that site before the latest pickup time. If a UAV arrives at a site before the earliest time, it has to wait until the earliest time in order to service that site.
  - The UAV cannot carry more than its data storage capacity.
  - At each site, the remaining energy of the UAV must be sufficient to reach the next site. Otherwise, a full refueling is required.
- Assumptions:
  - There is one depot/central entity.
  - There are several disconnected sites.
  - The number of available UAVs is large enough for the needs of the model and does not represent a constraint.
  - All UAVs have the same storage capacity and autonomy.
  - When a UAV leaves the depot, its energy is full.
  - Any site may be a pickup or a delivery location. Note that it can be both the pickup location and a delivery location for two different tasks.
  - UAV refueling can take place at any site.
  - When a UAV does not have enough energy to reach the next site, its battery will always be fully recharged at the local site.
  - If a UAV needs to recharge its battery to reach the next site, refueling begins as soon as it arrives at the current site, and it can be carried out during the waiting time and the service time.

3. PDPTW-UAV: Problem Statement

In our study, we adopted the following notations for the formulation of the proposed PDPTW-UAV. Insights into the mathematical formulation of the variants of the pickup and delivery problems can be found for instance in [4]. The general case is that each site can be a pickup or delivery site for an arbitrary number of tasks. Our formulation assumes that a site is either the pickup site or the delivery site of exactly one task: this does not lose any of its generality since as many virtual sites can be created at the same location as there are tasks to be picked up or delivered at a site.

- N: the set of vertices or sites (any site is either a pickup or a delivery location); it is partitioned as $N = P \cup D$ defined below.
- n: the number of pickup sites.
- p: the number of delivery sites; we consider only the case of paired pickups and deliveries; hence, $p = n$.
- P: the set of pickup vertices $P = \{1, \ldots, n\}$.
- D: the set of delivery vertices $D = \{n + 1, \ldots, 2n\}$.
- 0: the depot or central entity location.
- K: the set of UAVs.
- q_i: the demand/supply at vertex i; $q_0 = 0$.
- d_i: the service duration at vertex i; $d_0 = 0$.
- c_{ij}^k: the cost in terms of the duration of the travel of UAV k from vertex i to vertex j.
- e_{ij}: the energy consumed by the UAV from vertex i to vertex j.
- $[\text{earliest}_i, \text{latest}_i]$: the time window for pickup or delivery at vertex i.
- w_i: the waiting time at vertex i if a UAV reaches i before earliest_i.
- E: the energy capacity of each UAV.
- C: the storage capacity of each UAV.
- R: the refueling duration necessary for refueling or changing the battery.

Without loss of generality and for the ease of mathematical formulation, we also assume a numbering such that the pickup task at vertex $i \in P$ is paired with a delivery task at vertex $j \in D$ with exactly $j = n + i$.

3.1. Binary Variables

In our PDPTW-UAV, we define two binary variables:

- $x_{ij}^k \in \{0,1\}$: one if the kth UAV goes straight from vertex i to vertex j.
- $y_i^k \in \{0,1\}$: one if the kth UAV refuels its battery at vertex i.

3.2. Fractional Variables

Let Q, B and E be three fractional variables used in our problem model for UAV load, UAV service time, and UAV energy, respectively.

- Q_i^k: the load of UAV k when leaving vertex i
- B_i^k: the time of the beginning of the service of UAV k at vertex i
- E_i^k: the remaining energy of UAV k after visiting vertex i.
- β_i^k: the additional refueling time of UAV k before leaving vertex i.

3.3. Problem Model

The PDPTW-UAV model can be written by minimizing the following objective functions:

$$\min \sum_{k \in K} \sum_{(i,j) \in N \times N} c_{ij}^k x_{ij}^k \tag{1}$$

$$min \sum_{k \in K} \sum_{(i,j) \in N \times N} (c_{ij}^k + w_j + d_j) x_{ij}^k + \sum_{k \in K} \sum_{i \in N} \beta_i^k y_i^k \tag{2}$$

$$\min \sum_{k \in K} x_{i0}^k \tag{3}$$

subject to:

$$\sum_{k \in K} \sum_{(i,j) \in N \times N} x_{ij}^k = 1 \qquad \forall k \in K \tag{4}$$

$$\sum_{j \in N} x_{0j}^k = 1 \qquad \forall k \in K \tag{5}$$

$$\sum_{i \in N} x_{i0}^k = 1 \qquad \forall k \in K \tag{6}$$

$$\sum_{j \in N} x_{ji}^k - \sum_{j \in N} x_{ij}^k = 0 \qquad \forall i \in N, k \in K \tag{7}$$

$$\beta_i^k = \max(R - (w_i^k + d_i), 0) \qquad \forall i \in N, k \in K \tag{8}$$

$$B_i^k < B_{i+n}^k \qquad \forall i \in P, k \in K \tag{9}$$

$$\sum_{j \in N} x_{ij}^k - \sum_{j \in N} x_{i+n,j}^k = 0 \qquad \forall i \in N, k \in K. \tag{10}$$

$$x_{ij}^k = 1 \Longrightarrow B_i^k + d_i + \beta_i^k + c_{ij}^k \leq B_j^k \qquad \forall (i,j) \in N \times N, i \neq j, k \in K \tag{11}$$

$$x_{ij}^k = 1 \Longrightarrow \text{earliest}_j \leq B_j^k \leq \text{latest}_j \qquad \forall (i,j) \in N \times N, k \in K \tag{12}$$

$$x_{ij}^k = 1 \Longrightarrow Q_j^k = Q_i^k + q_j \qquad \forall (i,j) \in N \times N, k \in K \tag{13}$$

$$Q_i^k \leq \min\{C, C + q_i\} \qquad \forall i \in N, k \in K \tag{14}$$

$$e_{ij} x_{ij}^k \leq E_i^k \leq E \qquad \forall (i,j) \in N \times N, k \in K \tag{15}$$

$$x_{ij}^k = 1 \text{ and } y_i^k = 1 \Longrightarrow E_i^k = E \qquad \forall (i,j) \in N \times N, k \in K \tag{16}$$

The objective function (1) minimizes the total flight time of UAVs to satisfy all tasks: we assume a constant speed, which is equivalent to minimizing the total flight distance. The objective function (2) minimizes the total schedule duration, which is the sum of the flight time, waiting time, service time, and refueling duration if needed for all the UAV routes. The last objective function (3) minimizes the number of UAVs used.

The model constraint (4) ensures that each task is served exactly once. The constraints (5)–(7) guarantee that every UAV leaves the depot and returns to it only once during its tour and that the same UAV that enters a node leaves the node. The constraint (8) is to check the additional schedule time due to the refueling operation. The constraint (9) is the precedence constraint to check that the pickup task is satisfied before its corresponding delivery task, and the constraint (10) ensures that both tasks are serviced by the same UAV. The constraints (11) are the schedule time constraints. Constraints (12) are the time window constraints in which a service time cannot be started after the latest time, and if a UAV arrives before the earliest time, it has to wait until the earliest time to start the service. Constraints (13) and (14) aim at satisfying the storage capacity of each UAV. Finally, the constraints (15) and (16) consist of first checking the remaining UAV energy, then indicating whether refueling is necessary.

4. Solving the PDPTW-UAV Using the NSGA-II

Version 2 of the Non-dominated Sorting Genetic Algorithm [12] (NSGA-II) is designed to solve multi-objective optimization problems. In our study, we applied the NSGA-II, as illustrated in Algorithm 1, to solve our PDPTW-UAV. The proposed use of the NSGA-II algorithm starts by generating an initial P population of N individuals. Then, at each iteration, the NSGA-II randomly selects two individuals called parents from the P population, applies a crossover operator (Section 4.1.2) to them to obtain two new individuals called children, which will be mutated (Section 4.1.3) before being added in the new offspring Q population. Both parents and offspring populations are merged into a single set called R. Based on the non-dominated sorting approach and crowding distance sorting [12], only the best solutions passing the selection step are kept and included in the population of the next iteration P_{t+1}, while the other individuals are removed. Let $P_{t+1} = F_1, F_2, F_3, \ldots, F_i$, where F_i is the set of solutions in the Pareto front of level i.

If only a few solutions of the last front F_i should be kept in P_{t+1}, these solutions are chosen to have the best crowding distance. Therefore, $N - T$ solutions will be chosen from the last front set F_i, where N is the size of the population and T is the sum of the size of sets $F_1, F_2, ..., F_{i-1}$. Figure 2 illustrates the non-dominated sorting genetic algorithm process.

The principles of the NSGA-II are based on non-dominating sorting techniques, crowding distance techniques, and elitist techniques. The NSGA-II is characterized by the fact that it first provides solutions close to the Pareto-optimal ones, then it allows for a diversity of solutions, and finally, it preserves the best solution at each iteration for the next iteration.

Although the NSGA-II is an excellent algorithm for solving multi-objective optimization problems, it may require a large number of iterations and a long runtime to converge to the best solutions found. To speed up the convergence of the NSGA-II, we propose to generate good individuals for the initial population, as illustrated in Algorithm 2, and also to improve each individual of the offspring population.

Algorithm 1 NSGA-II for PDPTW-UAV.

N: {Population size}
P_0: {Initial population}
P: {Parent population}
Q: {Offspring population}
for $i = 1$ to N **do**
  $P_0 \leftarrow P_0 \cup$ Generate Individual {After improvement}
end for
$P_t \leftarrow P_0$
while $t < Max\_Iterations$ **do**
  for $s = 1$ to $N/2$ **do**
    $(Parent_1, Parent_2) \leftarrow$ randomly selected from P_t
    $(Child_1, Child_2) \leftarrow$ Crossover $(Parent_1, Parent_2)$
    Mutate $(Child_1, Child_2)$
    Improve $(Child_1, Child_2)$ {As explained in Section 4.2}
    Compute fitness values of $(Child_1, Child_2)$
    Add $(Child_1, Child_2)$ to Q_t
  end for
  $R_t \leftarrow P_t \cup Q_t$ {Combing parent and offspring population}
  $F_1, F_2, \ldots F_k \leftarrow$ result of a fast non-dominated sort of R_t
  $P_{t+1} \leftarrow \emptyset$
  $i \leftarrow 1$
  while $(|P_{t+1}| + |F_i| < N)$ **do**
    $P_{t+1} \leftarrow P_{t+1} \cup F_i$
    $i \leftarrow i + 1$
  end while
  {i now corresponds to the first set F_i, which cannot be entirely included in P_{t+1}}
  Compute the crowding distance in F_i
  $P_{t+1} \leftarrow P_{t+1} \cup \{(N - |P_{t+1}|)$ first solutions in $F_i\}$
  $t \leftarrow t + 1$
end while

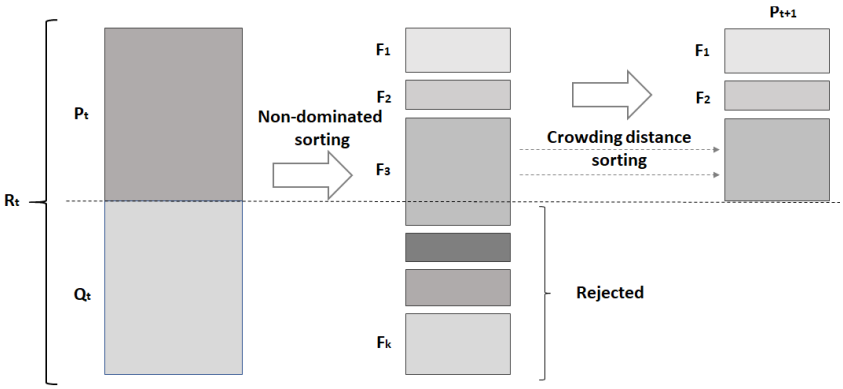

Figure 2. The NSGA-II.

Algorithm 2 Generate individual.

L : {list of all tasks}
l_i : {list of tasks served by UAV_i}
t : {task}
t_p : {pickup task}
t_d : {delivery task}
$i \leftarrow 1$; {indicates the number of UAVs}
while L is not empty **do**
  $l_i \leftarrow$ empty list
  Select paired tasks (t_p, t_d) randomly from L
  Append (t_p, t_d) to l_i
  Remove (t_p, t_d) from L
  repeat
    $T \leftarrow$ the list of paired tasks of L that could be added to l_i without violating constraints (including refueling)
    if T is not empty **then**
      Select the (t'_p, t'_d) in T that minimizes the total flying distance increase, when added to l_i
      Append (t'_p, t'_d) to l_i
      Remove (t'_p, t'_d) from L
    end if
  until T is empty
  $i \leftarrow i + 1$
  Evaluate fitness functions
end while

4.1. Genetic Operations

In this section, we specify the terminologies used for the NSGA-II [12] and the main elements of the genetic algorithm (individual or chromosome, crossover, and mutation).

4.1.1. Individual Representation

In genetic algorithms, a population is defined as a set of individuals called chromosomes, where each of them could be seen as a solution to the problem to be solved. Each individual is made up of a set of variables known as genes that form the chromosome. In our optimization problem, an individual is a vector of UAV IDs. The size of this vector is equal to the number of tasks to be performed. Each position in the vector refers to a task ID. Then, each UAV is a gene, and it occupies a cell of the vector. In our individual representation, a UAV should appear in at least two cells, reflecting the fact that the pickup task and the corresponding delivery task are served by the same UAV.

Consider the example of a pickup and delivery problem with 10 tasks specified in Table 1: each of the tasks, identified by a task ID, is paired with another corresponding task with a different task ID and with the opposite action. The numbering in the problem definition can be arbitrary (contrary to the mathematical formulation), and in the implementation, the "Action" column can be deduced from the action of the associated task.

Table 1. Task table example.

Task ID	Action	Associated Task (Task ID)	
		Pickup	Delivery
1	pickup	-	4
2	pickup	-	3
3	delivery	2	-
4	delivery	1	-
5	pickup	-	6
6	delivery	5	-
7	delivery	9	-
8	pickup	-	10
9	pickup	-	7
10	delivery	8	-

Figure 3 illustrates an example of an "individual" solution to satisfy the problem.

In this solution, three UAVs are used, denoted: `A`, `B`, and `C`. The UAV `A` is associated with tasks with the identifiers 1, 2, 3, and 4; the UAV `B` with Tasks 5 and 6; and the UAV `C` with Tasks 7, 8, 9, and 10. A possible corresponding tour of the UAV could be represented as in Figure 1. Although the tour of each UAV starts and ends at the depot, the depot is not considered as a task to be served and does not appear in the individual representation. The order of the tour itself is determined as described in Section 4.2.1.

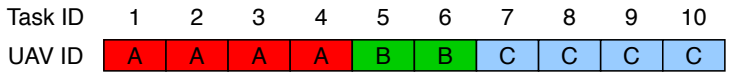

Figure 3. Individual representation.

We define a valid individual as a solution for which the assignment of tasks to UAVs (represented by the gene vector) satisfies the constraints defined above. In our problem, we generate each initial individual as described in Algorithm 2 to get an initial population of valid individuals and to speed up the convergence of the NSGA II. The random aspect of the generation of individuals comes from the random selection of the first task assigned to a UAV tour.

Then, a greedy selection of the other tasks assigned to the UAV is made as follows. The algorithm iteratively selects the "best" task, i.e., the one that minimizes the distance traveled by the UAV while satisfying the previously mentioned constraints (refueling constraints, capacity constraints, visit time constraints, and maximum travel time constraints). This process is carried out until it is no longer possible to add new tasks to the current UAV tour, in which case, the next UAV tour is generated (if necessary). Thanks to this greedy initialization, the initial individuals not only satisfy the constraints, and are therefore valid individuals, but they already perform well in terms of the objective functions (1) and (3).

4.1.2. Crossover Operation

The crossover operation is the action of generating an offspring population from a valid population. It consists of exchanging genes between two parents and creating new children inheriting characteristics of both parents. In our problem, the offspring population is obtained based on the two-point crossover operation [18]. In this operation, two individuals are selected, and the vector corresponding to each of them is fragmented into three parts on the basis of two randomly chosen vector positions. The first child vector is a recombination of two fragments from Parent 2 and one fragment from Parent 1, and the second child vector is created from the remaining fragments of both parents, as illustrated in Figure 4.

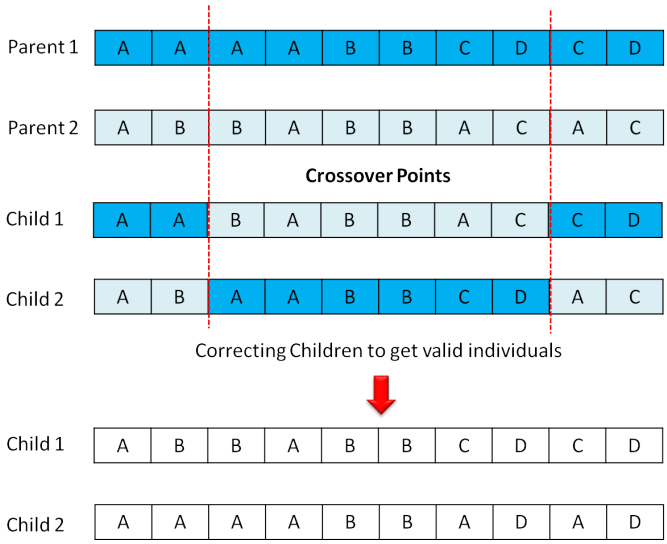

Figure 4. Crossover operation.

To obtain an offspring population composed of valid individuals, we make some modifications to the genes of the children to satisfy the constraints of our problem.

4.1.3. Mutation Operation

In genetic algorithms, a mutation is a random alteration of certain genes. In our optimization problem, a mutation operation consists of, on the one hand, randomly selecting a site belonging to the tour of one UAV and, on the other hand, injecting this site into the tour of another UAV. This operation is repeated k times. k is randomly selected. The individual resulting from the mutation operation may need a few corrections to become a valid individual, as in the case of the crossover operation. The mutation operation is illustrated in Figure 5.

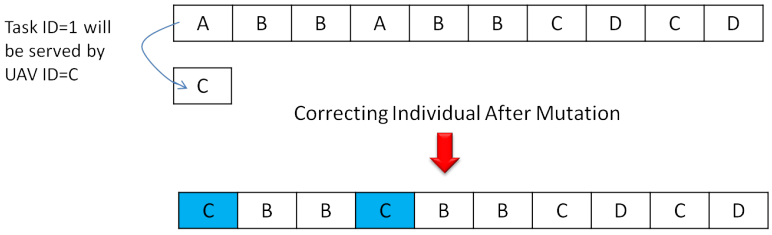

Figure 5. Mutation operation.

4.1.4. Selection Operation

The solutions are first sorted using a non-dominated sorting method in ascending order (F_1, F_2, F_3, F_n), and in each Pareto rank, the points are sorted according to crowding distance in descending order. If only a few solutions of the last front F_j need to be added to P_{t+1}, these solutions are chosen to have the best crowding distance.

To select the best solutions from the F_j set that can ensure the preservation diversity, a measure of solution density in the objective space is used. As defined in [12], we used the crowding distance to estimate the density of solutions surrounding a particular solution in a non-dominated F_j set. As shown in Figure 6, the crowding distance of the i^{th} solution corresponds to the semi-perimeter of the cuboid whose vertices are the closest neighbors of the solution i.

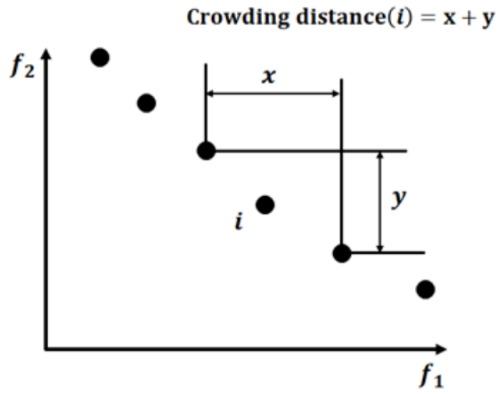

Figure 6. Crowding distance.

4.2. Heuristic Algorithms

In this section, we describe the different heuristics used in the NSGA II algorithm to improve the PDPTW solutions and to speed up its convergence.

4.2.1. Refinement UAV Task List Algorithm

The order in which a UAV visits the set of tasks to be served is very important. It decides the order of the UAV's tour, its length, its duration, and the number of times the UAV needs to refuel. In our study, we propose a branch and bound algorithm to refine each UAV task list according to the following constraints:

- The visiting time for each task must be satisfied. Thus, the UAV must start the task service between its earliest and latest times.
- The pickup task must be served before the corresponding delivery task. Note that a UAV can serve several tasks between a pickup and its corresponding delivery task if the UAV's capacity constraint is fulfilled.
- UAV capacity constraints must be met.
- The refueling requirement rules must be applied.

The objective is to minimize the travel distance (or equivalently, the duration of the flight). The travel distances of the best candidate solutions, already visited and valid, can be used as the upper bounds in the branch-and-bound search. This allows quickly discarding some branches that would correspond to necessarily worse candidate solutions. This refinement algorithm provides a near-optimal, but not absolutely optimal solution since we do not enumerate all refueling possibilities. In the solutions considered, a UAV only refuels when its remaining energy is not sufficient to reach the next site.

4.2.2. Individual Improvement Algorithm

A valid individual could be improved by reducing the total flight distance of UAVs and the number of UAVs deployed. For this, we first propose a greedy algorithm that, iteratively for each task, tries to place it in the best UAV task list, so that the total flight distance is reduced. Then, we propose an additional greedy algorithm that reduces the number of UAVs used to serve all tasks.

In this algorithm, we reduced the number of UAVs by eliminating the smallest tours if possible. To do this, we sorted the UAV' tours according to their number of tasks. Then, we tried to insert all the tasks from the smallest UAV tour into the other UAV tour as long as no other changes were made to the individual.

4.2.3. Individual Correction Algorithm

Crossover or mutation operations are very important in genetic algorithms. While the crossover operation enables generating new offspring and finding new solutions that converge to a local minimum, the mutation is a divergence operation that tends to add new

genetic characteristics in order to perform a global search on the space of solutions and to avoid the local minimum. After the application of the crossover or mutation operations, the resulting individual may not be valid, as described in Sections 4.1.2 and 4.1.3.

In order to obtain a valid individual after applying the crossover or mutation operations, we proceed as follow:

- For each UAV's tour:
  - We first check the constraint (10), which indicates that a pickup task and a corresponding delivery task must be performed by the same UAV. If this constraint is not satisfied for some tasks, we apply the following rule: a delivery task must follow its corresponding pickup task. Thus, for each of the paired tasks that violate (10), the delivery task of the pair is moved to the UAV task list of the corresponding pickup task (and consequently removed from its previous UAV task list).
  - Next, the UAV task lists are sorted as explained previously in Section 4.2.1.
  - If the sorted list does not satisfy the constraints (10) to (16), then we reformat a valid UAV tour for this UAV task list based on Algorithm 2 (more precisely, the sub-algorithm in Lines 8 to 19, with L = UAV task list).
- Tasks not used in the newly generated UAV tour will either be added to other UAV tours or used to generate one or more new UAV tours.

4.3. Refueling Constraints' Verification Algorithm

In the previously defined algorithms, the constraints must be checked to ensure the validity of the individuals. Each of them is associated with sub-algorithms. For the specific case of refueling constraints, an algorithm is defined in this section. In order to avoid obtaining an invalid individual, the algorithm checks the refueling constraints each time the individual is modified (generation, mutation, crossover, individual improvement).

Note that energy limitation is one of the major constraints for UAVs. In our study, we assumed that each UAV can refuel during its tour at any site it visits. The refueling process can have an impact on the schedule times. Indeed, the UAV must travel to the site to serve during a time window delimited by the earliest and latest time of the given problem instance. As a result of the refueling process, the UAV may be delayed and be forced to reach a site after the last authorized time. For this purpose, we designed the refueling constraints' verification algorithm that actually tries to compute the schedule while considering refueling time.

This algorithm is called for each task list of the individual's UAV. It determines if the individual is valid or not, and if so, it obtains the total schedule time. Algorithm 3 illustrates the steps of calculating the schedule times with refueling.

Algorithm 3 considers two cases to calculate the refueling time: the first one is when a UAV arrives at a site before its earliest time, and the second case is when it arrives after its earliest time and before its latest time. After some initialization (Lines 1–12), the algorithm starts by checking whether the UAV reaches a site before or after its earliest time (Line 13). If the UAV reaches a site before its earliest time, then the algorithm computes a waiting time (Line 14) that could be considered when performing the UAV's refueling (Line 15). In Lines 15 to 27, the algorithm checks whether the UAV's refueling is needed or not. If this is the case, then the refueling operation will start at the waiting time, and it can last until the service time (Line 19) or even exceed it (Line 25). Line 28 of the algorithm is executed when the UAV reaches a site after its earliest time and before its latest time. The refueling is performed if the UAV does not have enough energy to serve the current site and reaches the next site (Lines 32–34). Additional time is added to the scheduled time, if the refueling time exceeds the service time (Lines 38–39).

Algorithm 3 Refueling algorithm for a UAV tour/.

$E \leftarrow$ UAV maximum autonomy; {E is the remaining UAV energy expressed in flight duration}
$W \leftarrow 0$; {Waiting time}
$l \leftarrow$ UAV task list
$ScheduleTime \leftarrow 0$
t_i: {time to travel from the previous task site to the site of the i-th task of the task list l}
$earliest_i$: {earliest time}
$latest_i$: {latest time}
$service_i$: {service time}
FRT: {Full refueling time (constant)}
for $i = 1$ to $|l|$ **do**
  $ScheduleTime \leftarrow ScheduleTime + t_i$
  $E \leftarrow E - t_i$
  if $ScheduleTime \leq earliest_i$ **then**
    $W \leftarrow earliest_i - ScheduleTime$
    if $E > (W + service_i + t_{i+1})$ **then**
      $ScheduleTime \leftarrow earliest_i + service_i$
      $E \leftarrow E - (W + service_i)$
    else
      {need refueling during the service time and the waiting time}
      $E \leftarrow UAV\ maximum\ autonomy$
      if $(W + service_i) > FRT$ **then**
        $ScheduleTime \leftarrow earliest_i + service_i$
        $E \leftarrow E - (W + service_i - FRT)$
      else
        $ScheduleTime \leftarrow ScheduleTime + FRT$
      end if
    end if
  else if $ScheduleTime \leq latest_i$ **then**
    if $E > (service + t_{i+1})$ **then**
      $ScheduleTime \leftarrow ScheduleTime + service_i$
      $E \leftarrow E - service_i$
    else
      {need refueling during the service time}
      $E \leftarrow UAV\ maximum\ autonomy$
      if $service_i > FRT$ **then**
        $ScheduleTime \leftarrow ScheduleTime + service_i$
        $E \leftarrow E - (service_i - FRT)$
      else
        $ScheduleTime \leftarrow ScheduleTime + FRT$
      end if
    end if
  else
    {Refueling not possible; the individual is not valid}
  end if
end for

5. Performance Evaluation

In this study, we conducted a set of experiments to, firstly, validate our improved version of the NSGA-II against those proposed in [15,16] based on the well-known Lim benchmark [17] and, secondly, to solve the PDPTW-UAV. To the best of our knowledge, our study is the first to give results for the Lim benchmark [17] in the context of UAVs, so the results will be available for researchers to compare the performance of different future approaches.

5.1. Simulation Parameters

We conducted a series of experiments using the first nine instances of those proposed in [17]. Each instance included 100 pickup and delivery tasks. Since we added some heuristics to improve the NSGA-II and help it converge quickly, we generated only 10 individual populations for 100 generations. Table 2 illustrates the simulation parameters used in this study.

Table 2. Simulation parameters.

Problem Parameters	**Values**
Instances	9 test instances from [17]
Task's number	100 per instance
Maximum UAV number	25
Maximum capacity	200
Velocity	1 (not used in the instances)
UAV autonomy	500
Refueling duration	60, 90, 120
Genetic algorithm parameters	**Values**
Population size	10
Generation size	100

We conducted 15 simulation runs, for each of the nine benchmark instances. These 15 runs were actually based on 15 different initial populations that were generated using Algorithm 2. For each initial population, we applied separately the NSGA-II to generate the next generation (based on crossover and mutation operators) and selected the best 10 individuals resulting from the non-dominated sorting algorithm, as illustrated in Figure 2. This process was repeated 100 times (which was the number of generations, also called iterations). The final result of this simulation run was the Pareto front of one run. In this manner, we could collect 15 different Pareto fronts from the 15 separate runs. Our final step was the extraction of the non-dominated individuals from all the 15 Pareto fronts, selecting the ones that minimized the three initial objectives considered in our study. This yielded a final, global, Pareto front.

Notice that the results shown in the benchmark do not have any known unit for the time, distance, or speed; hence, we followed the same assumption to be compliant with that study.

5.2. Solving PDPTW Using Our NSGA-II

In this section, we present the simulation results of the improved NSGA-II without considering refueling using the instances from [17]. The aim was to prove the efficiency of the proposed NSGA-II by comparing the simulation results obtained with those presented in the literature [15,16]. Table 3 summarizes the comparative evaluation of our study with two other studies. Notice that in our work, we optimized three objective functions: number of vehicles, distance traveled, and schedule times. However, the studies proposed in the literature only focused on minimizing the number of vehicles and the distance traveled. For our MOEA proposalm, we included all the solutions obtained in the Pareto front. We solved the first nine instances of 100 tasks from [17] using our NSGA-II proposal. The results obtained were the same as those of [15,16] and differed only for two instances (e.g., instances lc103and lc109), wuth a slight difference. The Pareto front of some instances where several non-dominated solutions were found is represented in Figure 7.

Based on these very satisfactory results, we concluded that the proposed NSGA-II gave good results when applied to the PDPTW. We could now rely on our NSGA-II to solve the PDPTW-UAV while considering the energy constraints of UAVs.

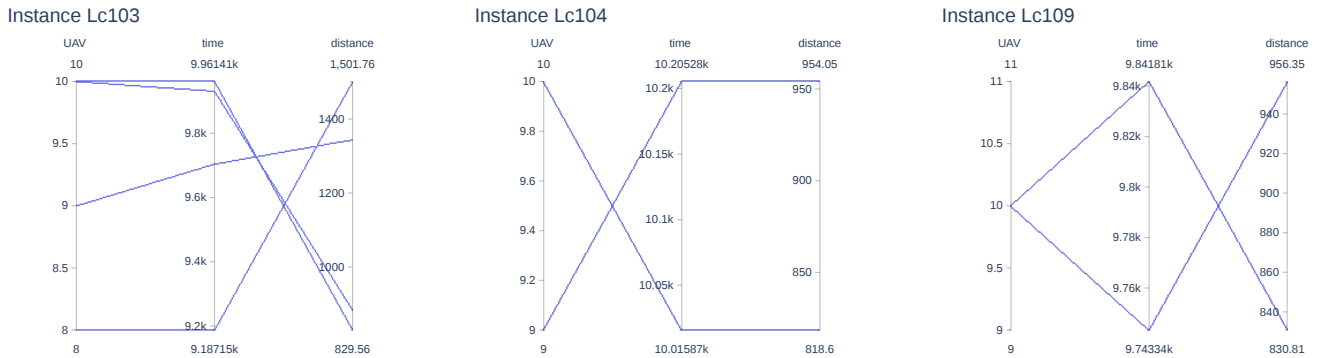

Figure 7. Pareto front of instances with multiple non-dominated solutions.

Table 3. Instance validation.

	Best Known [17]		**EMOA [15]**		**Our Proposed MOEA**			
Instance	**# veh**	**Dist**	**# veh**	**Dist**	**# veh**	**Dist**	**S.Time**	**W.Time**
lc101	10	828.94	10	828.94	10	828.94	9828.94	0.00
lc102	10	828.94	10	828.94	10	828.94	9828.94	0.00
lc103	9	1035.35	10	827.86	10 10 9 8	829.56 883.74 1344.41 1501.76	9961.41 9930.89 9704.38 9187.15	131.85 47.15 709.97 1385.39
lc104	9	860.01	9	917.70	9 10	954.05 818.60	10205.28 10015.87	431.23 197.27
lc105	10	828.94	10	828.94	10	828.94	9828.94	0.00
lc106	10	828.94	10	828.94	10	828.94	9828.94	0.00
lc107	10	828.94	10	828.94	10	828.94	9828.94	0.00
lc108	10	826.44	10	828.94	10	826.44	9826.94	0.00
lc109	9	1000.60	9 10	1068.59 827.82	10 10	830.81 956.35	9841.81 9743.34	11.00 1226.99

5.3. Convergence of Our NSGA-II Algorithm

In this section, we study the convergence of our algorithm. For that, we illustrate in Table 4 the Pareto front of each instance, from the first iteration to the iteration where the optimal solution appeared for the first time. Note that simulation runs were conducted using a desktop computer Fourth-Generation Intel(R) Core i7 Processors with 2.9 GHz and 8 Gb of cache memory. The convergence of our algorithm for instances lc101,lc102, lc105, lc106, lc107, lc108, and lc109 was very rapid since the optimal solutions were obtained during the first three iterations. However, instances lc103 and lc104 took longer to converge compared to the previously mentioned instances. Due to the different heuristics used to improve the individual, our NSGA-II algorithm converged to the optimal solution in only a few iterations and in a short simulation time, as shown in Table 4.

In our study, we assumed that a UAV must start refueling, if necessary, when it reaches a site. Hence, the refueling was always performed during the service time. It can also be performed earlier during the waiting time if the UAV has arrived before the earliest time of the site visit. The refueling time could then be totally covered by the waiting and service time. In such a case, the refueling operation had no impact on the UAV's tour. However, when the refueling time exceeded the service time, the UAV tour was not the same as for the original PDPTW optimization problem. Due to the additional refueling time, the UAV

may reach a site after the latest time of that site visit. In this case, the UAV tour is invalid, and the individual is corrected based on the algorithms described in Section 4.2.

Table 4. Convergence table including the Pareto front for each instance at each iteration.

	Iteration	Sim. Time (min)	Number of Vehicles	Distance	Schedule Time
lc101	**1**	**5.50**	10	**828.94**	9828.94
lc102	**1**	**5.23**	10	**828.94**	9828.94
lc103	1	3.89	10	922.06	10,312.80
			10	1170.13	10,362.34
	2	9.15	10	883.74	9930.89
	14	36.77	10	883.75	9930.89
			9	2034.00	10,687.14
	16	38.33	10	883.75	9930.89
			9	1903.89	10,703.33
	39	78.75	10	883.74	9930.89
			9	1750.61	9870.77
	40	80.50	10	883.74	9930.89
			9	1669.33	9828.81
	42	83.45	10	883.74	9930.89
			9	1669.33	9828.81
			10	860.95	10,242.13
	43	**85.36**	10	883.74	9930.89
			9	1669.33	9828.81
			10	**829.56**	9961.41
lc104	1	10.15	9	1128.16	10,320.64
			9	1163.18	10,306.54
			10	1037.35	10,795.21
	2	22.64	9	1099.38	10,348.29
			9	1125.95	10,269.31
			10	1037.35	10,795.21
			10	1054.54	10,528.29
	3	32.34	9	1099.38	10,348.29
			10	974.34	10,477.76
			9	974.34	10,402.67
	4	106.06	10	872.80	10,070.08
			9	954.05	10,205.28
	7	**141.74**	10	**818.60**	10,015.87
			9	954.05	10,205.28
lc105	**1**	**3.68**	10	**828.94**	9828.94
lc106	1	2.83	11	948.02	10,616.47
	2	9.57	11	948.02	10,616.47
			11	924.63	10,728.95
			10	957.02	10,128.62
	3	**12.81**	10	**828.94**	9828.94
lc107	1	8.04	10	832.63	9971.26
			10	919.93	9919.93
	2	**16.70**	10	**828.94**	9828.94
lc108	1	4.77	10	866.1	9970.99
	2	**7.50**	10	**826.44**	9826.44
lc109	**1**	**7.65**	10	956.35	9743.34
			10	**830.81**	9841.81

In this section, we solve our PDPTW-UAV using the proposed NSGA-II algorithm. We conducted a series of experiments using the same nine instances from [17] as in the previous section. As the refueling time had a significant impact on the UAV tour, we present all the solutions obtained in the Pareto front for a refueling duration equal to 120, 90, and 60, as

illustrated in Tables 5–7, respectively. Moreover, for more clarity, we visualize the Pareto fronts in Figure 8 corresponding to the solutions of Table 5.

Table 5. Our solution with refueling = 120.

Instance	UAV's Number	Distance	Schedule Time	Refueling Time	Refueling Times/Number
lc101	11	1266.22	11217.61	285.00	15
	12	1182.28	11485.72	316.29	14
lc102	14	1172.25	12,964.84	326.00	17
	12	1462.26	11,673.71	331.49	14
	13	1233.02	12,211.52	271.49	14
	12	1242.50	11,778.76	346.89	15
	13	1225.03	12,228.82	274.55	15
	12	1134.90	11,367.10	345.00	15
	10	1266.22	10,142.39	210.00	11
lc103	9	1247.76	10,127.65	300.00	14
	10	1104.99	10,987.30	351.83	14
	11	962.47	11,586.83	381.83	17
	9	1276.77	10,003.61	240.00	13
	11	964.32	11,063.51	381.83	15
	10	1130.96	10,299.71	240.00	13
	11	961.18	11,675.98	381.83	17
lc104	9	1018.93	10,552.24	480.00	17
	10	824.06	10,515.49	360.00	14
lc105	10	834.45	10,324.26	330.00	12
	10	834.92	10,251.70	390.00	13
lc106	11	946.71	10,992.46	390.00	14
	11	1039.85	10,938.36	330.00	14
lc107	10	832.62	10,361.26	390.00	13
lc108	10	834.12	10,221.94	330.00	11
	9	1789.18	9936.07	300.00	11
lc109	10	964.77	10,256.11	360.00	13
	11	866.80	10,363.12	390.00	13
	10	894.65	10,321.41	420.00	14
	10	876.32	10,325.29	360.00	13

Table 6. Our solution with refueling = 90 (the refueling time is 0.00 because the refueling operation is performed during the service time).

Instance	UAV's Number	Distance	Schedule Time	Refueling Time	Refueling Times/Number
lc101	10	828.94	9828.94	0.00	12
lc102	10	828.94	9828.94	0.00	12
lc103	10	829.56	9961.41	0.00	13
	9	1099.77	9820.78	0.00	17
	10	829.45	10,212.06	0.00	15
lc104	10	950.67	10,212.56	0.00	13
	9	989.79	10,295.67	0.00	15
	9	914.25	10,338.50	0.00	16
	9	1059.70	10,273.97	0.00	17
	9	1048.72	10,288.33	0.00	17
	9	1078.03	10,107.50	0.00	14
lc105	9	1055.55	9206.83	0.00	11
	10	828.94	9828.94	0.00	12
lc106	10	828.94	9828.94	0.00	12
lc107	10	828.94	9828.94	0.00	12
lc108	10	826.44	9826.44	0.00	11
lc109	10	869.53	9878.88	0.00	12

Table 7. Our solution with refueling = 60 (the refueling time is 0.00 because the refueling operation is performed during the service time).

Instance	UAV's Number	Distance	Schedule Time	Refueling Time	Refueling Times/Number
lc101	10	828.94	9828.94	0.00	14
lc102	10	828.94	9828.94	0.00	14
lc103	10	835.05	9966.91	0.00	16
	7	1238.57	8589.78	0.00	14
	8	1136.98	8963.44	0.00	16
	10	829.45	10,212.06	0.00	15
lc104	9	957.54	10,060.65	0.00	16
	10	818.60	10,015.87	0.00	16
lc105	10	828.94	9828.94	0.00	14
lc106	10	828.94	9828.94	0.00	14
lc107	10	828.94	9828.94	0.00	14
lc108	10	826.44	9826.44	0.00	13
lc109	9	1256.52	9456.03	0.00	15
	10	830.81	9841.81	0.00	14

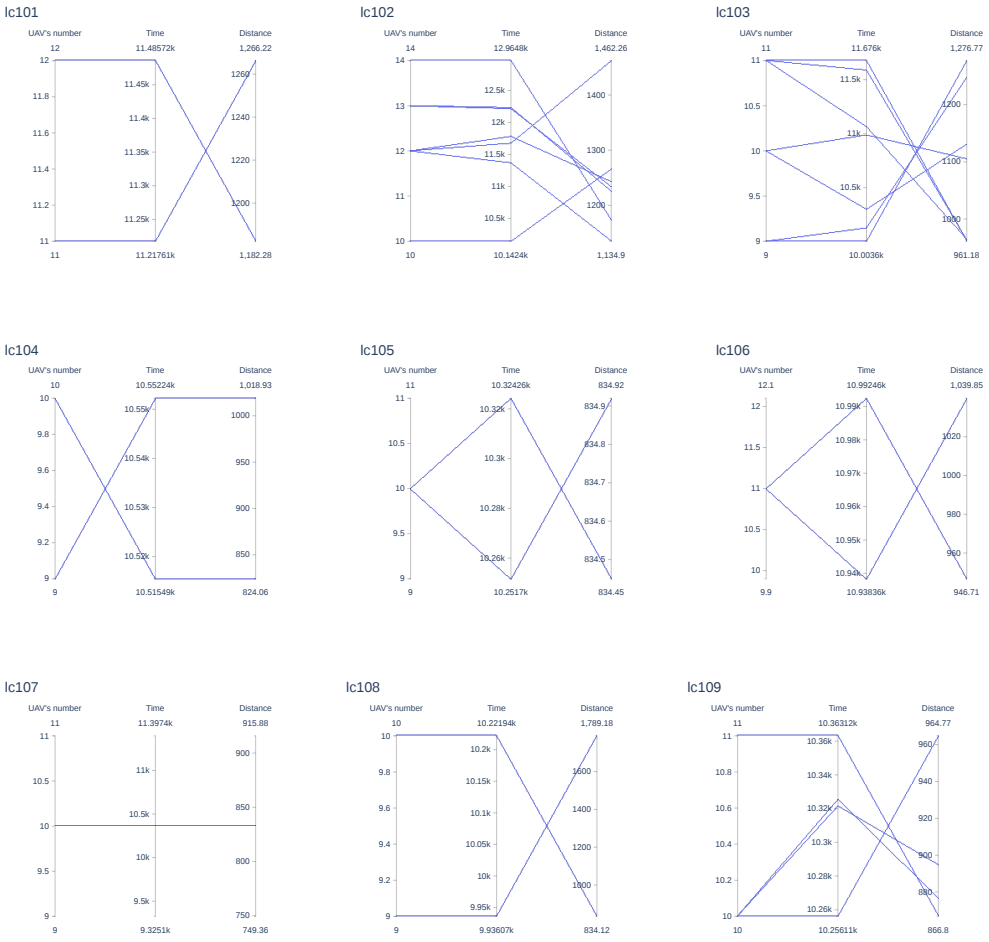

Figure 8. Pareto front of instances with multiple non-dominated solutions, with refueling time 120.

5.4. Solving the PDPTW-UAV Using Our NSGA-II

In Tables 5–7, we present for each instance the number of UAVs deployed, the cumulated flight distance, the schedule time, the cumulated additional refueling time, and the number of times the UAVs needed to refuel. Refueling time is the additional time that a

UAV spends refueling after the service time has expired. The refueling time is equal to zero if it is covered by the waiting time and/or service time.

When the refueling time was longer than the service time for all instances, we obtained solutions totally different from those presented in Table 3. This was due to the impact of the additional refueling time on the UAV's tours. In the new solutions shown in Table 5, we can observe an increase in flight distance and schedule time compared to the results obtained in Table 3.

However, when the refueling time was less than or equal to the service time in most instances, the additional refueling time was always equal to zero, as shown in Tables 6 and 7, and the results obtained were very close to the results of Table 3. This can be explained by the fact that a UAV recharges its battery during the service time, and then, there is no additional time to consider for refueling. As the service time is computed in the schedule time, the refueling time is considered as zero.

6. Coping with Uncertainties and a Real-World Scenario

In this article, we considered the PDPTW-UAV, which can be mathematically and exactly formulated as in Section 3, Equations (1)–(16). We proposed the NSGA-II algorithm to solve it. However, when considering the problem in the real world, the formulation is no longer accurate.The goal of this section is to give an overview and relevant references to solve the problem when uncertainties exist. Indeed, each constraint, such as the battery capacity, is known up to a certain approximation and, in addition, can be subject to random events, for instance the travel time from Point A to Point B; think of a situation with strong wind. In a case where the constraints are not really strict constraints, it is possible to solve the problem with the expected values, to consider the most pessimistic scenarios, or to include margins of error, for example to add 20% to the travel time to account for the uncertainties.

However, more methodologically sound approaches have been adopted: many researchers have properly formalized new variants of the considered problems that explicitly take into account randomness and uncertainty. One example of such variants is the "stochastic vehicle routing problem(s)" extending the "vehicle routing problem" with several options, and some related surveys can be found in [19] or [20], for instance.

Following [19], there exist two main approaches to cope with uncertainties. The first one is to write a Chance-Constrained Program (CCP) where the constraints can be satisfied with identified probabilities due to stochasticity; a solution still has to optimize an objective function, but now, the probability that it fails is bounded by an additional parameter. The second is to actually take into account the probability of failing to satisfy a constraint during the plan execution, then being able to react through recourse: these are Stochastic Programs with Recourse (SPRs). In the last case, the recourse might be pre-planned and its cost integrated in the objective function, or alternately, the solution might be updated dynamically during plan execution, or the problem might be solved online [20]. In all cases, effort is required, as the problem becomes more complex: for instance, for the CCP, the simple calculation of the objective function of a given candidate solution may require the resolution of another optimization problem. Numerous approaches have been proposed; recently, machine learning has been used for solving the class of VRP problems (see [21]); for intractable settings, it is an example of how stochastic constraints or demands may be incorporated (see Appendix [C.6] in [21].

In the following, we list some examples of sources of uncertainties that may affect the problem solving of our UAV planning mission:

- general reliability of the drone with respect to the mission (mean time to failure),
- external incident (like physical attacks on drones, eagle hunting, etc.),
- travel time that can be subject to weather conditions,
- consumed energy that can be subject to weather conditions or rotor engine conditions,
- energy capacity, which can be affected by battery aging,
- service duration, which depends on the actual throughput to transfer data

- positioning precision, which depends on the precision of the embedded localization system

The uncertainties may have an impact both on the variables and on the methodology that would be used. For real deployments of the PDPTW-UAV, in general, arguably one of the most important issues is the ability to recover the UAVs; hence, we would suggest formulating an extended version problem with the family of stochastic programs with recourse so that it includes a plan to recover the drone. Our proposal would constitute a baseline and benchmark for such extended approaches.

Real-world context:

Another problem from a real-world scenario that is worth discussing is that of gathering information about the requests of nodes in a network that does not provide full connectivity, in order to compute UAV tours by our algorithm in a centralized manner.

A real-world example could be that of nodes that are capturing images, videos, or time series from sensors with a high sample rate, therefore accumulating large volumes of data. One would make requests for the UAV transport of the data. The requests are considered as small packets and could be collected through a Low-Power Wide-Area Network (LPWAN) with very wide coverage, such as Long Range (LoRa) technology. The LPWAN network would be insufficient to transmit all the raw data (range of several kilometers for a LoRaWAN gateway, but with only a few bytes/seconds available per node with LoRa in practice). Instead, based on the requests, the proposed NSGA-II algorithm will compute the drones' tour where the large volumes of data can be carried out by these flying vehicles. Another option is to collect requests during the UAVs' tour while they are serving sites. The collected requests will be used by the NSGA-II to calculate the next tours. Machine learning could also be used to predict and optimize the computation of the upcoming UAVs' tour.

7. Conclusions

In this paper, we applied the NSGA-II to solve the pickup and delivery optimization problem with a time window with an intermittent connectivity network using UAVs, called PDPTW-UAV.

One of the main differences from previous studies is that we were able to introduce UAV constraints (such as refueling) in our variant of the NSGA-II. We were able to add a refueling constraint verification algorithm to our variant of the NSGA-II, because it is modular. The refueling constraint verification can be modified arbitrarily, to compute the most realistic energy consumption model, etc., without modifying other parts of the algorithm.

For this purpose, we introduced a new representation of individuals (genes), as well as a number of associated heuristics and algorithms for generation, crossover, and mutation. Several experiments were conducted with and without the consideration of a refueling constraint. The results presented in this paper clearly show that NSAG-II is capable of achieving excellent results when applied to the PDPTW-UAV problem.

In our future work, we plan to optimize the management of the refueling operations, and we will also propose a formula to make the refueling time proportional to the amount of energy required by the UAV.

Author Contributions: Conceptualization, I.K. and A.L.; methodology, I.K. and A.L.; software, I.K. and M.H.; validation, I.K., A.L. and C.A.; writing—original draft preparation, I.K., A.L. and C.A.; writing—review and editing, I.K., A.L. and C.A.; funding acquisition, A.L. and C.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Labex DigiCosmeGrant Number ANR-11-LABEX-0045-DIGICOSME. This research was partially supported by Labex DigiCosme (Project ANR-11-LABEX-0045-DIGICOSME) operated by ANR as part of the program "Investissement d'Avenir" Idex Paris-Saclay (ANR-11-IDEX-0003-02).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hernandez-Lopez, D.; Felipe-Garcia, B.; Gonzalez-Aguilera, D.; Arias-Perez, B. An automatic approach to UAV flight planning and control for photogrammetric applications. *Photogramm. Eng. Remote Sens.* **2013**, *79*, 87–98. [CrossRef]
2. Lau, H.C.; Liang, Z. Pickup and delivery with time windows: algorithms and test case generation. In Proceedings of the 13th IEEE International Conference on Tools with Artificial Intelligence. ICTAI 2001, Dallas, TX, USA, 7–9 November 2001; pp. 333–340. [CrossRef]
3. Cao, W.; Yang, W. A Survey of Vehicle Routing Problem. *MATEC Web Conf.* **2017**, *100*, 1006. [CrossRef]
4. Parragh, S.N.; Doerner, K.F.; Hartl, R.F. A survey on pickup and delivery problems. *J. für Betriebswirtschaft* **2008**, *58*, 21–51. [CrossRef]
5. Lau, H.C.; Liang, Z. Pickup and delivery with time windows: Algorithms and test case generation. *Int. J. Artif. Intell. Tools* **2002**, *11*, 455–472. [CrossRef]
6. Ben Alaia, E.; Dridi, I.H.; Bouchriha, H.; Borne, P. Optimization of the multi-depot Multi-vehicle pickup and delivery problem with time windows using genetic algorithm. In Proceedings of the 2013 International Conference on Control, Decision and Information Technologies (CoDIT), Hammamet, Tunisia, 6–8 May 2013; pp. 343–348. [CrossRef]
7. Gansterer, M.; Hartl, R.F.; Salzmann, P.E.H. Exact solutions for the collaborative pickup and delivery problem. *Cent. Eur. J. Oper. Res.* **2018**, *26*, 357–371. [CrossRef] [PubMed]
8. Wang, J.; Zhou, Y.; Wang, Y.; Zhang, J.; Chen, C.L.P.; Zheng, Z. Multiobjective Vehicle Routing Problems With Simultaneous Delivery and Pickup and Time Windows: Formulation, Instances, and Algorithms. *IEEE Trans. Cybern.* **2016**, *46*, 582–594. [CrossRef] [PubMed]
9. Sabo, C.; Cohen, K. *SMART Heuristic for Pickup and Delivery Problem (PDP) with Cooperative UAVs*; InInfotech@ Aerospace: San Juan, PR, USA, 2011. [CrossRef]
10. Karak, A.; Abdelghany, K. The hybrid vehicle-drone routing problem for pick-up and delivery services. *Transp. Res. Part C Emerg. Technol.* **2019**, *102*, 427–449. [CrossRef]
11. Khoufi, I.; Laouiti, A.; Adjih, C. A Survey of Recent Extended Variants of the Traveling Salesman and Vehicle Routing Problems for Unmanned Aerial Vehicles. *Drones* **2019**, *3*, 66. [CrossRef]
12. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
13. Guerreiro, P.M.M.; Cardoso, P.J.S.; Fernandes, H.C.L. Applying NSGA-II to a Multiple Objective Dial a Ride Problem. In Proceedings of the Computational Science—ICCS 2019, Faro, Portugal, 12–14 June 2019; Rodrigues, J.M.F., Cardoso, P.J.S., Monteiro, J., Lam, R., Krzhizhanovskaya, V.V., Lees, M.H., Dongarra, J.J., Sloot, P.M., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 55–69.
14. Velasco, N.; Dejax, P.; Guéret, C.; Prins, C. A non-dominated sorting genetic algorithm for a bi-objective pick-up and delivery problem. *Eng. Optim.* **2012**, *44*, 305–325. [CrossRef]
15. Phan, D.H.; Suzuki, J. Evolutionary Multiobjective Optimization for the Pickup and Delivery Problem with Time Windows and Demands. *Mob. Netw. Appl.* **2016**, *21*, 175–190. [CrossRef]
16. Li, H.; Lim, A. A Metaheuristic for the Pickup and Delivery Problem with Time Windows. *Int. J. Artif. Intell. Tools* **2001**, *12*, 160–167. [CrossRef]
17. Li & Lim Benchmark. Available online: https://www.sintef.no/projectweb/top/pdptw/li-lim-benchmark/ (accessed on 25 December 2019).
18. Kora, P.; Yadlapalli, P. Crossover Operators in Genetic Algorithms: A Review. *Int. J. Comput. Appl.* **2017**, *162*, 34–36. [CrossRef]
19. Oyola, J.; Arntzen, H.; Woodruff, D.L. The stochastic vehicle routing problem, a literature review, part I: models. *EURO J. Transp. Logist.* **2018**, *7*, 193–221. [CrossRef]
20. Ritzinger, U.; Puchinger, J.; Hartl, R.F. A survey on dynamic and stochastic vehicle routing problems. *Int. J. Prod. Res.* **2016**, *54*, 215–231. [CrossRef]
21. Nazari, M.; Oroojlooy, A.; Snyder, L.; Takac, M. Reinforcement Learning for Solving the Vehicle Routing Problem. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 3–8 December 2018; Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R., Eds.; Curran Associates, Inc.: Montreal, QC, Canada, 2018; Volume 31.