Expert Systems With Applications 243 (2024) 122809

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A bi-criteria approach to the truck-multidrone routing problem

Pedro L. Gonzalez-R [a,*], David Sanchez-Wells [a,c], José L. Andrade-Pineda [b]

[a] Department of Industrial Engineering and Management Science, School of Engineering, University of Seville, Camino de los Descubrimientos, s/n, Seville 41092, Spain
[b] Robotics, Vision & Control Group, School of Engineering, University of Seville, Camino de los Descubrimientos, s/n, Seville 41092, Spain
[c] Indaero Emergy Group, C/Espaldillas Diez, 10, Seville 41500, Spain

ARTICLE INFO

Keywords:
Multi-drone-truck logistics
Multi-objective
Makespan
Truck mileage
Last-mile delivery

ABSTRACT

In the rapidly expanding field of e-commerce logistics, the optimisation of last-mile delivery solutions is paramount. This paper introduces a novel methodology that addresses this challenge by generating approximate Pareto fronts in a hybrid truck-drone delivery system. Specifically, we examine a generalized single truck multi-drone problem that allows multi-visit flight missions and rendezvous points distinct from launch locations. Our goal is providing decision-makers with a portfolio of optimal routing solutions that balance service time and environmental impact, criteria that are increasingly shaping decision-making in this domain. To achieve this, we introduce a bivector coding scheme inspired by flow-shop scheduling problems and implement a Simulated Annealing algorithm. This algorithm features an advanced stopping mechanism, negating the need for manual adjustments by utilizing a distinctive blend of a domination rate and a Kalman filter. Importantly, our framework employs an iterated greedy search algorithm to evolve from initial solutions towards identifying non-dominated solutions sets, which are then ranked using a hypervolume coefficient. To validate our methodology, we conduct a sensitivity analysis on two different size instances using a full factorial design of experiments. Our analysis reveals crucial insights into the impact of the number of drones, their autonomy, and their flight speed settings. From it, we conclude that it is a robust and adaptable framework for its practical application for obtaining Pareto fronts solutions among which picking the ultimate routing to be implemented.

1. Background

The explosion of e-commerce logistics has led to increasing demands for contactless delivery (Lin, 2020), thereby pushing small package couriers to incorporate autonomous transport technologies, such as unmanned aerial vehicles (UAVs) or drones. While the regulatory framework starts to allow the hybrid truck-drone delivery service that companies such as Amazon and UPS have been largely preparing (Shavarani et al., 2018), their investments in new resources are being carefully revised, with their focus more and more on the flexible implementation of routing plans adapted to their logistics decision making.

While last-mile delivery has been by far the most important research stream on hybrid truck-drone logistics problems, certain features need to be added for more realistic routing planning decisions. For instance, we need to consider an active role for ground vehicles, assuming that they can also deliver packages to customers – see, e.g. (Murray & Chu, 2015; Ha et al., 2018; de Freitas and Penna, 2020; Dell'Amico et al., 2022) – instead of treating trucks only as supporting vehicles that solely act as mobile depots – see, e.g. (Karak & Abdelghany, 2019) and (Ferrandez et al., 2016). Despite the use of a drone's capability launching from/coming back to a truck while it is moving from one customer location to another has been recently suggested (Li et al., 2022; Marinelli et al., 2018; Masone et al., 2022; Schermer et al., 2019), safety issues persuade us on the convenience of avoiding this policy. Instead, we consider the case in which the truck launches drones and moves to a different customer location taken as a rendezvous point as the more promising trend for transforming last-mile logistics into a brand-new joint delivery service. Known as the hybrid synchronised truck and drone delivery model (Moshref-Javadi et al., 2023), this modality serves to alleviate overcrowded airspace and reduce the chance of accidents (Rojas Viloria et al., 2021). Along with this safety requirement, the redesign of the truck multi-drone service we consider here is aimed not only at the efficiency of the system itself, but also at the reduction of pollution, level of emissions, carbon footprint, and the issues on fuel availability that are within the current agenda of many states (singularly, within the European Union).

The problem of pairing drones with traditional delivery trucks was

* Corresponding author.
E-mail addresses: pedroluis@us.es (P.L. Gonzalez-R), davsanwel@alum.us.es (D. Sanchez-Wells), jandrade@us.es (J.L. Andrade-Pineda).

https://doi.org/10.1016/j.eswa.2023.122809
Received 16 November 2022; Received in revised form 8 September 2023; Accepted 30 November 2023
Available online 6 December 2023

first devised by Murray and Chu (2015) as the Flying Sidekick Travelling Salesman Problem (FSTSP), which was extended by (Raj & Murray, 2020) to the use of an arbitrary number of heterogeneous UAVs that can be deployed from the depot or the delivery truck. Our research is also devoted to study the multi-drone case but while (Raj & Murray, 2020) is limited to drones that are serving one solely customer per fly, we assume a more general version of the hybrid truck-drone research stream: multi-visit flight missions, truck and drone allowed for serving each customer location, and rendezvous at different points than that of launches (Gonzalez-R et al., 2020).

Our approach exhibits certain similarities with recent work (Poikonen & Golden, 2020; Luo et al., 2021; Gu et al., 2022; Leon-Blanco et al., 2022). Poikonen & Golden (2020) presented an interesting Travelling Salesman Problem (TSP) with multiple drones with consideration of adjustable speeds and battery consumption rate as a function of the payload, although they assumed the truck serves solely as a mobile depot which does not deliver packages to customers. Luo et al. (2021) have addressed a problem like ours, although their computational experience is severely limited by the number of drones in the fleet (only two drones per truck). Gu et al. (2022) have recently applied a variable neighbourhood descendant procedure to address instances of up to 200 customers in a last-mile delivery. However, while our study concerns a multi-objective approach, these authors aimed at minimising a simple weighted combination of objectives comprised of fixed vehicle costs and cumulative duration costs of both truck routes and drone trips. Finally, Leon-Blanco et al. (2022) is the most connected work to the one we present here. Therein, the authors stated the Truck-Multi-Drone Team Logistics Problem (TmDTL) which is solved by using multi-agent systems for large problem instances, but with a single objective criterion: minimisation of makespan – that is, the time elapsed until the completion of the service.

In this research we cope with the TmDTL but, aside from minimising makespan, we are also minimising the truck usage as a cost driver of the economic impact due to its consumption of energy or/and manpower factors. Since we assume that the truck travels at a constant speed, the added cost driver can be viewed as a truck mileage minimisation criterion which fits to both, electric trucks (usage as a driver of the electrical battery life-cycle consumption) or trucks with combustion engines (usage as a driver of the level of emissions). Besides, as in other hybrid truck-drone literature (Moshref-Javadi et al., 2020; Luo et al., 2021), we assume a constant UAV to truck speed ratio when studying the design of the brand-new joint delivery service with our bi-objective approach. Another assumption made is that the decision maker (DM) needs from applying *a posteriori* multi-optimisation approach (Zajac & Huber, 2021), namely, their interest is in being given a Pareto optimal set from which they will ultimately choose the final solution to implement.

In short, this paper covers DM's decision on how to best exploit a truck along with a variety of drones by choosing the number of drones and their operational settings while considering the current vectors guiding the sector: the quality of the service provided but also the environmental impact of implementing the routing. To this aim, we have devised a novel methodology capable of providing trade-off solutions in a reasonable time. This is reached by applying a combination of innovative heuristic solution evolution methods to estimate Pareto-efficient solution sets by using an iterated greedy heuristic globally guided by simulated annealing.

The remainder of the paper is as follows. Section 2 is devoted to shortly presenting our problem description and then proceeding to detail a novel vector-based coding of the routing solutions which poses our TmDTL problem as a permutation flow-shop scheduling problem. Section 3 revises related work, both on the drone routing and in multi-objective optimisation (MOO) contexts. Section 4 presents our approach to the methodology. As the computational experience presented in Section 5 shows, this method dramatically reduces the cost in time and resources for estimating Pareto fronts for the challenging bi-objective TmDTL instances. Section 6 discusses the results of our experiments along with two post hoc analyses on the effect that the parameter setting has on the Pareto fronts. Finally, Section 7 is devoted to drawing conclusions and further research lines.

2. Problem description and solution coding

The problem at hand is to determine the best set of routes to serve several customers for both total service time and truck usage time in a logistics system, where a fleet of drones with limited autonomy works collaboratively with a truck. In our approach, the collaborative work between the drone fleet and the truck is defined by the fact that both can serve customers (nodes) equally, so all nodes could be served by either type of vehicle, in addition to the fact that both the truck and drones are considered to start the service at a start node and conclude the service at an end node. Therefore, the difference between the two types of vehicles lies in two factors: travel autonomy and displacement speed.

As far as autonomy is concerned, commercial drones, for the most part, and in the current state of the art, obtain the energy necessary to perform their functions from the electrical energy contained in batteries, which they therefore necessarily charge. The amount of energy these batteries can store defines the autonomy of the drone that equips them, so this autonomy is limited. As a relatively novel idea, this is solved by using the truck as a mobile battery exchange station, since the autonomy of the truck is considered infinite since it is much greater than the autonomy of the drones.

To make these battery exchanges possible, in which any number of them can occur, we adopt the realistic assumption that they must necessarily happen at some node, where the drone and the truck meet. This will imply that at the nodes where the encounter occurs, the vehicle that arrives first will be forced to wait for the other. Any nodes where the drone-truck meeting occurs are called "synchronisation nodes". After the drone and the truck meet at a certain synchronisation node, the autonomy of the drone is fully recovered. Fig. 1 shows graphically an example of the problem introduced for the case of two drones. Customers are visited by the truck or by one of the two drones, with synchronisation occurring at nodes 5 and 8 for drones 1 and 2, respectively. After every synchronisation, the drones can choose to stay parked on the truck (travelling with no autonomy consumption until the next node to be visited by the truck) or to take off to keep performing deliveries.

Once the drones parked on the truck arrive at the next node, they can take off to start a new service route or stay parked on the truck for any number of trips. In this context, it may also happen that part of the drone fleet on the truck is never used, so a correct sizing of the fleet is necessary.

In this context, assuming that the speed of the drone can be considered superior to that of the truck, we follow (Gonzalez-R et al., 2020) and (Moshref-Javadi et al., 2020) and consider that the speed of the drones is higher than the speed of the truck. Furthermore, our assumption of drones' constant speed neglects operational features that are beyond the scope of our study, e.g., drone energy consumption in landing and take-off manoeuvres.

As a result, to sum up, we can establish the following list of rules, which we call **visit assumptions**:

1) All vehicles meet at the first and last nodes of the list, as they are assumed to be the necessary beginning and end nodes for the complete service.
2) Any vehicle can serve any node and all nodes must be served.
3) If a node is served solely by a drone, no other drone can serve it as well.
4) If a node is served by a truck, any number of drones can meet the truck at that node and fully recover its autonomy. From this point on, there are two possibilities for any drone:
   a) Take off again to continue to serve the nodes.
   b) Stay landing on the truck without consuming any autonomy until the truck reaches its next node.

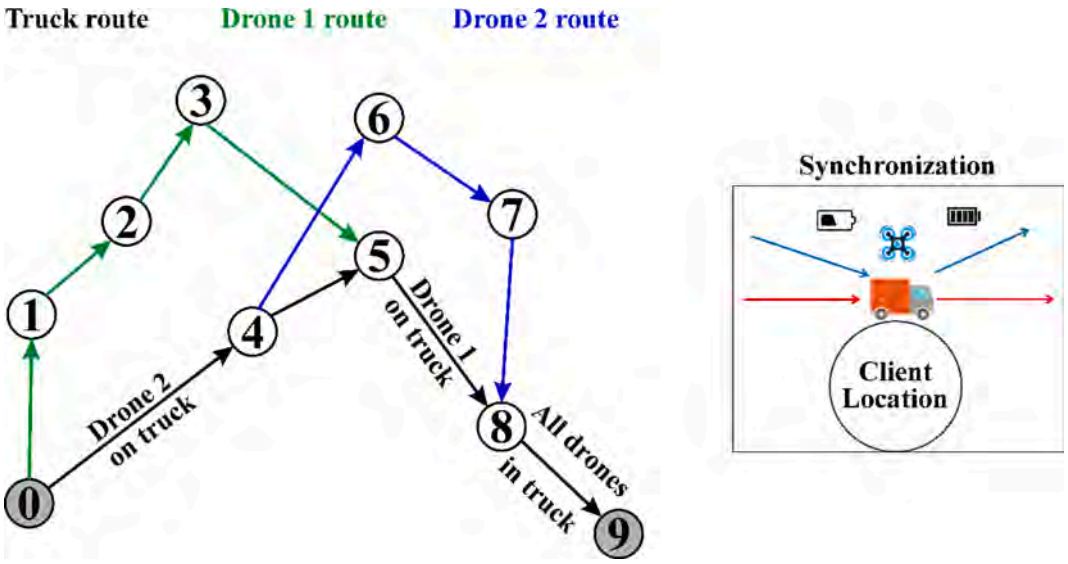

Fig. 1. Collaborative logistics example of a truck and two drones and synchronisation diagram.

5) The battery exchange time is negligible compared to the mission time, so it has been neglected.
6) Meeting at a node necessarily involves a waiting time, as all vehicles will have to wait until the last minute to reach that node. These waiting times do not consume any autonomy for any drone, nor indeed for the truck. Therefore, hovering is not considered.
7) Autonomy of the truck is assumed to be not limited, namely, truck technology allows the completion of all potential routes within the set of locations, no matter its length.

The above assumptions make TmDTL a very combinatorial problem whose resolution easily leads to a great computational burden. To mitigate the latter, we need to define novel solution coding as a key part of the process of obtaining efficient solutions. In the following, we present a novel codification scheme capable of reducing the combinatorial space of solution search. Specifically, the codification scheme is extensively used in an IG heuristic, similar to what was done by Gonzalez-R et al. (2020) but generalised to a multi-drone context for providing trade-off solutions in a reasonable time.

2.1. Solution coding

First, we need to define the coding of the solution as a key part of the process of obtaining efficient solutions.

Each solution is defined by a combination of two vectors: nodes vector (π_n) and modes vector (π_m). The indexes of both vectors are a key characteristic, as they are used to point within both vectors sequentially to define a specific solution. π_n or *nodes vector* is an integer list where every integer is a node number, and the list index defines the evaluation order. An example of a nodes vector for five nodes would be $\pi_n = [0, 3, 1, 2, 4]$. On the other hand, π_m or *modes vector* is an integer list where each integer value represents a visit mode number, and where the list index defines the evaluation order as well. An example of a *modes vector* for five nodes would be $\pi_m = [7, 2, 4, 3, 7]$. The *modes vector* codification is one of the value proposals of our method. Every visit mode defines how its index-corresponding node in the *nodes vector* is visited.

First, to define the visit modes, we need to define how synchronisation works and how feasible solutions are constructed. These visiting modes are defined below.

For a defined number of drones k, we create a binary matrix of visit mode, where the number of columns will be $k+1$ (for the truck), which is equal to the number of total vehicles, and where the number of rows is equal to the sum of as many different binary combinations as possible for the number of total vehicles (2^{k+1}). In this binary matrix, 1 means that a specific vehicle visits a node and 0 means that it does not. Taking this into account and according to the problem description, we must not use some of the visit mode rows, as they do not obey our visit assumptions. For the case of two drones, we must delete row 1, as it would be a no-visit and all nodes must be visited (visit assumption 2), and row 7, as it would imply that two different drones serve the same node without a truck (visit assumption 3). These cases (rows 1 and 7 of Table 1) are highlighted in bold.

To obtain the vector of coded modes, it is necessary to merge the values of drones and truck columns for every row to get a binary value (binary merge) and then convert every binary visit mode into a decimal value (decimal codification), as shown in Table 1. Finally, all these values build a list called the visit modes options vector. In this example, this vector is $[1, 2, 3, 4, 5, 7]$. Note that every odd value, apart from 1 defines a meeting mode, where any number of drones meet the truck at a certain node.

Now that the vector of options for visit modes has been created for a certain number of drones, we can say that any feasible vector of *modes* must start and finish with the maximum value of the vector of modes (7 in this example), as it indicates that all vehicles meet at its index-corresponding node of the vector node, and this is mandatory for the first and last values of this vector (visit assumption 1). Additionally, any possible combination of the decimal values contained in the visit modes options vector defines a different but not necessarily feasible solution, as it may not fulfil the autonomy constraints. For example, see in Fig. 2 the corresponding solution that nodes vector $\pi_n = [0, 3, 1, 2, 4]$ and modes vector $\pi_m = [7, 2, 4, 3, 7]$ are coding in a case with two drones and five nodes. In this example, all vehicles meet at the starting node, 0 (7 in binary: 111), node 3 is visited by drone 1 (2 in binary: 010), node 1 is

Table 1
Visit modes matrix. Example for two drones and five nodes.

Row Num.	Drone 2	Drone 1	Truck	Binary merge	Decimal cod.
1	**0**	**0**	**0**	**000**	**0**
2	0	0	1	001	1
3	0	1	0	010	2
4	0	1	1	011	3
5	1	0	0	100	4
6	1	0	1	101	5
7	**1**	**1**	**0**	**110**	**6**
8	1	1	1	111	7

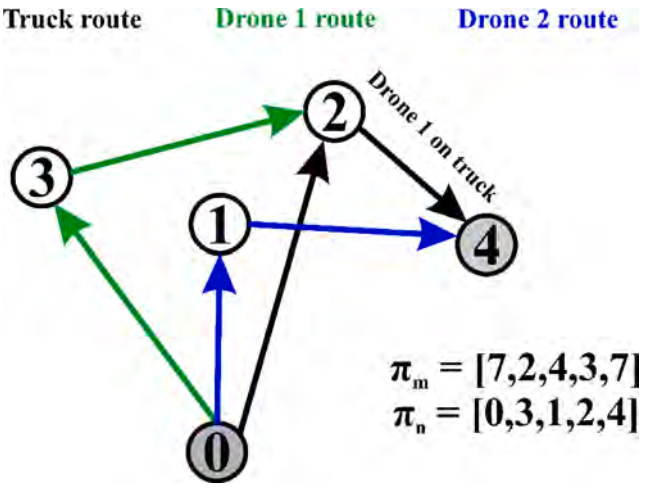

Fig. 2. Example solution scheme (two drones, five nodes).

visited by drone 2 (4 in binary: 100), node 2 is visited by drone 1 and truck (3 in binary: 011) and all vehicles meet at the end node, 4 (7 in binary: 111).

In this type of problem, a binary matrix has been commonly used to define when a given vehicle visited a given node. This, which from a mathematical approach is not a major problem, is a slowing factor when carrying out spatial searches based on heuristics since each possibility requires the handling of a list of a length of, in our case, the number of drones +1, which exponentially increases the time required to handle each list as the drone number increases. Therefore, this efficient coding scheme is suitable for handling such a problem whose complexity increases with the size of the instances. Taking into account the elimination of unfeasible modes, the number of modes that will arise from a given combination of k drones is $2^k + k$, so with one drone $2^1 + 1 = 3$ options would be handled, with two drones $2^2 + 2 = 6$ options would be handled, with three drones $2^3 + 3 = 11$ options would be handled, and so on. In these same cases and with a binary matrix, the first case would imply three lists of two bits (six elements vs three), in the second case six lists of three bits (18 elements vs six) and in the third case 11 lists of four bits (44 elements vs 11). The search complexity reduction is, therefore, of $k+1$ order, and so is the computational time reduction as well.

The presented coding scheme is highly beneficial for generating a set of initial solutions that constitute the initial Pareto front, which is further evolved for the best estimate set until a no progress state is identified.

3. Related work

3.1. Multi-objective optimisation in drone routing problems

In the drone routing context, multi-objective optimisation models are mostly aimed at minimising a function that aggregates the given objectives to identify a single preferred solution. Such approaches are mostly based on objectives that are in conflict and compete among them. Guerriero et al. (2014) tackle the deployment of drones to film a live sports event throughout a MILP model minimising the number of drones and the distance travelled along with the commitment to soft time windows whereas di Puglia Pugliese & Guerriero (2017) address a multi-truck multi-drone problem aimed at the minimisation of completion time, also considering the existence of differentiated time windows at each customer. Coelho et al. (2017) propose an evolutionary approach based on using weighted sum functions in fitness allocation when tackling the problem of a heterogeneous fleet of drones with capacity and autonomy constraints that, supported by static charging stations, are routed to pick up packages from one node and deliver them to another. Indeed, they generate multiple solutions: first solving the stated MILP including the charging station constraints and then solving the subsequent MOO employing a metaheuristic termed Multi-Objective Smart Pool Search (Coelho et al., 2016).

Unlike the above works, one could assume that the decision maker needs to be given a Pareto optimal set from which it selects one final solution. Wang et al. (2020) apply such a method when proposing a non-dominated sorting genetic algorithm to solve a bi-objective FSTSP which minimises completion time and operational costs, although referring to a single-truck-single-drone setting which poses a lesser combinatorial burden than the TmDTL, more even owing to their assumption on single-customer per flight delivery. Tavana et al. (2017) apply it when approaching a cross-docking setting where some small items are delivered by drones from a supplier directly to a customer, while other items are delivered by trucks through the cross-docking system that is located between a supplier and a customer: since drone deliveries are faster, they seek a trade-off of drones scheduling costs and the time savings it provides. Omagari & Higashino (2018) study a bi-objective drone delivery problem with travelling distance and delivery time minimisation, to decide at which node the drone is launching, which nodes it visits, and at which rendezvous node it reconnects with the truck. The authors do not estimate the Pareto front but use the provisional ideal point method, where a point where the search would lead is first estimated. This makes the calculation of the entire Pareto front unnecessary. When compared to our TmDTL problem, the following differences arise: their drone unary load capacity leads to serving one customer per drone flight, whereas we need to consider a multi-drop approach; the rendezvous points are predefined, while in our problem it can be any service node; and the truck must have the drone when the truck returns to the depot, while in our case there is no depot, but equally the truck can complete its route without waiting for the drones. Ramirez-Atencia et al. (2017) address a mission planning problem involving a team of UAVs and a set of Ground Control Stations (GCS), modelled as a Constraint Satisfaction Problem (CSP), which they solved employing a multi-objective genetic algorithm, therein using a performance indicator pairing distance and makespan. Finally, Ramirez Atencia et al. (2019) addressed the same problem with a MOO for which an estimation of the Pareto optimal frontier is inferred to get a portfolio of solutions (mission plans) differently, albeit optimally balancing the considered conflicting objectives: the fuel consumption, the makespan, the cost of the mission, or the number of UAV or GCS to employ and different risk factors that could compromise the mission. Like them, in our research work, we are concerned with the provision of a decision-making tool for a real-life application in which it is necessary to find a good solution and not the complete space of possible solutions. To this aim, we determine multiple Pareto optimal solutions by concurrently optimising the multiple objectives to obtain a set of non-dominated solutions according to a certain performance indicator.

3.2. The performance indicator guiding the shortlist of Pareto solutions

Regarding MOO performance indicators, (Audet et al., 2021) have analysed a total of 63 performance indicators partitioned into four groups according to their properties (cardinality, convergence, spread and distribution) and also present applications for them. Noticeably, the authors state that *"The hypervolume indicator and some closely related metrics are the only known unary indicators to be strictly monotonic"*. In particular, the dominated hypervolume (or S-metric) is a commonly accepted quality measure for comparing approximations of Pareto fronts generated by multi-objective optimisers (Beume, 2009). Given a set of points, the S-metric figure is calculated as a sum of exclusive hypervolumes (see Fig. 3), and each exclusive hypervolume is calculated by limiting the underlying set with a contributing point and subtracting the hypervolume of the modified set from the inclusive hypervolume of the contributing point (While et al., 2012; Demir et al., 2019).

According to (Petchrompo et al., 2022), the hypervolume indicator

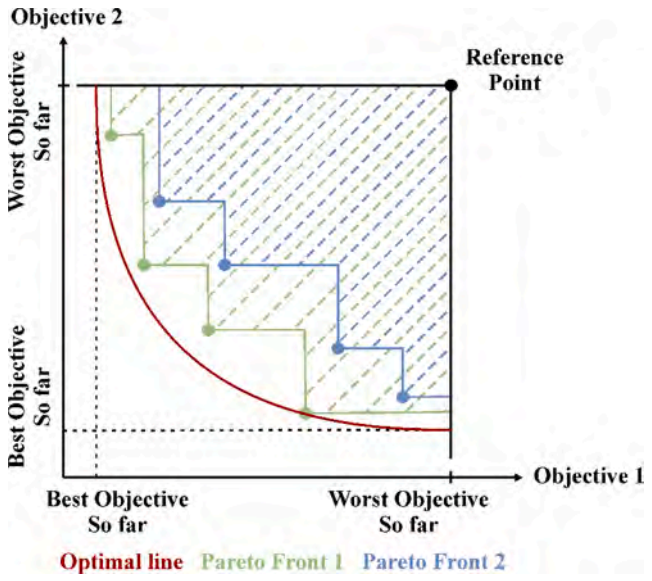

Fig. 3. Hypervolume indicator (Adapted from Demir et al., 2019).

can measure both the convergence and diversity of Pareto's optimal solution, with the rationale that a solution set is the most diverse when the space covered by its members and a selected reference point is maximised. Despite the merits of the hypervolume indicator being well recognised, putting it in place necessarily implies embedding it within an optimisation algorithm, generating well-distributed sets of solutions efficiently.

Noticeably, the search procedures to find solutions that yield better trade-offs are computational efforts whose stops need to be instructed. A study of the algorithm search stopping criteria used in drone-assisted routing problems heuristic-based approaches shows that the most common criteria are threefold: using a stopping criterion that considers that a certain number of iterations have been performed without finding any improvement in the solution; using a maximum number of iterations without considering any other factor, as in (Yan et al., 2021) or (Almuhaideb et al., 2021); or using a maximum computation time without considering any other factor. There are also algorithms in which two criteria are used simultaneously for stopping, such as Zhou, Zhang, Shi, Liu, and Huang (2018), which jointly employs a maximum number of iterations without improvement and a maximum number of iterations, and (Luo et al., 2021), where a maximum computation time and a maximum number of iterations without improvement are jointly employed with the application of a coefficient.

The effect of different stopping criteria on MOO is an open research stream, but, according to Abu Doush et al. (2021), the use of the Kalman filter applied on an indicator to determine a 'no progress state' is still competitive against other more commonly used stopping criteria for multi-objective evolutionary algorithms, such as running average or online convergence detection applied on hypervolume. Let us briefly explain that the Kalman filter is an efficient recursive filter that is typically exploited for estimating the internal state of a linear dynamical system from a series of noisy measurements. In our context, its usefulness is in gaining evidence that a variable (chosen as an indicator) is stabilised around a reference, thereby indicating that the computational effort of inspecting that variable over time should be saved.

3.3. Concluding remarks

Once clarified that we are not interested in estimating the entire Pareto optimal frontier – what is considered neither meaningful nor computationally practical (Dasdemir et al., 2020) – but in drawing a shortlist of Pareto optimal solutions, this subsection outlines the novel methodology we propose for addressing the multi-objective TmDTL problem since the proposed methodology can better adapt to the proposed problem than existing methods.

The novelty is threefold. First, we create a solution coding that presents the TmDTL as a permutation flow-shop problem, which is highly efficient for the iterative generation of solutions. Second, taking advantage of this better formulation, the resolution is adapted to the Multi-Objective Iterated Greedy Search (MOIGS) algorithm (Framinan & Leisten, 2008). Third, we present a novel integrated environment in which the makespan (the total time to complete the task of serving all existing nodes) and the total truck time (proportionally related to CO2 emissions and labour cost) are integrated into the hypervolume indicator (*chi*, Demir et al., 2019), which is used as a guide for efficient frontier construction, along with a Kalman filter-based stopping criterion (Martí et al., 2016).

4. The BOA-KS methodology

The BOA-KS methodology comprises the following steps:

- Step 1. Initialisation. In this step, the global parameters of the system and the instance data are read.
- Step 2. Generation of the initial solutions set (Bunch-SG algorithm). This step generates a spread of the Pareto front using in its process the LocalSIS algorithm for each objective.
- Step 3. Estimation of the Pareto frontier (BO-LocalS algorithm), for which using the concept of the MOIGS method used in a flow-shop problem (Framinan & Leisten, 2008), has been transformed in a novel way to solve the proposed routing problem.

According to the notation presented in Table 2, we provide a track of the key input variables involved in each of the components of BOA-KS. The details of each of the steps are shown below in Table 3.

4.1. The LocalSIS algorithm to generate initial solutions

The LocalSIS algorithm explores the best way to obtain ranked solutions using a generic objective function, consisting of the performance measure Z and penalisation in the case of solution infeasibility. Although the calculation method is explained later, the basic procedure consists of

Table 2
Variables and their use as input in the algorithms.

Notation	Variable (V)/ Parameter (P)	Description	Algorithms where the variable is an input		
			LocalSIS	Bunch-SG	BO-LocalS
π_n	V	Nodes vector	X		
π_m	V	Modes vector	X		
k	P	Number of drones	X	X	X
V	P	Visit modes	X		X
T_i	P	Initial temperature	X		
T_e	P	End temperature	X		
ΔT	P	Temperature gradient	X		
N_{bun}	P	Number of initial solutions for every objective function		X	
PS	P	Pareto scale		X	X
Π_k	P	Set of non-dominated initial solutions			X
SGA_{min}	P	Stopping criteria indicator limit			X
d	P	Number of positions to be stolen			X

Table 3
Phase 1 example for $i = 1$ and $j = 3$

	Input solution	Output solution
Nodes vector	$\pi_n = [0,3,1,2,5,4]$	$\pi_n^{'} = [0,2,1,3,5,4]$
Modes vector	$\pi_m = [7,2,4,3,1,7]$	$\pi_m^{'} = [7,3,4,2,1,7]$

objectifying the degree of non-compliance with the restrictions P affected by a penalty factor α. Therefore, OF can be calculated as $OF = Z + \alpha \cdot P$.

Specifically, the threat of LocalSIS consists of three steps: *(i)* '*destruction and construction*', *(ii)* 'local *search*', and *(iii)* '*decision*'. The latter may result in two-fold: either a better solution is found, or a non-improved solution is obtained. In both cases, we get into another LocalSIS loop, either with a better solution or with a 'mutation' performed on the best of the search loop solutions found so far.

- Phase 1: Solution Destruction and Construction Phase (SDCP). The SDCP is an initialisation based on the destruction and construction of the input solution. Two random intermediate positions i and j from the input solution are selected as initial and final indexes to consider sub-arrays in both the nodes and modes vectors and reverse them. There is a limiting rule on the index's selection: neither the initial nor the final position of any vector can be selected.
- Phase 2: Local Search Phase (LSP). The second step starts with a random swap among two positions, p and q, in the nodes vector (again, the first and last positions are not eligible for the swap). Then, a solution generation loop is executed, using the recently swapped nodes vector in combination with every possible mode from the visit modes options vector in the position of the modes vector. By using a memory variable, we guarantee that the same swap is not performed twice for the same input, saving computation time.
- Phase 3: Decision Phase (DP). The third step chooses the next solution to be evaluated as a tentative initial point to relaunch the local search. If an improved solution arises from the second step, this would be the re-entry solution; otherwise, the local search step was unfruitful, and we decide that the re-entry solution is a 'mutation' performed on the best solution found so far. These 'mutated' variables are defined with superscript *mt*. Randomly, this mutation can be accepted and sent as an input solution for the LSP. If the mutation is not able to improve the initial solution, the initial solution is considered again.

The algorithm is guided by a standard stochastic simulated annealing (SA) algorithm, starting at temperature T_i. The solution construction and destruction phases are performed by the *SDCP*() function. The function *calculate_OF*() takes as input the vectors of nodes π_n and modes π_m and returns a tuple formed by the value of both the objective function and the value of the penalisation P for non-compliance with the constraints. OF_l and P_l store each evaluated objective function and penalisation respectively within the same global iteration.

Algorithm 1 LocalSIS Pseudocode.

Input: $\pi_n, \pi_m, k, V, T_i, T_e, \Delta T$
Output: Improved (π_n, π_m)
calculate_OF$(\pi_n, \pi_m) \rightarrow (O, P)$*# Returns the objective function and penalisation*
$(\pi_n, \pi_m) \rightarrow (\pi_n^o, \pi_m^o)$*# The original input solution is saved apart*
while *True* **do**:*# No solution without penalisation has been generated*
  $(\pi_n^o, \pi_m^o) \rightarrow (\pi_n, \pi_m)$
  $T_i \rightarrow T$
  $O \rightarrow OF^{best}$
  $O \rightarrow OF'$
  $M =$ ***false****# Auxiliary variable: if false, actual solution is not a mutation*
  while $T > T_e$ **or** M **do**:*# End temperature has not been reached or the actual solution is a mutation*
    if *notM*:*# The actual solution is not a mutation*
      SDCP$(\pi_n, \pi_m) \rightarrow (\pi_n^{'}, \pi_m^{'})$ *# Solution construction and destruction phase (Phase 1)*

(*continued on next column*)

(*continued*)

    end if
    while *True* **do**: *# Local Search Phase (Phase 2)*
      node_random_swap$(\pi_n^{'}) \rightarrow (\pi_n^{''}, p)$
      random_mode_permutation$(\pi_m^{'}) \rightarrow (\pi_m^{''})$
      $OF_l =$ **{}**# OFs list creation or reset
      $P_l =$ **{}**# Penalisations list creation or reset
      for m **in** V: *# Mode intensification search*
        $m \rightarrow \pi_m^{''}[p]$
        calculate_OF$(\pi_n^{''}, \pi_m^{''}) \; += (OF_l, P_l)$ *# Add to lists*
      end for
      min_LSP$(OF_l) \rightarrow (OF^{min}, \pi_n^{''}, \pi_m^{''})$
      if $OF_{min} < OF^{i}$:
        $(\pi_n^{''}, \pi_m^{''}, OF^{min}) \rightarrow (\pi_n^{'}, \pi_m^{'}, OF^{i})$
      else:
        $(\pi_n^{''}, \pi_m^{''}, OF^{min}) \rightarrow (\pi_n^{mt}, \pi_m^{mt}, OF^{mt})$
        break
      end if
    end while
    if $OF' < OF^{best}$: *# Decision Phase (Phase 3)*
      $(\pi_n^{'}, \pi_m^{'}, OF') \rightarrow (\pi_n, \pi_m, OF^{best})$
      $M =$ ***false***
    else if ***Mistrue***:*# The SA has been executed using a mutation and no improvement has been found*
      $(\pi_n, \pi_m) \rightarrow (\pi_n^{'}, \pi_m^{'})$*# Select again as input solution that which generated the mutation*
      $M =$ ***false***
    else if $e^{\frac{OF^{mt} - OF}{T}} > \textbf{\textit{random}}(0,1)$:
      $(\pi_n^{mt}, \pi_m^{mt}, OF^{mt}) \rightarrow (\pi_n^{'}, \pi_m^{'}, OF^{i})$
      $M =$ ***true***
    end if
    $T = T - \Delta T$
  end while
  calculate_OF$(\pi_n, \pi_m) \rightarrow (OF, P)$
  if $P = \{\varnothing\}$:
    break
  end if
end while
return (π_n, π_m)

Finally, although the generation of a bunch of solutions will require a variety of executions of the LocalSIS routine, it is worth noting that running the LocalSIS aimed solely at a single criterion is not the better strategy for the Pareto front estimation for the bi-criteria TmDTL. Despite the multiple feasible and non-feasible solutions that are found when executing the LocalSIS procedure, only one feasible solution remains: the best that could be obtained with the established objective function and parameters. Thus, using a single criterion would be a wrong strategy that would heavily reduce the Pareto front spread and would result in a sort of solutions that easily could dominate each other, which would invalidate the dominated solutions from being part of the Pareto front, thereby reducing the number of solutions contained in the initial Pareto front as well. In short, aside from wasting time on computing and not improving solutions, the arising front would be of poor quality.

4.2. The Bunch-SG algorithm for creating a set of initial solutions

In what follows, we detail the upper control designed to invoke the LocalSIS in a manner that increases the quality of the future estimation of the Pareto front of our bi-objective TmDTL problem launching the search for a bunch of solutions at both extremes of the front. To this aim, we use a weighted sum objective function out of these two:

$$OF_1 = T + \beta_1 \cdot e$$

$$OF_2 = T_t + \beta_2 \cdot e$$

where:

OF_1 First objective function.

OF_2 Second objective function.
T Total completion time.
T_t Total truck time.
β_1 First penalisation factor for excess time for drone autonomy.
β_2 Second penalisation factor for excess time for drone autonomy.
e Total autonomy time excess.

Note that since we are executing the Bunch-SG algorithm either using OF_1 or OF_2, we only use a single variable "excess" and try to obtain feasibility by penalising its contribution. The correct values of β_1 and β_2 must be considered so that they penalise sufficiently and exercise the admissibility search function of the solutions required in each objective function. In this regard, we must remark that since both objective functions are of the same order, we set both at the same value $\beta = \beta_1 = \beta_2$.

Bunch-SG – see pseudocode in Algorithm 2 – has been designed to spread the Pareto front and create multiple solutions that are not dominated by each other. The number of drones k, the number of initial solutions n, and the Pareto scale PS are set to start the procedure, which subsequently updates the dominances on a list called Π_k. In an inner loop, a second list, Π_s, is devoted to storing temporal solutions that arise for each number of drones. The specific strategy to spread the Pareto fronts is as follows.

- First, OF_1 is chosen as the performance criterion for executing LocalSIS for n times. Along it, every new solution is stored in Π_s. Second, OF_2 is selected and LocalSIS is run n times, again storing any new solution in Π_s.
- Once both objective functions have been explored, the resulting solutions set Π_s dominance is checked, and the dominated solutions are deleted.
- The Π_s set of dominant solutions for the current number of drones k is added to the set of the dominant solutions, Π_k.

This is repeated until k is reached, when a check on the number of solutions contained in Π_k is made. The above is repeated till k is reached. Then, there is a check on the number of solutions contained in Π_k. If any of the solutions sets' length is bigger than PS, a user-defined scaling is applied so that an uniformly sized set of only PS elements is generated for every number of drones.

Algorithm 2 Bunch-SG Pseudocode.

Input: k, N_{bun}, PS
Output: Π_k set of non-dominated initial solutions
$\Pi_k = \{\}$
for 1 to k:
  $\Pi_s = \{\}$
  OF_1 is selected as the performance criterion
  for 1 to N_{bun}:
    A random solution (π_n, π_m) is generated
    LocalSIS is run for the current objective function
    if current solution not in Π_s:
      current solution $\rightarrow \Pi_s$
    end if
  end for
  OF_2 is selected as the performance criterion
  for 1 to N_{bun}:
    A random solution (π_n, π_m) is generated
    LocalSIS is run for the current objective function
    if current solution **not in** Π_s:
      current solution $\rightarrow \Pi_s$
    end if
  end for
  Π_k dominance is checked and corrected for k
  Π_k sets are scaled according to PS
end for
return Π_k

The set of the dominant solutions Π_k, which contains a different set of dominant solutions for every number of drones considered, is now the input for the last part of this methodology, which aims to generate an estimate of the Pareto front as explained below.

4.3. The BO-LocalS algorithm for Pareto front estimation

In (Gonzalez-R et al., 2020), the coded TDTL problem is identified as similar to typical permutation problems, so the authors propose an adaptation of an iterative greedy search algorithm (IG) that was used efficiently to minimise the completion time of tasks in permutation problems. As our codification to the TmDTL problem is based on the TDTL bivector codification but from a multi-drone and bi-objective approach, we can consider it comparable to a bi-objective permutation flow-shop scheduling problem, too.

Framinan & Leisten (2008) solved multi-objective permutation flow-shop problems employing the MOIGS algorithm, concluding that it was highly efficient and outperformed the best heuristics so far. As the MOIGS algorithm has not been beaten in its field, we consider it a perfect basis for developing a specific algorithm for our problem. However, since the MOIGS algorithm was formulated for a pure permutation flow-shop problem, a deep adaptation to the coding scheme of our targeted TmDTL problem has been attained, giving rise to a novel algorithm that we call BO-LocalS (Bi-Objective Local Search). The idea behind this can be outlined as follows:

- Bunch-SG is launched, creating a set of non-dominated solutions Π_k for every k number of drones. Each one of these solutions $(\pi_k^1, \pi_k^2, \cdots)$ is subsequently sent to the next step.
- A d integer number of randomly selected nodes is removed from the vector of nodes of the current solution and stored in vector D, one at a time, using the function *select_random_nodes*. Their index-correspondent modes are also removed from the modes vector, too.
- The remaining nodes and modes combinations are kept in the solution: this solution is now *partial,* as some of its nodes are not included. The *create_partial_solutions*() function generates a set of partial solutions, which are stored in Π_r.
- Each stolen node in D is then taken and inserted in all possible slots of its correspondent partial solution in Π_r using the *create_solution*() function. In every insertion, all possible visit modes are assigned to the index-correspondent position in the modes vector and used to generate new complete solutions, which are provisionally appended to the set denoted by *NWS* after being evaluated by using the *calculate_OF*() function.
- When the whole algorithm iteration ends for the current original solution, the provisional solution dominance is checked using the *evaluate_dominance*() function. Then, all non-dominant solutions in *NWS* are removed. The remaining solutions are set to the non-dominated set NDS_k and *NWS* is emptied. Then, the iteration begins again with the next solution.
- This iteration continues until all solutions are processed for every number of drones in Π_k.
- Every time iteration of a k number of drones iteration ends, the dominance of NDS_k is checked again, as the generated solutions by different partial solutions may have overlapped. When all non-dominant solutions are removed from NDS_k, the remaining set of solutions is scaled according to the defined PS.
- Pareto scale length using the *scale_solutions*() function. Then, the iteration for the next number of drones begins.

Algorithm 3 BO-LocalS Pseudocode.

Input: $\Pi_k, k, V, PS, SGA_{min}, d$
Output: NDS_k Pareto front solutions
$NDS_k = \{\}$
while $SGA_{min} < SGA_{value}$ **do**: *# Check stopping criteria according SGA*
  for 1 to k: *# For every number of drones*
    for every *complete_solution* in Π_k:
    $NWS = \{\}$
      select_random_nodes(*complete_solution*, d)$\rightarrow D$
      create_partial_solutions(*complete_solution*, D)$\rightarrow \Pi_r$
      for every *partial_solution* in Π_r:

(*continued on next page*)

(*continued*)

```
        for every slot in partial_solution:
            for every mode in V:
                create_solution(mode, slot, partial_solution)→current_solution
                calculate_OF(current_solution)→OF
                NWS += (OF, current_solution)
            end for
        end for
    end for
    dominated_solutions = check_dominance(NWS)
    NWS = NWS.delete(dominated_solutions)
    NDS_k += (NWS)
    end for
end for
dominated_solutions = check_dominance(NDS_k)
NDS_k = NDS_k.delete(dominated_solutions)
scaled_solutions = scale_solutions(NDS_k, PS)
SGA_value = SGA(NDS_k)
end while
return NDS_k
```

Notice that the above tasks are iteratively executed to the point at which we consciously decide that it does not deserve further computational efforts (see Algorithm 3). We have designed a stopping criterion mechanism, computed by the function $SGA()$, from Sánchez, Gonzalez, Andrade, which is based on a domination rate combined with a Kalman filter in a way similar to (Martí et al., 2016). Specifically, let *chi* be the coefficient of hypervolume indicator in use, which we set to the percentage evolution of the hypervolume given a contributing point p set at $(0, 0)$. The percentage of improvement in this indicator is computed at function $SGA()$ to give rise to a value which is compared to a selected threshold value SGA_{min}. In this study, we consider $SGA_{min} = 1\%$ to infer that the *chi* indicator has reached the desired stability.

We refer the reader to Fig. 4 for an example of a BO-LocalS execution. Noticeably, while the *chi* indicator is highly unstable in the first iterations of the BO-LocalS algorithm, it tends to stabilise after a certain number of iterations. Finally, the *chi* improvement is confused with noise and is assumed to be in a nonevolving state.

Importantly, the described mechanism establishes a reliable and adaptable stopping criterion for the BO-LocalS algorithm without the need for manual adjustment. Therefore, the BO-LocalS can remain to refine the Pareto front estimation with subsequent good solutions that are non-dominated.

4.4. BOA-KS limitations

As a tailored algorithm, the main limitations of BOA-KS are related to the TmDTL problem description as well as the usual limitations of MOO frameworks. These can be summarised as follows:

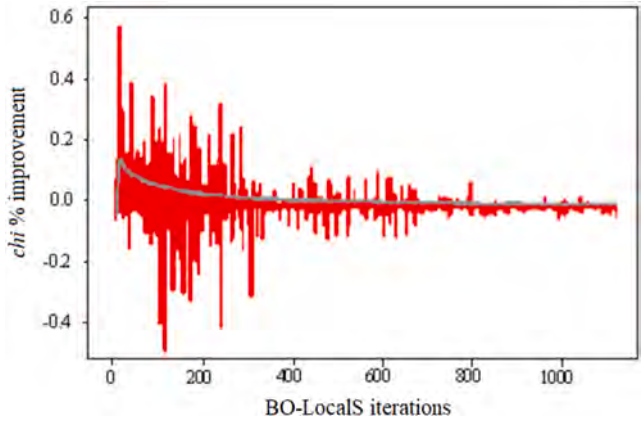

Fig. 4. BO-LocalS Execution: Evolution of *chi* (red) and Kalman filter estimator (grey).

- **Related to the problem description nature**. As previously justified, the current algorithm is designed to work only under the following conditions:
  - o **Number of vehicles**: Only one truck and any number of drones.
  - o **Vehicle autonomies:** The truck has unlimited autonomy, while the drones do not.
  - o **Vehicle speeds:** Constant for both types of vehicles.
  - o **Additional factors:** Wind and acceleration are neglected.
  - o **Synchronisation time:** Battery exchange time is neglected.
- **Related to the nature of multi-objective optimisation.** Researchers and practitioners always count on the presented drawbacks when developing new multi-objective algorithms and methodologies:
  - o ***chi* metric:** Using a hypervolume-based metric for MOO has some pitfalls, such as scalability limitations (solved by Pareto Scale), difficulty in interpretation, lack of diversity assessment, and sensitivity to the scale and range of the objectives.
  - o **Pareto Dominance**: Pareto dominance relationships may not capture certain complex relationships between objectives such as compromises and nonlinear interactions.
  - o **Lack of a single solution:** Although this can be adjusted using the *PS* parameter, by not counting single solutions, DMs may find it difficult to select solutions from a Pareto optimal solution set.
  - o **Difficulty in Decision Making**: The choice of the most suitable solution from the optimal Pareto set often involves subjective preferences and trade-offs, which may be difficult for the DM. Additional tools and methods to assist decision making can help make better informed decisions.
  - o **Computational Complexity**: The search for multiple Pareto optimal solutions requires the exploration of a larger solution space, the evaluation of multiple objectives, and the handling of compromises. This increased complexity can limit the scale and complexity of problems that can be effectively solved in a reasonable time frame.
  - o **Limited Knowledge of the True Pareto Front**: In many practical applications, the true Pareto front is unknown and difficult to obtain. This makes it difficult to accurately assess the performance of the algorithm. Approximation techniques or substitute models are often used to estimate Pareto fronts, which can generate additional uncertainties.

Additionally, and as a drawback of the flexibility of the algorithm, the technological parameters such as drone autonomy and vehicle speed need to be manually selected by the DM to configure the system. Despite these limitations, BOA-KS remains a valuable approach to tackling the problem at hand.

5. Computational study

The present study aims to facilitate the efficient operation of a truck in conjunction with drones, considering both the quality of the service provided and the environmental impact of the route application in a combined delivery with drones. For this purpose, we have used the usual statistical methodology in experimentation, the DOE (Design of Experiments) as well as two post hoc analyses. The study has focused on the analysis of two cases that differ in size, according to the number of customers served. The experiments have been set up and executed on an Intel Core i7 CPU 930 @ 2.80 GHz and 12 GB of RAM computer.

The instances chosen to perform the experimental study are the 20 nodes (small-size instance with one depot and 19 customers) and 100 nodes (medium-size instance with one depot and 99 customers) obtained from the uniform instances 61_n_20 and 91_n_100, respectively, from Agatz, Bouman, and Schmidt (2018) as representative for the more stringent scenarios.

5.1. Design of Experiments (DOE)

The study is focused on assessing, from an applied point of view, the effect that modifying the number, speed, and autonomy of drones has on the performance of the system.

Drone autonomy is defined as the product of the mean truck arc time among all paths in the network multiplied by the drone autonomy factor (D_{af}), as in Gonzalez-R et al. (2020). In addition to being instance-dependent, autonomy levels are a value of the intensity of the drone's "multi-trip" feature.

The speed of the drones (D_{sf}) is defined as proportional to truck speed, according to the drone speed factor, as done by Gonzalez-R et al. (2020). Again, this parameter favours the possibility that the drone visits a larger number of customer nodes before its next rendezvous with the truck.

The size of the drone fleet (k) defines the number of drones available for the delivery mission. It is defined by the user. This factor is the main source of complexity for the algorithm, as it defines the number of delivery agents able to simultaneously deliver goods or travel on the truck.

The response variable is *chi*, as the indicator can measure the quality of the different Pareto front approximations.

To reach conclusions, each instance is analysed performing a full factorial DOE. To estimate the nonlinearity of the response, two factors at three levels and one factor at four levels were used for each experiment (see Table 4).

5.2. Results

The main results obtained in the experiments are shown below. First, Tables 5 and 6 show the ANOVA statistical analysis, which indicates which factors (and/or interactions) have a significant impact on the response variable (*chi*) and which factors' influence is not distinguishable from the error term. Factors and interactions among factors that are meaningful are marked with an asterisk (*) and highlighted by the *p*-values. According to the methodology and the selected level of significance (5 % in our study), those factors (or interactions) with $p < 0.05$ are considered meaningful. Note that the values of the F-ratio indicate the factors (or interactions) with greater influence, showing a relative score of importance in the system performance.

This analysis suggests that all main factors and interactions are meaningful in the small-scale scenario.

Therefore, in the medium-scale instance, the main factors (D_{sf} and k) have a significant impact on the response variable, but most interaction terms are not statistically significant. Furthermore, Figs. 5 and 6 show the marginal means plots, which provides a visual representation of how the levels of each factor influence the response (*chi*) on average, allowing for an assessment of non-linearity.

Table 4
DOE configuration table.

Factor	Description	Levels
D_{af}	Drone autonomy factor	{**2**,**3**,**4**}
D_{sf}	Drone speed factor	{**2**,**3**,**4**}
k	Number of drones	{**1**,**2**,**3**,**4**}
Performance indicator	**Description**	
chi	Coefficient of Hypervolume Indicator	
DOE configuration	Order of data collection	Random
	Design	36-run Full Factorial
	Replications	5
	Total trials	180
	Alpha level	0.05

Table 5
Small-scale instance. ANOVA analysis.

Source	SS	DF	MS	F-ratio	I-hat	p-value
D_{af}	2,204E + 07	2	1,102E + 07	**14,42**	*	<0,001
D_{sf}	1,588E + 09	2	7,940E + 08	**1039**	*	<0,001
k	2,199E + 09	3	7,330E + 08	**959**	*	<0,001
D_{af} x D_{sf}	1,859E + 09	4	4,647,000	**6,079**	*	<0,001
D_{af} x k	3,052E + 09	6	5,087,000	**6,655**	*	<0,001
D_{sf} x k	6,584E + 09	6	1,304E + 08	**143,6**	*	<0,001
D_{af} x D_{sf} x k	4,670E + 09	12	3,892,000	**5,092**	*	<0,001
Within (error)	1,101E + 08	144	764,300			
TOTAL	**4,673E + 09**	**179**				

Table 6
Medium-scale instance. ANOVA analysis.

Source	SS	DF	MS	F-ratio	I-hat	p-value
D_{af}	3,298E + 08	2	1,649E + 08	2,512		0,085
D_{sf}	1,247E + 11	2	6,235E + 10	**949,8**	*	<0,001
k	2,535E + 11	3	8,451E + 10	**1287**	*	<0,001
D_{af} x D_{sf}	1,748E + 08	4	4,371E + 07	0,6658		0,617
D_{af} x k	5,036E + 08	6	8,393E + 07	1,279		0,271
D_{sf} x k	7,308E + 10	6	1,218E + 10	**185,5**	*	<0,001
D_{af} x D_{sf} x k	1,202E + 09	12	1,002E + 08	1,526		0,121
Within (error)	9,453E + 09	144	6,565E + 07			
TOTAL	**4,630E + 11**	**179**				

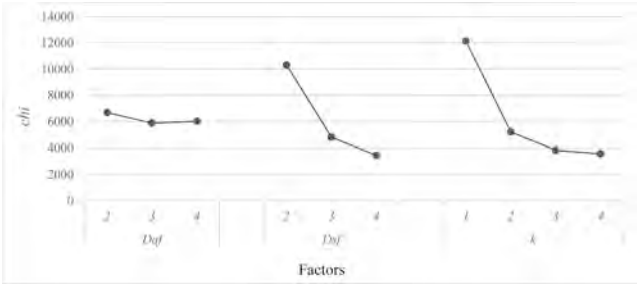

Fig. 5. Small-scale scenario. Marginal means response averages plot.

6. Results discussion

According to the ANOVA results (Tables 5 and 6), drone speed (D_{sf}) and number of drones (k), as well as their interaction, have a significant impact on the response (*chi*) at the specified levels in the instances studied. However, changes in autonomy levels (D_{af}) do not have a meaningful impact on system performance in the medium-scale scenario, nor any of the interactions containing this factor.

This result is much more evident in the marginal means chats in both instances (Figs. 5 and 6), where the effect on the response is much lower in the autonomy factor (D_{af}) than in the rest of the factors, i.e. drone speed D_{sf} and number of drones k). Far from being unexpected, this is a

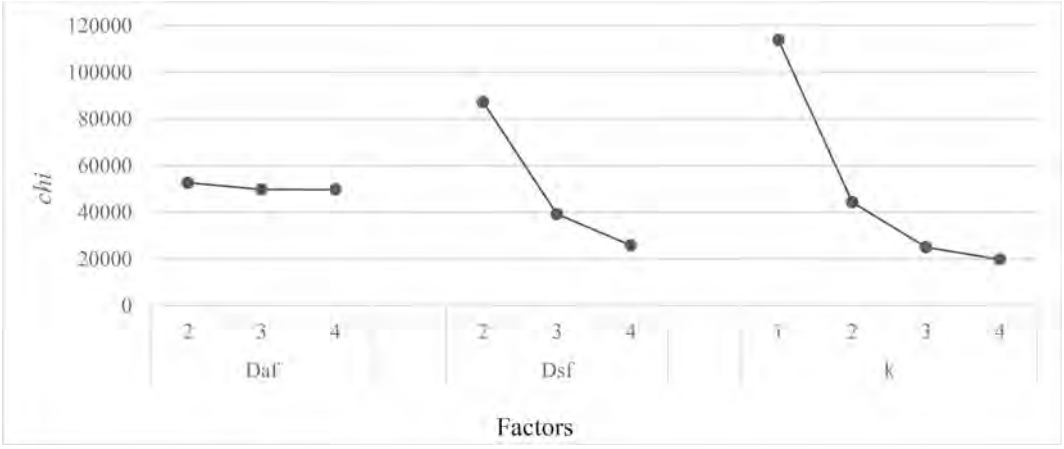

Fig. 6. Medium-scale instance. Marginal means response averages.

common effect when the density of nodes (number of nodes for the same area) is increased.

Therefore, in scenarios with a larger number of nodes, the combinatorial possibilities for autonomous actions may become less significant compared to scenarios with fewer nodes. This can lead to the following observation: for the same service area, the importance of the autonomy factor is greater when dealing with smaller numbers of customers or nodes. As the size of the instance grows, the potential improvement from increasing autonomy becomes less significant.

Above results highlight the sensitivity of the chosen performance indicator to changes in the factor levels. In smaller instances, even slight changes in factors can lead to a significant percentage of performance improvement, whereas in larger instances these changes may have, comparatively, a minor impact.

Further insights with practical implications for decision making are connected to the number of customers to serve. In the case of a small number of customers, increased levels of autonomy may produce greater performance improvements during the optimisation process. In contrast, in the case of a large number of customers, the impact of autonomy variations is expected to be relatively lower.

6.1. Post hoc analysis on the effect of speed and number of drones

According to Figs. 5 and 6, the decreasing effect on (*chi*) with higher levels of drone speed (D_{sf}) and number of drones (k) suggests that increasing these factors beyond a certain point may not result in a proportional improvement in system performance. Furthermore, the latter is also observed when studying the effects graph in Fig. 7.

Comparing levels 3 and 4 for D_{sf} when $k = 3$, it can be stated that $D_{sf} = 3$ is technologically more efficient since almost the same result is obtained while consuming fewer resources.

This observation highlights the practical importance of proper selection of both the speed and number of drones. Since blindly increasing the speed and/or number of drones does not always lead to the desired improvements in the system performance, there may be an optimal balance that achieves the desired performance without an unnecessary allocation of resources.

6.2. Post hoc analysis on the service time (T) and the total truck time (T_t)

The chosen response variable *chi* is a quality metric that assesses the proximity of solutions to the ideal and unattainable point where the problem is solved with zero resource consumption. Smaller values of *chi* indicate better solutions, this is, closer to this ideal point. The Pareto fronts obtained by the proposed methodology represent trade-offs between the two objectives under study, i.e., service time and total truck time. Indeed, these fronts depict the set of non-dominated or estimated Pareto-optimal solutions, where improving one objective comes at the expense of the other.

Moreover, understanding the shape of these fronts provides deeper insights into the trade-offs between conflicting objectives and is essential for effective decision-making when considering real-world parameters like total service time (T) and total truck time (T_t).

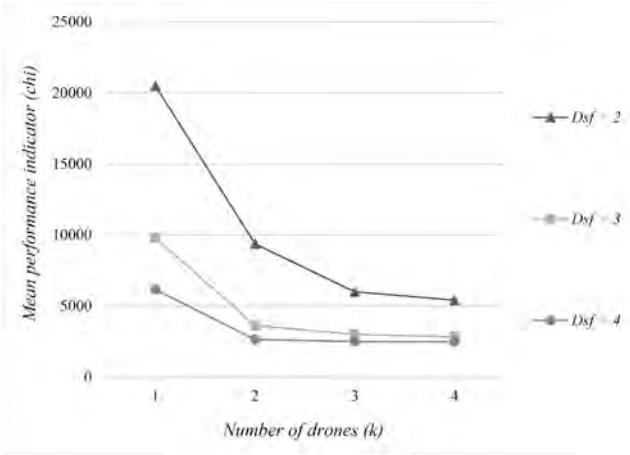

Fig. 7. Small-scale instance effects plot for D_{sf} and k factors.

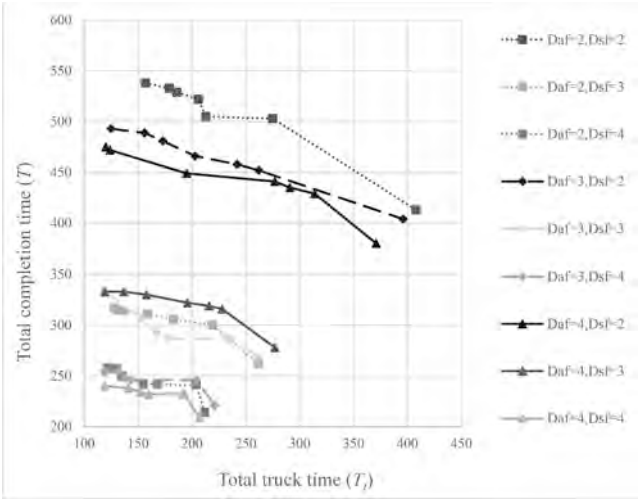

Fig. 8. Medium-scale experiment Pareto fronts for $k = 1$

First, we will analyse Fig. 8, where the graphical representation of the Pareto fronts of all combinations of the factors D_{af} and D_{sf} for the case of a drone ($k = 1$) can be found. The following facts jump out:

1. As would be expected after the analysis already performed of the influence of the factors on *chi*, rising the level of the D_{sf} factor represents a large overall improvement in the Pareto fronts obtained, which is visually evident in the distinguishable division between the fronts corresponding to the different levels of this factor.
2. However, far from the ideal Pareto fronts represented in the literature and having a curvilinear shape, the obtained fronts are relatively straight and represent a much lower variability in T compared to T_t. This effect is accentuated in cases where D_{sf} is higher.
3. The closest point to the far right indicates a high increase in T_t, while not having a great impact on T.

Point 2 leads to the following result: It is possible to achieve large reductions in truck time (along with emissions and fuel usage) while obtaining small increases in total service time, and this trade-off is even more accentuated as the drone speed becomes higher than the truck speed. Noticeably, the ability to reduce truck time while maintaining a relatively small increase in total service time allow companies to transform their logistics operations towards energetic efficiency.

Point 3, on its side, gives us the following result: greedily pursuing a total service time reduction increases the risk of having a great negative impact on total truck usage. Since increasing total truck usage means increased fuel consumption and maintenance costs, it is a driver of higher economic and environmental costs within logistics operations.

When analysing Fig. 9, where the same fronts are represented for the case of four drones ($k = 4$) the shape of the Pareto fronts is closer to those normally represented in the literature (knee shape). In this representation, as the use of four drones implies a great logistic advantage, nearly half of the fronts are reduced to a single point, where the results are $T = 119, Tt = 119$ time units. This point could be considered a unique efficient adjustment for the multi-objective problem.

The managerial lesson on this occasion would be as follows: making use of a higher number of drones leads the Pareto front shape closer to those in the literature (Choachaicharoenkul & Wattanapongsakorn, 2020) where the search algorithms generate natural 'knee zones', closer to the origin. This behaviour becomes even more evident as higher values for the factors are chosen, that is, as greater availability of resources is considered.

Besides, the T_t value for the points located in the knee zones is approximately the same. Therefore, the efficient configuration and management of multiple drone fleets is the main enabler of improvement. In fact, for each combination of factors and while their levels are increased, this continuous improvement is obtained by the decrease of the T value, as solutions capable of taking advantage of the system possibilities of using multiple drones lead to a best-fit synchronisation, leading to solutions where the truck waiting times are minimised.

To have a better image of how this enormous change is possible in the shapes and values of the Pareto front, Fig. 10 represents how the level variation in factor k while maintaining the same combination for the rest of the factors ($D_{sf} = 2, D_{af} = 2$) influences on the Pareto front shape.

7. Conclusions and future research

This paper proposes an MOO methodology for supporting DMs on how to best exploit a general version of the hybrid truck multi-drone problem referred to as TmDTL: multi-visit flight missions, truck and drone allowed for serving each customer location and rendezvous at different points than that from launches.

When addressing the TmDTL problem from the operational point of view, the focus is evolving from the sole consideration of the quality of service (makespan) to also adding the environmental impact (truck time) of implementing the routing. The proposed methodology applied to the bi-objective has shown to be useful in providing DMs with a spread front of good solutions among which they should pick the ultimate routing to be implemented.

One of the limiting aspects of Pareto front estimations is the computational effort required. In our research, we have applied a novel solution coding which turns the routing problem into a permutation flow-shop scheme. This transformation has allowed us to use, for the very first time in the multi-drone research stream, a greedy-based heuristic to drastically improve the quality of the solutions. Moreover, inspired by the state of the art of previous MOO research, we propose the so-called Bi-Objective Algorithm with Kalman Stop (BOA-KS) consisting of a global optimisation scheme using an SA algorithm, along with an innovative use of a Kalman filter indicator as a stopping criterion (self-triggered mechanism) to get an evolutionary path to transit from the

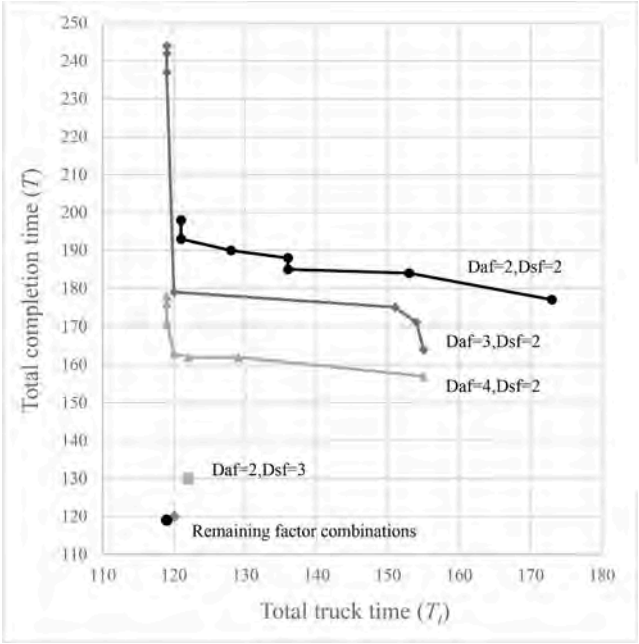

Fig. 9. Medium-scale instance Pareto fronts for $k = 4$

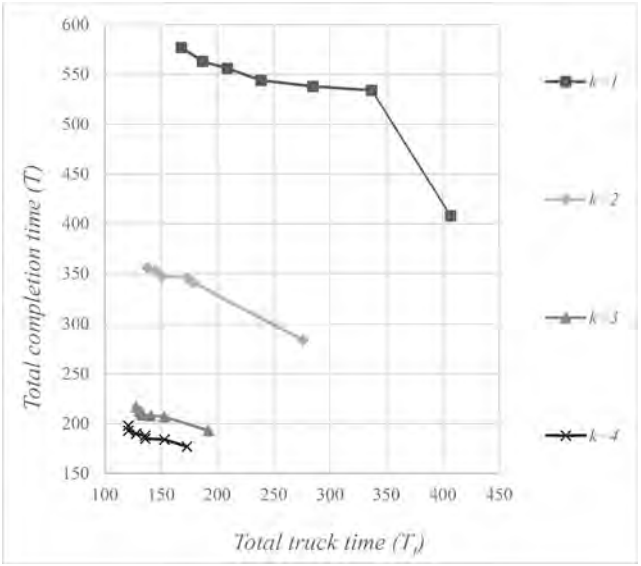

Fig. 10. Medium-scale instance Pareto fronts for $D_{af} = 2,\ D_{sf} = 2$

good initial solutions to the generation of estimated Pareto fronts for the problem at hand. Furthermore, the number of solutions on the Pareto front can be adjusted by the DMs as an input parameter.

The proposed methodology has allowed to perform a DOE in two instances of different sizes, to assess the impact of different operations factors such as values of drone autonomy, as well as different values of drone speed and fleet size, using the *chi* indicator as response variable.

The study shows that the marginal profit from adding additional resources (i.e., drone fleet size, drone autonomy, and flight speed) is scale decreasing, in all the cases. Regarding drone autonomy (D_{af}) the smaller the number of customers to be served in the same area (i.e., the higher density of customers), the more significant this factor becomes. Therefore, efficient optimisation in this type of environment is decisive.

As expected, an increase in fleet size (k) leads to a general improvement in performance. However, according to the experiments, it usually results in a greater reduction in service time (makespan) rather than in truck time. An important aspect to be considered by the DM is drone speed (D_{sf}), since the increase in this factor results in a greater reduction in environmental impact (truck time), rather than in service time (makespan). Furthermore, a deep search for a reduction in truck time might ultimately lead to a negative impact on total service time.

The present work was conceived with the idea of covering a need for decision making in complex last-mile MOO problems. Once we have given rise to this first level of MOO route planning, our future work aims at incorporating operational features like uncertainty, that is, uncertain truck and/or drone travel times or balanced use of the batteries in the system. Robust planning is still an open issue, therefore evolving our methodology for providing robust delivery services is a natural future research line.

CRediT authorship contribution statement

Pedro L. Gonzalez-R: Conceptualization, Formal analysis, Visualization, Supervision. **David Sanchez-Wells:** Methodology, Software, Writing – original draft. **José L. Andrade-Pineda:** Investigation, Resources, Data curation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Funding

This work was supported by the University of Seville [Own Plan Predoctoral Training Programme, reference VII PPIT-2022-II.2] as part of the DESLOGA project, along with funding from the OMICRON project (European Union's Horizon 2020 research and innovation programme under grant agreement No 955269).

References

Abu Doush, I., El-Abd, M., Hammouri, A. I., & Bataineh, M. Q. (2021). The effect of different stopping criteria on multi-objective optimization algorithms. In *Neural Computing and Applications, 1*London: Springer. https://doi.org/10.1007/s00521-021-05805-1.

Agatz, N., Bouman, P., & Schmidt, M. (2018). Optimization approaches for the traveling salesman problem with drone. *Transportation Science, 52*(4), 965–981. https://doi.org/10.1287/trsc.2017.0791

Almuhaideb, S., Alhussan, T., Alamri, S., Altwaijry, Y., Aljarbou, L., & Alrayes, H. (2021). Optimization of truck-drone parcel delivery using metaheuristics. *Applied Sciences (Switzerland), 11*(14). https://doi.org/10.3390/app11146443

Audet, C., Bigeon, J., Cartier, D., Le Digabel, S., & Salomon, L. (2021). Performance indicators in multiobjective optimization. *European journal of operational research, 292*(2), 397–422. https://doi.org/10.1016/j.ejor.2020.11.016

Beume, N. (2009). S-Metric Calculation by Considering Dominated Hypervolume as Klee's Measure Problem. *Evolutionary Computation, 17*(4), 477–492. https://doi.org/10.1162/evco.2009.17.4.17402

Choachaicharoenkul, S., & Wattanapongsakorn, N. (2020). Post Pareto-optimal ranking algorithm for multi-objective optimization using extended angle dominance. *Expert Systems with Applications, 158*. https://doi.org/10.1016/j.eswa.2020.113446

Coelho, B. N., Coelho, V. N., Coelho, I. M., Ochi, L. S., Haghnazar, K. R., Zuidema, D., Lima, M. S. F., & da Costa, A. R. (2017). A multi-objective green UAV routing problem. *Computers and Operations Research, 88*, 306–315. https://doi.org/10.1016/j.cor.2017.04.011

Coelho, V. N., Coelho, I. M., Coelho, B. N., Cohen, M. W., Reis, A. J. R., Silva, S. M., … Guimarães, F. G. (2016). Multi-objective energy storage power dispatching using plug-in vehicles in a smart-microgrid. *Renewable Energy, 89*, 730–742. https://doi.org/10.1016/j.renene.2015.11.084

Dasdemir, E., Köksalan, M., & Tezcaner Öztürk, D. (2020). A flexible reference point-based multi-objective evolutionary algorithm: An application to the UAV route planning problem. *Computers and Operations Research, 114*. https://doi.org/10.1016/j.cor.2019.104811

de Freitas, J. C., & Penna, P. H. V. (2020). A variable neighborhood search for flying sidekick traveling salesman problem. *International Transactions in Operational Research, 27*(1), 267–290. https://doi.org/10.1111/itor.12671

Dell'Amico, M., Montemanni, R., & Novellani, S. (2022). Exact models for the flying sidekick traveling salesman problem. *International Transactions in Operational Research, 29*(3), 1360–1393. https://doi.org/10.1111/itor.13030

Demir, I., Ergin, F. C., & Kiraz, B. (2019). A New Model for the Multi-Objective Multiple Allocation Hub Network Design and Routing Problem. *IEEE Access, 7*, 90678–90689. https://doi.org/10.1109/ACCESS.2019.2927418

di Puglia Pugliese, L., & Guerriero, F. (2017). Last-Mile Deliveries by Using Drones and Classical Vehicles. In A. Sforza, & C. Sterle (Eds.), *Optimization and Decision Science: Methodologies and Applications* (Vol. 217, pp. 557–565). Springer International Publishing. https://doi.org/10.1007/978-3-319-67308-0.

Ferrandez, S. M., Harbison, T., Weber, T., Sturges, R., & Rich, R. (2016). Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm. *Journal of Industrial Engineering and Management, 9*(2), 374–388. https://doi.org/10.3926/jiem.1929

Framinan, J. M., & Leisten, R. (2008). A multi-objective iterated greedy search for flowshop scheduling with makespan and flowtime criteria. *OR Spectrum, 30*(4), 787–804. https://doi.org/10.1007/s00291-007-0098-z

Gonzalez-R, P. L., Canca, D., Andrade-Pineda, J. L., Calle, M., & Leon-Blanco, J. M. (2020). Truck-drone team logistics: A heuristic approach to multi-drop route planning. *Transportation Research Part C: Emerging Technologies, 114*, 657–680. https://doi.org/10.1016/j.trc.2020.02.030

Gu, R., Poon, M., Luo, Z., Liu, Y., & Liu, Z. (2022). A hierarchical solution evaluation method and a hybrid algorithm for the vehicle routing problem with drones and multiple visits. *Transportation Research Part C: Emerging Technologies, 141*(March). https://doi.org/10.1016/j.trc.2022.103733

Guerriero, F., Surace, R., Loscrí, V., & Natalizio, E. (2014). A multi-objective approach for unmanned aerial vehicle routing problem with soft time windows constraints. *Applied Mathematical Modelling, 38*(3), 839–852. https://doi.org/10.1016/j.apm.2013.07.002

Ha, Q. M., Deville, Y., Pham, Q. D., & Hà, M. H. (2018). On the min-cost Traveling Salesman Problem with Drone. *Transportation Research Part C: Emerging Technologies, 86*, 597–621. https://doi.org/10.1016/j.trc.2017.11.015

Karak, A., & Abdelghany, K. (2019). The hybrid vehicle-drone routing problem for pick-up and delivery services. *Transportation Research Part C, 102*(September 2018), 427–449. https://doi.org/10.1016/j.trc.2019.03.021

Leon-Blanco, J. M., Gonzalez-R, P. L., Andrade-Pineda, J. L., Canca, D., & Calle, M. (2022). A multi-agent approach to the truck multi-drone routing problem. *Expert Systems with Applications, 195*. https://doi.org/10.1016/j.eswa.2022.116604

Li, H., Chen, J., Wang, F., & Zhao, Y. (2022). Truck and drone routing problem with synchronization on arcs. Naval Research Logistics, May 2021, 1–18. 10.1002/nav.22053.

Lin, C. (2020, March 17). Delivery Technology Is Keeping Chinese Cities Afloat Through Coronavirus. Harvard Business Review. https://hbr.org/2020/03/delivery-technology-is-keeping-chinese-cities-afloat-through-coronavirus.

Luo, Z., Poon, M., Zhang, Z., Liu, Z., & Lim, A. (2021). The Multi-visit Traveling Salesman Problem with Multi-Drones. *Transportation Research Part C: Emerging Technologies, 128*(April), Article 103172. https://doi.org/10.1016/j.trc.2021.103172

Marinelli, M., Caggiani, L., Ottomanelli, M., & Dell'Orco, M. (2018). En route truck-drone parcel delivery for optimal vehicle routing strategies. *IET Intelligent Transport Systems, 12*(4), 253–261. https://doi.org/10.1049/iet-its.2017.0227

Martí, L., García, J., Berlanga, A., & Molina, J. M. (2016). A stopping criterion for multi-objective optimization evolutionary algorithms. Information Sciences, 367–368 (November 2017), 700–718. 10.1016/j.ins.2016.07.025.

Masone, A., Poikonen, S., & Golden, B. L. (2022). The multivisit drone routing problem with edge launches: An iterative approach with discrete and continuous improvements. Networks, December 2021, 1–23. 10.1002/net.22087.

Moshref-Javadi, M., Hemmati, A., & Winkenbach, M. (2020). A truck and drones model for last-mile delivery: A mathematical model and heuristic approach. *Applied Mathematical Modelling, 80*. https://doi.org/10.1016/j.apm.2019.11.020

Moshref-Javadi, M., Van Cauwenberghe, K. P., McCunney, B. A., & Hemmati, A. (2023). Enabling same-day delivery using a drone resupply model with transshipment

points. *Computational Management Science, 20*(1), 22. https://doi.org/10.1007/s10287-023-00453-3

Murray, C. C., & Chu, A. G. (2015). The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies, 54*, 86–109. https://doi.org/10.1016/J.TRC.2015.03.005

Omagari, H., & Higashino, S.-I. (2018). Provisional-Ideal-Point-Based Multi-objective Optimization Method for Drone Delivery Problem. *International Journal of Aeronautical and Space Sciences, 19*(1), 262–277. https://doi.org/10.1007/s42405-018-0021-7

Petchrompo, S., Coit, D. W., Brintrup, A., Wannakrairot, A., & Parlikad, A. K. (2022). A review of Pareto pruning methods for multi-objective optimization. Computers and Industrial Engineering, 167(December 2021), 108022. 10.1016/j.cie.2022.108022.

Poikonen, S., & Golden, B. (2020). Multi-visit drone routing problem. *Computers & Operations Research, 113*, Article 104802. https://doi.org/10.1016/j.cor.2019.104802

Raj, R., & Murray, C. (2020). The multiple flying sidekicks traveling salesman problem with variable drone speeds. *Transportation Research Part C: Emerging Technologies, 120*, Article 102813. https://doi.org/10.1016/j.trc.2020.102813

Ramirez Atencia, C., del Ser, J., & Camacho, D. (2019). Weighted strategies to guide a multi-objective evolutionary algorithm for multi-UAV mission planning. Swarm and Evolutionary Computation, 44(June 2018), 480–495. 10.1016/j.swevo.2018.06.005.

Ramirez-Atencia, C., Bello-Orgaz, G., & R-Moreno, M. D., & Camacho, D.. (2017). Solving complex multi-UAV mission planning problems using multi-objective genetic algorithms. *Soft Computing, 21*(17), 4883–4900. https://doi.org/10.1007/s00500-016-2376-7

Rojas Viloria, D., Solano-Charris, E. L., Muñoz-Villamizar, A., & Montoya-Torres, J. R. (2021). Unmanned aerial vehicles/drones in vehicle routing problems: A literature review. *International Transactions in Operational Research, 28*(4), 1626–1657. https://doi.org/10.1111/itor.12783

Schermer, D., Moeini, M., & Wendt, O. (2019). A hybrid VNS/Tabu search algorithm for solving the vehicle routing problem with drones and en route operations. *Computers and Operations Research, 109*, 134–158. https://doi.org/10.1016/j.cor.2019.04.021

Shavarani, S. M., Nejad, M. G., Rismanchian, F., & Izbirak, G. (2018). Application of hierarchical facility location problem for optimization of a drone delivery system: A case study of Amazon prime air in the city of San Francisco. *The International Journal of Advanced Manufacturing Technology, 95*(9–12), 3141–3153. https://doi.org/10.1007/s00170-017-1363-1

Tavana, M., Khalili-Damghani, K., Santos-Arteaga, F. J., & Zandi, M. H. (2017). Drone shipping versus truck delivery in a cross-docking system with multiple fleets and products. *Expert Systems with Applications, 72*, 93–107. https://doi.org/10.1016/j.eswa.2016.12.014

Wang, K., Yuan, B., Zhao, M., & Lu, Y. (2020). Cooperative route planning for the drone and truck in delivery services: A bi-objective optimisation approach. *Journal of the Operational Research Society, 71*(10), 1657–1674. https://doi.org/10.1080/01605682.2019.1621671

While, L., Bradstreet, L., & Barone, L. (2012). A fast way of calculating exact hypervolumes. *IEEE Transactions on Evolutionary Computation, 16*(1), 86–95. https://doi.org/10.1109/TEVC.2010.2077298

Yan, M., Yuan, H., Xu, J., Yu, Y., & Jin, L. (2021). Task allocation and route planning of multiple UAVs in a marine environment based on an improved particle swarm optimization algorithm. *Eurasip Journal on Advances in Signal Processing, 2021*(1). https://doi.org/10.1186/s13634-021-00804-9

Zajac, S., & Huber, S. (2021). Objectives and methods in multi-objective routing problems: A survey and classification scheme. *European Journal of Operational Research, 290*(1), 1–25. https://doi.org/10.1016/j.ejor.2020.07.005

Zhou, T., Zhang, J., Shi, J., Liu, Z., & Huang, J. (2018). Multidepot UAV routing problem with weapon configuration and time window. *Journal of Advanced Transportation, 2018*. https://doi.org/10.1155/2018/7318207