Expert Systems With Applications 247 (2024) 123106

Contents lists available at ScienceDirect

Expert Systems With Applications

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

New features for customer classification in the Flying Sidekick Traveling Salesman Problem

Maurizio Boccia [a], Andrea Mancuso [a], Adriano Masone [a,*], Teresa Murino [b], Claudio Sterle [a,c]

[a] Department of Electrical Engineering and Information Technology, University of Naples "Federico II", Naples, 80125, Italy
[b] Department of Chemical, Materials and Industrial Production Engineering, University of Naples "Federico II", Naples, Italy
[c] Istituto di Analisi dei Sistemi ed Informatica "A. Ruberti", IASI-CNR, Rome, Italy

ARTICLE INFO

Keywords:
Drone
FS-TSP
Routing
Customer classification
Customer features
Computational study

ABSTRACT

The increasing interest in the deployment of truck and drone delivery systems leads to the definition of new and complex vehicle routing problems. In this context, the flying sidekick traveling salesman problem (*FS-TSP*) is the first truck-and-drone routing problem defined in the literature. Several variants appeared in the last years differing in the operating conditions and the structure of the hybrid truck-and-drone delivery system. Exact and heuristic solution methods have been proposed in the literature to solve these problems effectively. However, the exact solution methods can generally solve only small-size instances due to the complexity of these problems. On the other hand, heuristic solution methods are able to find feasible solutions with an acceptable computational burden but without any guarantee of the quality of the solution. This work aims to investigate the possibility of using data science and machine learning techniques to reduce the complexity of solving an *FS-TSP* instance. The idea is to determine a good/optimal customer-to-vehicle assignment apriori to reduce the number of decisions involved in the *FS-TSP* solution. The assignment is determined through the classification of customers based on a subset of features specifically defined for the *FS-TSP*. This information can be exploited by existing solution approaches for the *FS-TSP* to improve their performance. An extensive computational campaign on benchmark instances is carried out with a twofold objective. On the one hand, we aim to evaluate the impact of the features on customer classification. On the other hand, we intend to show the relationship between classification and combinatorial optimization results by observing the effect of the customer classification on the *FS-TSP* solution.

1. Introduction

Truck and drone delivery systems have received a lot of attention from the scientific community in the last years due to the potential benefits that they can lead to last mile logistics (*LML*). This kind of hybrid system promises new levels of efficiency for the *LML*, which is notoriously known as the most expensive part of the freight distribution process. In this context, several tactical and operational problems need to be addressed to deploy a hybrid system able to support and improve *LML* operations (Chung, Sah, & Lee, 2020). Routing problems represent one of the most investigated issues due to the synchronization requirement between the two vehicles. In particular, the Flying Sidekick Traveling Salesman Problem (*FS-TSP*) is the first truck and drone routing problem studied in literature (Murray & Chu, 2015). It involves designing the optimal route for a hybrid delivery system consisting in a truck and a drone working in tandem, with the aim of minimizing the delivery completion time. The *FS-TSP* assumptions can be briefly summarized as follows: the truck-and-drone system departs from and returns to a central depot; every customer is visited exactly once, by either the truck or the drone; the truck serves as a mobile depot for the drone, supplying it with parcels and swapping its battery due to the drone limited flight endurance. Furthermore, the drone is launched from the truck to deliver to a single customer and subsequently returns either to the truck or to the depot, in what is termed a "drone sortie". For safety reasons, the drone cannot land and must hover if it has to wait for the truck at a recovery location. The delivery completion time is equal to the moment when both the truck and the drone return to the depot after serving all customers.

Afterward, the literature on this topic exponentially increased with several contributions introducing variants of the *FS-TSP* involving different problem assumptions and system operations and/or proposing exact and heuristic solution methods for the *FS-TSP* or its variants. A

* Corresponding author.
E-mail addresses: maurizio.boccia@unina.it (M. Boccia), andrea.mancuso@unina.it (A. Mancuso), adriano.masone@unina.it (A. Masone), murino@unina.it (T. Murino), claudio.sterle@unina.it (C. Sterle).

https://doi.org/10.1016/j.eswa.2023.123106
Received 17 March 2023; Received in revised form 3 November 2023; Accepted 29 December 2023
Available online 11 January 2024

complete review of the contributions on this topic is out of the scope of this work. Thus, we refer the interested reader to the surveys (Madani & Ndiaye, 2022; Moshref-Javadi & Winkenbach, 2021; Rojas Viloria, Solano-Charris, Muñoz-Villamizar, & Montoya-Torres, 2021). However, we point out that the proposed solution methods for the *FS-TSP* or its variants generally present some drawbacks. In particular, exact solution methods suffer from dimensional issues that make them impractical to solve, even for small instances. On the other hand, heuristic solution methods are able to determine a feasible solution with acceptable computational burdens but often without any guarantee of the quality of the solution. In the following, we will briefly outline the main differences between the *FS-TSP* and its variants, and then we will focus on the contributions that specifically tackle the *FS-TSP*. Regarding the literature that introduces variants of the *FS-TSP*, these variants differ from the *FS-TSP* with respect to the objective function (Ha, Deville, Pham, & Hà, 2018; Tamke & Buscher, 2023), the drone capability to land (de Freitas & Penna, 2020; Freitas, Penna, & Toffolo, 2023), the number of customers a drone can serve in a single sortie (Mara, Rifai, & Sopha, 2022; Masone, Poikonen, & Golden, 2022), the possibility of launching and retrieving a drone at the same location (Agatz, Bouman, & Schmidt, 2018; Boccia, Masone, Sforza, & Sterle, 2021d), the option of multiple drone launches from the same spot (Boccia, Mancuso, Masone, Sforza and Sterle, 2021; Chang & Lee, 2018), the availability of multiple drones on a truck (Salama & Srinivas, 2022; Tiniç, Karasan, Kara, Campbell, & Ozel, 2023), and the use of several truck-and-drone systems for deliveries (Kuo, Lu, Lai, & Mara, 2022; Sacramento, Pisinger, & Ropke, 2019).

Regarding contributions that focus on the *FS-TSP*, the majority of studies present a *FS-TSP* formulation along with exact or heuristic solution methods. Among the different exact solution approaches (Boccia, Mancuso, Masone, & Sterle, 2023; Boccia, Masone, Sforza, & Sterle, 2021c; Dell'Amico, Montemanni, & Novellani, 2022; Roberti & Ruthmair, 2021; Schermer, Moeini, & Wendt, 2020), the state-of-the-art is represented by the branch-and-cut and branch-and-price methods proposed in Boccia et al. (2023) and Roberti and Ruthmair (2021), respectively. These methods are able to optimally solve instances up to 40 customers. On the other hand, concerning the heuristic solution methods (Dell'Amico, Montemanni, & Novellani, 2021a, 2021b; Murray & Chu, 2015; Yu, Lin, Jodiawan, & Lai, 2023), the state-of-the-art is the algorithm based on branch-and-bound presented in Dell'Amico et al. (2021a), being able to address instances with up to 200 customers.

In this work, we investigate the potential benefits of using data science and machine learning techniques in conjunction with combinatorial optimization methods to solve the *FS-TSP*. The idea is to identify promising or good customer-to-vehicle assignments through classification methods. These assignments can be used to reduce the size and complexity of the *FS-TSP* instances. The ultimate goal is to effectively and efficiently solve the *FS-TSP* through solving a reduced problem.

The use of machine learning and combinatorial optimization is another cutting-edge topic (Bengio, Lodi, & Prouvost, 2021). Different studies explore the benefits of combining these approaches in different ways (de Holanda Maia, Plastino, & Penna, 2018, 2020; Karimi-Mamaghan, Mohammadi, Meyer, Karimi-Mamaghan, & Talbi, 2022; Martins, Vianna, Rosseti, Martins, & Plastino, 2018; Mazyavkina, Sviridov, Ivanov, & Burnaev, 2021). However, to the best of our knowledge, only the study reported in Boccia, Mancuso, Masone and Sterle (2021) investigates the possibility of using machine learning techniques in combination with combinatorial operation specifically for truck and drone routing problems. The authors propose seven features directly inferred by the *FS-TSP* assumptions. Then, they use the Scikit-learn package to build classification models on a defined data set. The classification results are then used to solve the *FS-TSP*. In this work, from a methodological point of view, we extend this study by defining a subset of new features that consider the topography of the *FS-TSP* underlying graph, and by integrating feature selection and classifier parameter tuning in the hybrid solution approach. On this basis, we carry out an extensive computational campaign with a twofold objective. On the one hand, we aim to determine the impact of the features on customer classification, identifying the most important one. On the other hand, we intend to evaluate the relationship between classification and combinatorial optimization results and the impact of the features and classification parameters on the *FS-TSP* solution.

The remainder of the paper is organized as follows: in Section 2, we provide a brief description of the *FS-TSP* and the customer classification solution approach proposed in Boccia, Mancuso, Masone, Sterle (2021); in Section 3, we present the features defined for the *FS-TSP* customer classification; Section 4 is devoted to the experimentation on the customer classification; Section 5 reports the experimentation related to the use of the results of the customer classification for the *FS-TSP* solution; finally, conclusions are given, and perspectives on future works on this topic are discussed in Section 6.

2. FS-TSP description and customer classification approach

In this section, we first describe the *FS-TSP* and introduce the corresponding notation used in the next section. Then, we discuss the *FS-TSP* customer classification and the related solution approach.

2.1. Problem description and notation

The *FS-TSP* involves a truck and a drone working together to deliver packages to a set of customers, denoted by C. The route for the truck-and-drone system starts and ends at a depot, denoted by o. The truck can perform two types of operations: serving customers and supporting the drone in its deliveries. Therefore, let T be the subset of customers that can be served by the truck. Generally, the truck is capable of serving all customers, so T is equal to C. In supporting the drone, the truck provides it the parcels for delivery to customers, launches the drone, and then recollects it at the end of the drone sortie. For safety reasons, the drone is not permitted to land while awaiting the truck arrival in a rendezvous location. The launch and retrieval operations can be performed at the depot or at a customer location. The launch and rendezvous locations of a sortie must be distinct. Both launching and retrieval operations incur overhead times, indicated by SL and SR, respectively. Then, the truck also supports the drone by replacing its battery since its capacity, expressed in terms of flight time and denoted as Dtl, is limited. Additionally, the drone expends energy during retrieval operations, hence the associated overhead time is accounted for in the drone flight duration. Due to the drone technological limitations, if a customer parcel exceeds the maximum payload, or if the flight time required to serve a customer surpasses the drone endurance, then the drone cannot serve these customers. Accordingly, let D represent the subset of customers serviceable by the drone.

In terms of the routes that the two types of vehicles can follow, the truck and the drone each have their own set of arcs they can traverse, referred to as A_T and A_D, respectively. The drone arc set can be further broken down into three subsets: A_T, A_D^L, and A_D^R. The drone travels on arcs of A_T when is carried by the truck. Instead, it travels on arcs of A_D^L (A_D^R) when it is launched from (retrieved by) the truck before (after) a delivery. The truck and drone can consider different distance metrics ($d_{ij}, \forall (i,j) \in A_T$ and $\delta_{ij}, \forall (i,j) \in A_D$) and have different speeds (s and σ). Therefore, the truck and drone travel time can be different (indicated with t_{ij} and τ_{ij}, respectively). The objective of the *FS-TSP* is to minimize the time it takes for the truck and drone to complete all deliveries and return to the depot.

Fig. 1 shows an example of a *FS-TSP* solution for a small instance. Fig. 1(a) presents the instance details with the depot represented by a square and customers by circles. Solid lines indicate arcs that the truck can traverse in both directions, while dashed lines correspond to arcs that the drone can fly in both directions. Customer 1 cannot be serviced by the drone, hence it is not connected by a dashed line to any other

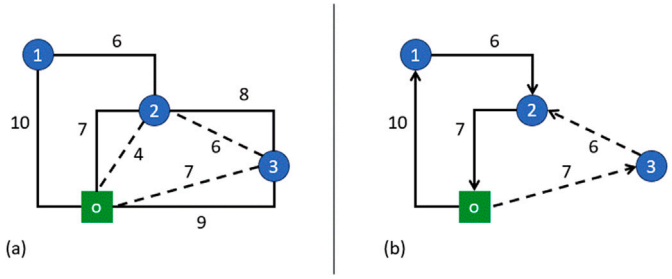

Fig. 1. Example of a *FS-TSP* solution.

node. Travel times are reported alongside the respective lines in time units. For the sake of simplicity, we point out that overhead related to launching and retrieving operations are set equal to 0. Fig. 1(b) reports a feasible *FS-TSP* solution. In this solution, the drone departs from the depot to serve customer 3, as the truck travels towards customer 1. The drone is later collected by the truck at customer 2. We highlight that since the drone reaches customer 2 before the truck, it will hover until the truck arrives. The delivery ends with both the truck and drone returning to the depot, with the drone on the truck, for a total delivery completion time of 23 time units. It is important to note that the feasibility of this solution depends on the drone endurance. If the drone endurance is sufficient to cover its flight time, including the hovering time at customer 2, the solution is feasible. If not, the solution will be infeasible.

2.2. The FS-TSP customer classification approach

Three kinds of decisions are involved in solving a *FS-TSP* instance: the subsets of customers served by the truck and the drone, respectively; the launch and the rendezvous nodes of each drone sortie; the sequence according to which the customers are served. However, these decisions cannot be taken independently. Indeed, the drone is not completely autonomous in its delivery operations due to its limited payload and battery capacity. Therefore, the truck and the drone movements have to be carefully coordinated (synchronized) to achieve the advantages arising from the use of this hybrid delivery system. On this basis, it is straightforward that the synchronization and coordination between the vehicle represent the main issues when addressing the *FS-TSP*.

In this context, customers demanding parcels whose weight exceeds the drone maximum payload will be obviously served by the truck. These customers will be referred to in the following as customers '*not eligible for drones*' (*NED*). Instead, we will refer to all the other customers as '*eligible for drones*' (*ED*). The *NED* customers require only two kinds of decisions out of three due to their nature. Therefore, the greater the percentage of *NED* customers, then the smaller the *FS-TSP* solution space. In particular, if the percentage of *NED* customers is equal to 100, then the optimal solution of the *FS-TSP* coincides with the optimal solution of the Traveling Salesman Problem (*TSP*).

Generally, *FS-TSP* instances are characterized by 10%–20% of *NED* customers. However, the minimum number of customers that will be served by the truck is equal to $|C| - \lceil |C|/2 \rceil$, being $\lceil |C|/2 \rceil$ the maximum number of customers that the drone can serve in a *FS-TSP* solution due to its assumptions. On this basis, the percentage of customers that can be served by the truck is about 50%. However, it should be noted that the number of customers served by the truck could potentially be higher, as the number of customers the drone serves in the optimal solution depends on customer locations and vehicle speeds. Indeed, results reported in Boccia et al. (2021d) indicate that in the majority of optimal solutions, the truck services between 80%–90% of the customers.

Based on these observations, the *FS-TSP* customer classification approach aims to determine in advance through machine learning techniques if a customer is *ED* or *NED* in the optimal solution. The idea is to reduce the solution space by fixing the value of the corresponding decision variables and then solve reduced *FS-TSP*. The fixing aim is to solve large-size instances by using exact solution methods or commercial *MILP* solver that, without the fixing, would not be able to address instances of that size. This approach is mainly based on the features defined to characterize the customers. These features have to consider problem aspects and system features that are not obvious and can provide insights on the possible *FS-TSP* optimal solution. However, as we will see in the computational experiments section on the *FS-TSP* solution, an effective method that combines machine learning with combinatorial optimization must also select an appropriate subset of features for classification and opportunely determine the parameters for the classification methods to further improve the performance of the combinatorial optimization solution method.

3. Features for customer classification in the FS-TSP

In this section, we first recall the seven features described in Boccia, Mancuso, Masone, Sterle (2021) and straightly based on the problem assumptions and system operations. Then, we present the new subset of features mainly based on the graph topography.

3.1. Features based on problem assumptions and system operations

The seven features used to classify customers for the *FS-TSP* were chosen based on the problem assumptions and system operations:

- Feature I: related to the weight of the parcel. It is a binary variable set to 1 if the parcel exceeds the maximum payload for the drone and 0 otherwise.
- Feature II: measures the potential savings if a customer (denoted as c) is not served. It is a continuous variable equal to the percentage difference between the optimal *TSP* solution length with and without c. Customers with large potential savings may be good candidates for drone delivery to parallelize the process. However, calculating this feature may be time-consuming or even impossible, in which case the length of the *TSP* can be replaced by a proxy that is easier to compute.
- Feature III: a measure of the customer centrality, modeled as a continuous variable equal to the sum of the distances from c to all other customers divided by the sum of all pairwise distances between customers. Customers with low centrality values may be suitable for truck delivery.
- Features IV and V: identify customers that could be promising launch and rendezvous locations for drone sorties. A direct feasible drone sortie is one in which the drone travels from one location (i) to another (j) via a third location (k), the truck travels directly between i and j, and the total travel time for both the drone and truck is less than drone endurance. Feature IV is an integer variable counting the number of direct feasible drone sorties involving a particular customer. In contrast, Feature V counts the number of feasible sorties where at least one of the launch or rendezvous locations is not eligible for drone delivery. Customers with a high number of direct feasible drone sorties are more likely to be used as launch or rendezvous locations in the *FS-TSP* solution.
- Features VI and VII: related to the duration of the shortest and longest drone paths that pass through a customer (c), respectively. These are continuous variables equal to the minimum and maximum values, respectively, of the sum of the drone distances from all other customers to c and from c to all other customers. If the shortest or longest path exceeds Dtl, the corresponding feature value will be set to a very large number.

3.2. Feature based on the graph topography

The newly identified features are designed by observing the optimal solution of several *FS-TSP* instances varying different instance parameters (e.g., vehicle speed, endurance). This analysis allows us to identify features that cannot be inferred by the problem assumption or system operations. Two ideas form the basis of the new features: a central point of the customer network, referred to as *barycenter*, and customer reachability. The *barycenter* will be denoted as $g(C)$ and its position can be computed for the instances where the customers are characterized by coordinates. Since truck and drone routing problems arise from logistics applications in most of the literature instances, this information is available. Therefore, indicating with x_p and y_p the generic coordinates of a point $p, p \in \mathbb{R}$, or a node $i, i \in C \cup \{o\}$, of the network, the expression of the coordinates of $g(C)$ is the following:

$$(x_{g(C)}, y_{g(C)}) = \left(\frac{\sum_{c \in C} x_c}{|C|}, \frac{\sum_{c \in C} y_c}{|C|} \right) \tag{1}$$

The instances in the *FS-TSP* are generally characterized by a complete graph (i.e., there is an arc for each pair of nodes of the graph). Therefore, each customer can reach the other customer in theory. However, drone endurance limits reachability when the two vehicles operate in tandem. Therefore, we define the subsets $R(c, \tau^*), \forall c \in C$, as the subsets of customers that are reachable by the drone within the travel time threshold τ^* from customer c, $R(c, \tau^*) = \{i \in C \setminus \{c\} : \tau_{ic} \leq \tau^*\}$. Based on this notation, it is possible to introduce the new *FS-TSP* customer features. Since the new features are not intended to replace the basic features but instead used in combination with the former ones, we will refer to them following the same enumeration. All the feature expressions are referred to a generic customer $c, c \in C$.

3.2.1. Feature VIII

It expresses the ratio between the truck travel time and the drone travel time from a customer to all the others. Feature VII is computed as follows:

$$\frac{\sum_{i \in C} t_{ic}}{\sum_{i \in C} \tau_{ic}} \tag{2}$$

The idea is that a customer with a high value of this ratio is a promising candidate to be served by the drone.

3.2.2. Feature IX

It highlights the centrality of the customer in the network by measuring its weight into the determination of the *barycenter*. Its expression is the following:

$$\frac{Eucd(g(C \setminus \{c\}), g(C))}{\sum_{i,j \in C} Eucd(i,j)/|C|^2}, \tag{3}$$

where the operator $Eucd(a, b)$ indicates the Euclidean distance between a and b. The numerator expresses the distance between the *barycenter* computed with and without the customer c. The lower the numerator, the more central the position of the customers. Therefore, it is more likely to be served by the truck. On the other hand, the denominator expresses the average euclidean distance between all the customers. It is used to normalize the feature value.

3.2.3. Feature X

It indicates the centrality of the customer, similar to Feature IX. However, it expresses the distance between the customer itself and the *barycenter* compared to the other customers. It is computed as follows:

$$\frac{Eucd(c, g(C))}{\sum_{i \in C \setminus \{c\}} Eucd(c,i)/(|C|-1)} \tag{4}$$

The numerator is the euclidean distance between the customer and the *barycenter*. The denominator, instead, is the average distance between all the other customers and the *barycenter*. The higher the value of this feature, the more peripheral the customer is compared to the others, where a peripheral customer is a promising *ED* customer.

3.2.4. Feature XI

This feature is based on the idea of reachability. It counts the customers reachable within the drone endurance. Its expression is the following:

$$\frac{|R(c, Dtl)|}{|C|} \tag{5}$$

The numerator is the number of customers reachable within the endurance. The denominator is used to normalize the value of the feature. The idea is that customers characterized by a high value of this feature are unlikely to be served by the drone since there are few customers reachable within the drone endurance.

3.2.5. Feature XII

This feature indicates the reachability of a customer due to the drone endurance weighted for the related distance from the other customers. It is computed through the following mathematical expression:

$$\frac{\sum_{i \in R(c,Dtl)} \tau_{ic}/|R(c, Dtl)|}{\sum_{i,j \in C} \tau_{ij}/|C|^2} \tag{6}$$

This feature compares the average distance between the customer c and the other customers within the endurance threshold with the average distance of the network. If the feature value is low, the customer is closer to customers reachable within the drone endurance than the other customers. Therefore, it could be convenient to serve it with the drone.

3.2.6. Features XIII and XIV

These two features are based on the same idea behind Features XI and XII, but instead of considering the Dtl as a threshold, we consider the average drone euclidean travel time of the customer c from all the others $\tau_{avg}(c), \tau_{avg}(c) = \sum_{i \in C} \tau_{ic}/|C|$. On this basis, the two features can be computed as follows:

$$\frac{|R(c, \tau_{avg}(c))|}{|C|}, \tag{7}$$

$$\frac{\sum_{i \in R(c, \tau_{avg}(c))} \tau_{ic}/|R(c, \tau_{avg}(c))|}{\sum_{i,j \in C} \tau_{ij}/|C|^2}, \tag{8}$$

where expressions (7) and (8) correspond to Features XIII and XIV, respectively. The motivation that leads to these features arises from the observation that, in some instances, the endurance is too high with respect to the average travel time, allowing all the customers to be reached with a drone. However, as previously shown, there is a limit on the number of customers a drone can serve. Therefore, customers that are "nearer" than others will be more likely to be served by drone.

4. Customer classification experiments

This section presents and discusses the customer classification experiments and the related results. The experiments have been performed on an Intel(R) Core(TM) i7-6500U, 2.50 GHz, 16.00 GB of RAM. The classification experiments have been implemented in Python language through the use of the Scikit-learn package.

We first present the data set used for the classification experiments in the following. Then, we briefly describe the classification models considered in the experimentation. Finally, we present the customer classification experiments and discuss the related results.

4.1. Data set

The literature instances for truck-and-drone routing problems are represented by the two test beds proposed in Murray and Chu (2015), the three test beds in Agatz et al. (2018), the test bed in Poikonen, Golden, and Wasil (2019), and the test bed in de Freitas and Penna (2020). In particular, the two test beds proposed in Murray and Chu (2015) have been used specifically for the *FS-TSP*. They consist of 72 instances with 10 customers and 120 with 20 customers, respectively. Two different values of endurance (20 and 40 min) are considered for both instances, leading to 144 and 240 *FS-TSP* instances. The optimal solution of all the instances with 10 customers is known in the literature. However, the optimal solution for different instances with 20 customers is still unknown when the present study is devised.

In our experiments, we have to classify a customer in *ED* or *NED*. Therefore, each customer represents a single observation. We know the classification of a customer in the *FS-TSP* only for the instances solved to optimality. Therefore, only the instances with 10 customers could be used as training and test sets for the customer classification. On this basis, the size of our data set would be quite limited (1440 observations), being equal to the number of instances (144) multiplied by the number of customers (10). Therefore, we considered the same data set proposed in Boccia, Mancuso, Masone, Sterle (2021) for the customer classification. The data set contains 2376 instances with 10 customers. These instances were generated coherently with the 10 customer instances proposed in Murray and Chu (2015). The customers are located across a square area of 8 by 8 miles. Then, the depot location is chosen among three possibilities: the origin (i.e., setting the x- and y- coordinates to 0); the center of the customer area computed, making the average of the customer x- and y-coordinates; the bottom of the area computed making the average of the customer x-coordinates and setting the y-coordinate to 0. Two percentage values are considered for customers whose package exceeds the maximum drone payload (10% and 20%). The truck and drone travel by considering a Manhattan and a Euclidean metric, respectively. Only one speed is considered for the truck (25 miles/h). On the other hand, three possible drone speeds are considered (15, 25, and 35 miles/h). The drone endurance was chosen to be either 20 or 40 min. The launch and rendezvous setup times are both equal to one minute. For each combination of these parameters, 66 different instances are generated, leading to a total number of instances and customers/observations equal to 2376 and 23 760, respectively. Then, we compute the values of all fourteen features for each customer. We point out that all the feature values are centered and scaled based on the instance, as they are different types and orders of magnitude. Finally, we determine if a customer is *ED* or *NED* in the *FS-TSP* optimal solution by solving all the instances through the column-and-row generation algorithm proposed in Boccia et al. (2021c).

4.2. Classifiers

As discussed above, we used the classifiers implemented in the Scikit-learn package (Pedregosa et al., 2011) for our experimentation as a black box. In particular, we used the same seven classifiers considered in Boccia, Mancuso, Masone, Sterle (2021), namely: k-nearest neighbors (*KNN*), linear support vector machine (*LSV*), kernel support vector machine (*RSV*), neural networks (*NNE*), decision tree (*DET*), random forests (*RAF*), adaptive boost algorithm (*ADB*). These classifiers are widely known in literature; therefore, we provide a brief description of each of them in the following.

The *KNN* belongs to the neighbors-based classification methods (Goldberger, Hinton, Roweis, & Salakhutdinov, 2004). It is a type of instance-based learning, or lazy learning, where the model only stores the training data and does not build a general internal model. The *KNN* classifier compares the new data point to the training data and finds the k training points closest to it, according to a distance metric, to predict a new data point. The classifier then assigns the new data point to the most common class among the k-nearest neighbors. The value of k is a hyperparameter that is chosen by the user, and it determines the number of neighbors that are considered when making a prediction. The optimal choice of the k value depends largely on the data set. A large value of k reduces the noise, while a small value makes the classification boundaries less distinct.

Support vector machines (*SVM*) are classifiers based on the concept of hyperplanes. A hyperplane ideally can linearly separate the data points in a given dataset into different classes. This hyperplane is chosen to have the maximum margin, or the maximum distance, between the closest data points of each class. Once the hyperplane is determined, new data points can be easily classified by assigning them to the appropriate side of the hyperplane. The closest values to the classification margin are known as support vectors. The most commonly known *SVM* is a linear classifier (*LSV*) that can predict the membership of each input between two possible classifications using straight lines as class boundaries (Wang, 2005).

If the data are not linearly separable, within the *SVM* is used the kernel method. The idea behind the kernel method is to transform the data space in such a way that in the transformed space the data are linearly separable. One of the most popular kernel method for the *SVM* is the Radial Basis Function (RSV) (Amari & Wu, 1999).

Neural networks are computational models inspired by biological neural networks capable of approximating nonlinear functional relationships between input and output variables. A neuron is represented by a node, and a collection of nodes is referred to as a layer. Finally, the collection of interconnected layers forms the neural networks (Richard & Lippmann, 1991). The *NNE* implemented in the Scikit-learn package belongs to the class of the Multi-layer Perceptron (*MLP*) algorithms using backpropagation. An *MLP* is a supervised learning algorithm that, given a set of features (representing the leftmost layer) and a target (the output layer), can learn a nonlinear function that can be used for classification by training on a dataset. The layers between the input and the output layers are known as hidden layers. The node output is computed through a nonlinear function of its inputs. The weights represent the connections between nodes from adjacent layers in a model. The weights are modified as learning proceeds, representing the strength of the connections.

A decision tree (*DET*) is a non-parametric supervised learning method used for classification. The basic idea is to break up a complex classification into a union of several simpler classifications through a multistage approach, hoping the final solution obtained this way would resemble the intended desired solution (Friedl & Brodley, 1997). It works by constructing a tree-like model of decisions based on the input data, where each internal node represents a test on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label. The decision tree classifier is trained by dividing the input data into subsets based on the values of the attributes and recursively applying the same process to each subset until the leaf nodes are reached. This process is called recursive partitioning, and it results in a tree-like structure that can be used to make predictions on new data points.

The *RAF* are classifiers that are generated by the ensemble of randomized decision trees (Pal, 2005). The prediction of the ensemble is given as the averaged prediction of the individual classifiers. In particular, each decision tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set. Moreover, when splitting each node during the tree construction, the best split can be found considering all input features or just a random subset of them. These two sources of randomness are used to overcome the overfitting issue that generally arises using individual decision trees.

The *ADB* is the popular boosting algorithm AdaBoost, first introduced in Freund, Schapire, and Abe (1999). *ADB* belongs to the category of ensemble classifiers. It is a combination of two key ideas:

boosting and adaptive weighting. Boosting is a general method for improving the performance of a machine learning algorithm by training a series of "weak" learners, which are models that are only slightly better than random guessing, and combining their predictions to form a stronger, more accurate model. In the case of *ADB*, the weak learners are decision trees trained on different input data subsets. Adaptive weighting is a method for giving more emphasis to the data points that are harder to classify and less on the data points that are easy to classify. In *ADB*, each decision tree is trained on a weighted version of the input data. The weights are initially set to be equal for all data points. After each decision tree makes its prediction, the weights of the incorrectly classified data points are increased, and the weights of the correctly classified data points are decreased. This process is repeated for a number of iterations, and the final classification is obtained through a weighted sum (or majority vote) of the single classification of the weak learners.

4.3. Classification experiments

To assess the impact of different features on customer classification, we conduct three distinct experiments. In the first, we evaluate the impact of the new features. Subsequently, considering that a large number of features might introduce noise into the results (Tang, Alelyani, & Liu, 2014), we aim to identify a good subset of features for effective customer classification. Finally, we aim to determine a good classifier parameter configuration to enhance the performance of the classification process.

4.3.1. Feature comparison

To assess the impact of the new features on customer classification, we compared the results of the classifiers using the features proposed in Boccia, Mancuso, Masone, Sterle (2021) (Feature I–Feature VII), those introduced in this study (Feature VIII–Feature XIV), and the complete set of features (Feature I–Feature XIV). For simplicity, we use the default parameter setting for each classification model. Then, we use a 10-fold cross-validation scheme for all the experiments on the data set with the 23 760 observations and compute different classification metrics (Lever, 2016). In particular, we compute the accuracy, precision, recall, and f1-score for each subset of features and classifier. For the sake of completeness, we report for each metric the related expression:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{9}$$

$$precision = \frac{TP}{TP + FP}, \tag{10}$$

$$recall = \frac{TP}{TP + FN}, \tag{11}$$

$$f1 - score = 2 * \frac{precision * recall}{precision + recall}, \tag{12}$$

where TP, TN, FP, and FN indicate true positive, true negative, false positive, and false negative, respectively. We point out that in our experiment, a TP is a customer classified as *NED* and served by the truck in the *FS-TSP* optimal solution. A TN is a customer classified as *ED* and served by the drone in the optimal solution. An FP is a customer classified as *NED* but served by the drone in the optimal solution. Finally, an FN is a customer classified as *ED* but served by the truck in the optimal solution. The results of the customer classification are reported in Table 1.

On average, it is evident that when considering only the new features, only the *KNN* classifier improves its performance compared to the feature subset I–VII. This can be explained by considering that the new features were developed taking into account *FS-TSP* subtle aspects not covered by the previous features. Thus, they are intended to integrate rather than replace them. Indeed, we can observe that five classifiers out of seven with all the features improve their performance compared to the feature subset I–VII. In particular, we can observe that *RAF* classifier is the best classifier showing the highest accuracy, precision, and f1-score values. The *DET* classifier shows the highest value of recall. Therefore, we can confirm that, as also shown in Boccia, Mancuso, Masone, Sterle (2021), decision tree classifiers perform well in customer classification. However, using a larger number of features only sometimes leads to an improvement in the classification performance, as expected (Hua, Xiong, Lowey, Suh, & Dougherty, 2005). Indeed, the *KNN* and *RSV* show better performance with the features I–VII.

Table 1
Impact of the new features on the classifier performance.

Classifier	Score	Features I–VII	Features VIII–XIV	Features I–XIV
KNN	Accuracy	0.858	0.860	0.853
	Precision	0.680	0.694	0.678
	Recall	0.584	0.562	0.568
	f1-score	0.616	0.604	0.603
LSV	Accuracy	0.862	0.830	0.869
	Precision	0.780	0.758	0.793
	Recall	0.455	0.248	0.494
	f1-score	0.557	0.359	0.592
RSV	Accuracy	0.860	0.827	0.849
	Precision	0.716	0.658	0.708
	Recall	0.532	0.368	0.473
	f1-score	0.597	0.442	0.545
DET	Accuracy	0.865	0.829	0.869
	Precision	0.691	0.617	0.712
	Recall	0.603	0.438	0.622
	f1-score	0.635	0.493	0.648
RAF	Accuracy	0.878	0.846	0.888
	Precision	0.767	0.729	0.802
	Recall	0.598	0.438	0.612
	f1-score	0.654	0.516	0.676
NNE	Accuracy	0.861	0.825	0.863
	Precision	0.712	0.663	0.699
	Recall	0.563	0.382	0.601
	f1-score	0.612	0.448	0.634
ADB	Accuracy	0.861	0.825	0.863
	Precision	0.732	0.669	0.726
	Recall	0.571	0.369	0.566
	f1-score	0.608	0.434	0.610

4.3.2. Feature selection

Based on the previous results, we observed that simply considering all the features does not necessarily yield better results than using a subset of them. Therefore, we run an additional experiment to determine a good subset of features for customer classification. To this end, we use the sequential feature selection algorithm implemented in the Scikit-learn package (Ferri, Pudil, Hatef, & Kittler, 1994).

For completeness, we recall that sequential feature selection is a type of algorithm used to select a subset of features from a larger set of features in a dataset. It is called "sequential" because it works by iteratively adding (forward) or removing (backward) features from the subset based on some performance criterion. Forward selection starts with an empty set of features and adds one at a time. Backward selection starts with the complete set of features and removes features one at a time. The sequential feature selection algorithm implemented in the Scikit-learn package uses the *KNN* classifier as default with k equal to 4. In our experiment, we use four different setups:

- forward selection with $k = 4$ (*F-4NN*);
- backward selection with $k = 4$ (*B-4NN*);
- forward selection with k determined automatically by the algorithm (*F-kNN*);
- backward selection with k determined automatically by the algorithm (*B-kNN*).

Table 2
Classification results after the feature selection.

Classifier	Score	F-4NN	B-4NN	F-kNN	B-kNN
KNN	Accuracy	0.896	0.901	0.901	0.897
	Precision	0.759	0.777	0.779	0.760
	Recall	0.701	0.719	0.718	0.714
	f1-score	0.720	0.738	0.738	0.730
LSV	Accuracy	0.852	0.868	0.869	0.869
	Precision	0.800	0.791	0.798	0.794
	Recall	0.366	0.476	0.475	0.478
	f1-score	0.489	0.586	0.585	0.588
RSV	Accuracy	0.855	0.876	0.873	0.874
	Precision	0.761	0.759	0.756	0.755
	Recall	0.440	0.574	0.561	0.564
	f1-score	0.534	0.64	0.631	0.632
DET	Accuracy	0.857	0.869	0.875	0.871
	Precision	0.740	0.694	0.724	0.700
	Recall	0.468	0.618	0.62	0.628
	f1-score	0.558	0.645	0.657	0.654
RAF	Accuracy	0.858	0.887	0.888	0.886
	Precision	0.753	0.801	0.799	0.796
	Recall	0.469	0.598	0.606	0.598
	f1-score	0.555	0.667	0.672	0.665
NNE	Accuracy	0.854	0.872	0.872	0.872
	Precision	0.737	0.737	0.744	0.740
	Recall	0.471	0.579	0.571	0.574
	f1-score	0.550	0.637	0.634	0.633
ADB	Accuracy	0.852	0.867	0.867	0.865
	Precision	0.715	0.72	0.726	0.712
	Recall	0.465	0.589	0.587	0.579
	f1-score	0.544	0.636	0.633	0.627

Table 3
Feature determined by each selection setting.

Feature	F-4NN	B-4NN	F-kNN	B-kNN
I		✓	✓	✓
II		✓	✓	✓
III	✓	✓	✓	✓
IV				
V				
VI	✓	✓	✓	✓
VII				
VIII	✓	✓	✓	✓
IX		✓		
X			✓	✓
XI		✓		
XII			✓	
XIII				
XIV				✓

The feature selection is repeated for all the setups until the accuracy improves by at least 0.01.

The classification results after the feature selection are reported in Table 2.

The results show that all the setups determine similar improvements. However, on average, the *F-kNN* setting determines the best performance for all the classifiers. Moreover, the lowest accuracy value is equal to 13.3%, lower than the lowest accuracy determined in Boccia, Mancuso, Masone, Sterle (2021) (14.2%). We can also observe that the *KNN* classifier performs best. This result can be explained considering that *KNN* is the classifier used within the feature selection. The features determined at the end of the feature selection for each setup are reported in Table 3.

We can observe that the *F-4NN* setup selects 3 features while all the others select 7 features. Moreover, we can observe that features III, VI, and VIII are selected in all the setups. Instead, Features IV, V, and XIII are not selected by any setup. Finally, we also point out that there are both new and old features in all the setups. In particular, the best setups select at least one feature based on the *barycenter* and the customer reachability. Therefore, the classification results show that the proposed features allows to improve the *FS-TSP* customer classification process.

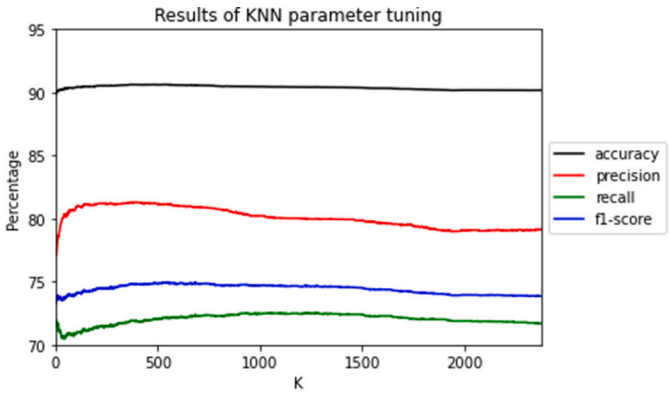

Fig. 2. Performance of KNN for different numbers of neighbors (K).

4.3.3. Classifier tuning

The previous experiments were conducted using the default parameter configurations for the classifiers, as our primary focus was evaluating the impact of features on the classification process. In this subsection, we present an experiment assessing how classification results vary with different classifier parameter configurations. For this experiment, we selected the *KNN* classifier and the features chosen by the *F-kNN* setup, as they yielded the best classification results in the previous experiment. As mentioned earlier, the main hyperparameter for the *KNN* classifier is the number of neighbors (K). Generally, a suitable value for K is the square root of the sample size; however, the optimal value depends on the specific dataset in use (Zhang, Li, Zong, Zhu, & Wang, 2017). Hence, we conduct several experiments, varying the value of K between 1 and 2376 (which represents 10% of the sample size). The results are depicted in Fig. 2, with the x-axis representing the value of K and the y-axis showing the results for the different classification metrics in percentage.

We note that the accuracy is relatively stable across different values of K, with a small variation between its minimum value of 89.88% and maximum value of 90.62%. In contrast, precision exhibits a slightly greater variability than accuracy, with values spanning from 77.07% to 81.30%. The recall, however, has a narrower range, from a minimum of 70.47% to a maximum of 72.57%. The f1-score similarly ranges between a minimum of 73.25% and a maximum of 74.94%. Concluding, it can be observed that best results align with values similar to the square root of the sample size. Moreover, as expected, performance tends to decline with values of K that are either too small or excessively large.

5. FS-TSP experiments

The results of the customer classification cannot be used to directly evaluate the impact on the *FS-TSP* in terms of the quality of the resulting solution. Indeed, the effects of the customer classification on the *FS-TSP* solution are the following:

- a TP reduces the solution space without a loss of the optimality;
- a TN has no effect on the solution space and, as a consequence, on the optimality;
- an FP reduces the solution space with a loss of optimality;
- an FN does not reduce the solution space, so there is no loss of optimality.

Therefore, it is possible to determine the *FS-TSP* optimal solution even with an accuracy lower than 1. On the other hand, a high accuracy value does not necessarily guarantee a good *FS-TSP* solution.

In this section, we present and discuss the computational results of the experimentation performed to evaluate the impact of the new features and the classification parameter tuning in terms of *FS-TSP* solution. This experimentation consists in solving *FS-TSP* instances modified using the results of the customer classification. The objective is to investigate the benefits of the new features on the *FS-TSP* solution in terms of objective function value and computation times. For this experimentation, we considered medium- and large-sized instances taken from the literature. The medium-sized instances consist of the 20-customer instances presented in Murray and Chu (2015). The large-sized instances, introduced in de Freitas and Penna (2020), are an adaptation of a subset of instances from the TSPLIB (Reinelt, 1991) for the *FS-TSP*, with the number of customers ranging from 51 to 200.

In the following two sections, we outline the experimental setup and discuss the computational results for each test bed.

5.1. FS-TSP experiments on medium-sized instances

Each customer in the *FS-TSP* instances is characterized by a binary parameter that is equal to 1 if the drone cannot serve it, 0 otherwise. This parameter indicates if the customer parcel exceeds the drone maximum payload. In our approach, we exploit this parameter to include the results of the customer classification into the *FS-TSP* solution. The literature instances considered for this experimentation consist of 120 instances with 20 customers, where each instance can be characterized by two *Dtl* values (20 and 40 min). In particular, for each instance and classification, we generate a modified instance where each customer is set as *ED* or *NED* according to the classification results. For the customer classification, we considered the *KNN* as a classifier, the one with the best performance in the previous classification experimentation. Moreover, the *KNN* is the classifier showing on average the best *FS-TSP* results in Boccia, Mancuso, Masone, Sterle (2021). Then, we select three feature subsets for classification: one that only considers the new features (referred to hereafter as *New*), one that includes all the features (referred to as *All*), and one derived from the *F-kNN* feature selection (denoted as *Sel*). For these feature subsets, the *KNN* classifier is employed with the default value of K in the Scikit-learn package, which is $K = 5$. Additionally, we introduced a classification setup based on the tuning experiments conducted in the previous section (labeled as *Tun*). For this setup, we employed the features resulting from the *F-kNN* feature selection and the *KNN* classifier set at $K = 369$, which yielded the maximum value for the accuracy metric.

On this basis, we generate two versions of each instance, one for each *Dtl*, for each subset of features and classifier parameter configuration, resulting in a total of 960 instances. Then, we solved the modified *FS-TSP* instances using the same solution method considered in Boccia, Mancuso, Masone, Sterle (2021) to ensure a consistent comparison. Specifically, we used the column-and-row generation approach proposed in Boccia et al. (2021c), with a time limit of 1 h, employing Cplex 12.7 with default setting as *MILP* solver. All experiments were carried out on an Intel(R) Core(TM) i7-6500U, 2.50 GHz, with 16.00 GB of RAM. Subsequently, we compared these results with the best-known upper bounds for these instances reported in Boccia et al. (2021c) and Dell'Amico et al. (2022). Moreover, for the sake of completeness, we reported the best results from Boccia, Mancuso, Masone, Sterle (2021) (denoted as *Lit*) where the features I–VII were first proposed and tested. We recall that these results were obtained using the *KNN* classifier with default parameter settings.

The results of the *FS-TSP* experiments on the instances with *Dtl* equal to 20 min are reported in Tables 4 and 5. Table 4 shows the results in terms of quality of the *FS-TSP* solution. The results are grouped in three sets based on the depot location (center, edge, and origin) in the instances. Therefore, each row gives an average value calculated over 40 instances. For each classification, we report the average percentage difference (*Diff%*), the percentage of solutions with an objective value equal to or better than the best known in the literature (*BES%*). We point out that the *Diff%* is computed as *Diff% = (UB-BUB)/BUB · 100*, where *UB* is the objective value of the solution determined by the proposed approach and *BUB* is the best upper bound in literature.

Table 4
Quality of the *FS-TSP* solution with *Dtl* = 20.

Features	Diff%					BES%				
	Lit	*New*	*All*	*Sel*	*Tun*	*Lit*	*New*	*All*	*Sel*	*Tun*
Center	3.13	8.34	3.86	3.04	4.11	5.00	0.00	0.00	0.00	5.00
Edge	1.48	5.61	2.29	1.58	1.99	12.50	0.00	12.50	10.00	12.50
Origin	2.00	3.81	2.50	2.02	2.45	0.00	0.00	2.50	2.50	0.00
Average	2.20	5.92	2.88	2.21	2.85	5.83	0.00	5.00	4.17	5.83

Table 5
Running time on the instances with *Dtl* = 20.

Features	Running time					TL%				
	Lit	*New*	*All*	*Sel*	*Tun*	*Lit*	*New*	*All*	*Sel*	*Tun*
Center	26.38	1.00	21.26	156.73	101.42	0.00	0.00	0.00	2.50	2.50
Edge	151.48	1.67	139.08	258.18	135.21	2.50	0.00	2.50	2.50	2.50
Origin	5.15	1.28	10.30	11.10	7.09	0.00	0.00	0.00	0.00	0.00
Average	61.00	1.32	56.88	142.00	81.24	0.83	0.00	0.83	1.67	1.67

The results show that by considering the *New* classification we obtain the largest *Diff%*. This confirms that these features, due to their nature, have to be used in combination with other features. However, we recall that in terms of classification, the *New* features showed better results than both *Lit* and *All*. Additionally, we can observe that the *Sel* and *Lit* classifications have a similar impact on the *FS-TSP* solution with an average *Diff%* of about 2.20. However, the *Lit* classification performs better on the edge group of instances. In contrast, the *Sel* classification performs better on the center instances. These results can be explained by considering that new features, mainly based on the instance topography, allow to take into account elements not considered by the *Lit* features. Subsequently, we can observe that by only considering all the features, i.e. the *All* classification, does not yield better results, since it shows an average *Diff%* of 2.88. Indeed, as for the customer classification, a large feature subset might introduce noise in the classification and, as a result, in the *FS-TSP* solution. Finally, it is worth noting that even when using the same subset of features, adjusting the classifier parameters can lead to different results in terms of *Diff%*. Specifically, the *Tun* classification shows a slightly higher *Diff%* compared to the *Sel* one.

Concerning the *BES%*, we can observe that the *Lit* and *Tun* classifications show slightly better results than the other classifications. However, the *All* and the *Sel* classifications perform better on the origin group instances. Moreover, unlike for the *Diff%*, the *All* classification performs better than the *Sel* one since the first is able to determine the best solution on the 5% of the instances versus the 4.17% of the latter. These results show that the new features used in combination with other features can improve the *FS-TSP* quality obtainable with the proposed approach.

Table 5 shows the running time results. In particular, we report the average running time in seconds and the percentage of modified instances not solved to optimality within the time limit (*TL%*).

Concerning the running time results, the *New* classification is the fastest, while the *Sel* classification is the slowest. In particular, we note that the average running time for the *New* classification is at least an order of magnitude less than that of the others, partially justifying its higher *Diff%*. On the other hand, in terms of instances solved within the time limit, all the classifications show similar results. Only, the *Sel* and *Tun* classifications are slightly worse, being unable to solve just one instance of the center group within the time limit.

Tables 6 and 7 report the results on the instances with *Dtl* equal to 40 in terms of quality of the *FS-TSP* solution and running time, respectively.

Table 6
Quality of the *FS-TSP* solution with $Dtl = 40$.

Features	Diff%					BES%				
	Lit	*New*	*All*	*Sel*	*Tun*	*Lit*	*New*	*All*	*Sel*	*Tun*
Center	2.34	14.13	4.76	4.45	4.38	15.00	0.00	17.50	17.50	12.50
Edge	3.86	10.33	1.77	1.74	1.85	20.00	0.00	37.50	30.00	30.00
Origin	1.87	13.56	4.18	3.46	1.84	25.00	2.50	22.50	17.50	30.00
Average	2.69	12.67	3.57	3.22	2.68	20.00	0.83	25.83	21.67	24.17

The results on the instances with *Dtl* equal to 40 show that, on average, the *Tun* classification performs slightly better than the others in terms of *Diff%*. Moreover, we can observe that on the edge group of instances, the *All* and the *Sel* classifications show the lowest *Diff%*. These results show the effectiveness of the new features of taking into account some significant aspects for the *FS-TSP* solution quality, not considered by the feature *Lit*. The results confirm these impacts also in terms of *BES%*. Indeed, the *Tun* classification exhibits the highest percentage of best solutions for the origin instance group, the *Sel* for the center instance group, and the *All* for both the center and edge instance groups. In particular, the *All* classification determines, on average, the highest *BES%* (25.83).

Regarding the running time, on average, the *New* classification outperforms the other classifications. However, the corresponding *Diff%* suggests a trade-off between running time and the *Diff%* when employing the *New* classification. Instead, the *Tun* classification, which shows the smallest average *Diff%* on these instances, also maintains a reasonable average running time when compared to the *Lit*, *All*, and *Sel* classifications. These trends are further confirmed when looking at the results in terms of *TL%*.

In conclusion, the overall results highlight the substantial impact of the proposed features on the *FS-TSP* solution. Furthermore, we note that, by utilizing feature selection and classifier parameter tuning, the presented method can effectively tackle *FS-TSP* instances. However, as previously discussed, there is not clear evidence pointing to a particular subset of features or a parameter setting that universally outperforms the others across all instances. Therefore, a further investigation of these aspects is necessary.

5.2. FS-TSP experiments on large-sized instances

In this section, we evaluate the performance of the proposed approach on large-sized instances. The literature instances considered for this experimentation consists of 24 instances with up to 200 customers, with each instance having a *Dtl* of 40 min. To address these *FS-TSP* instances, we employed the same approach used in the experimentation on medium-sized instances. However, we only considered the *Tun* classification, as the experimentation on medium-sized instances showed that it yielded better results compared to the others.

Therefore, for each instance, we generate a modified version where each customer is designated as *ED* or *NED* based on the results of the *Tun* classification. We then addressed the modified *FS-TSP* instances using the column-and-row generation approach proposed in Boccia et al. (2021c), setting a time limit of 1 h and using Cplex 12.7 with its default settings as *MILP* solver. All experiments were conducted on an Intel(R) Core(TM) i7-6500U, 2.50 GHz, equipped with 16.00 GB of RAM.

Subsequently, we compared these results with the upper bound generated by the column-and-row generation technique proposed in Boccia et al. (2021c), without using the classification approach, and with the best-known upper bounds for these instances reported in Dell'Amico et al. (2021a). The first comparison shows the impact of the classification process on both the quality and the computational time of the *FS-TSP* solution method. The second comparison aims to highlight the effectiveness of the proposed methodology by comparing it with the best heuristic for the *FS-TSP* available in the literature, to the best of our knowledge. Lastly, it should be noted that we used the solution method from Boccia et al. (2021c) to maintain consistency with the previous experimentation. However, we point out that we could have opted for any *FS-TSP* solution method that is capable of exploiting the outcomes of the classification process.

Table 8 presents the results of our experimentation. The first column lists the instance name. The successive two columns report the upper bounds obtained from the solution methods proposed in Boccia et al. (2021c) and Dell'Amico et al. (2021a), denoted as BMS21 and DMN21, respectively. The third and fourth columns provide the *Diff%* between the proposed method and both BMS21 and DMN21. Specifically, *Diff%* is computed as $Diff\% = (UB_{ML}\text{-}UB_X)/max\{UB_{ML},UB_X\}$, where UB_{ML} represents the upper bound determined by the proposed approach, and UB_X is the upper bound obtained through the solution method X, with X being either BMS21 or DMN21. Then, the final two columns indicate the running times for DMN21 and our proposed method. We have chosen not to include the running time for BMS21 as, for every instance, the solution method was stopped due to the 1-h time limit.

Observing the results, we notice that the proposed approach enhances the upper bound determined by BMS21 for nearly all instances. Only in two cases does the proposed method yield the same upper bound as BMS21. However, it is worth noting that our approach never yields an upper bound greater than that determined by BMS21 across any instances. This indicates that the proposed method improves the performance of BMS21 for large-sized instances, which the exact solution method struggles to tackle due to the high dimensionality. These results highlight the potential of the proposed hybrid approach to enhance the effectiveness of a combinatorial optimization solution method.

When comparing our method with DMN21, the average Diff% is equal to 15.40. Nonetheless, this value is biased by instances not solved within the time limit. By excluding those instances, the Diff% drops to 4.12. Additionally, we can observe that for 5 instances, the proposed approach is able to determine the best upper bound known in literature for those instances. In terms of execution time, our approach on average runs just about an order of magnitude longer than DMN21. However, in one instance, it is even faster than DMN21. In conclusion, these results underscore that the proposed approach can also be suitable employed as a heuristic method for solving the *FS-TSP*.

6. Conclusion

In this work, we study the benefit arising from the use of machine learning techniques for combinatorial optimization problems. In particular, we investigate the impact of a customer classification into *ED* and *NED* on a solution of a truck-and-drone routing problem known in the literature as *FS-TSP*. The idea is to use this customer classification to reduce the solution space of *FS-TSP* and thus improve the performance of the existing solution method. Unlike the previous study, we have developed original customer features based mainly on graph topography and have incorporated both feature selection and classifier parameter tuning into the hybrid solution approach. We have validated the impact of these new features through twofold experimentation. First, we evaluate the effect of the new features and the classifier parameter setting on customer classification. To this end, different classification methods, feature subsets, and classifier parameter configurations are tested. The results show that the new features can improve the performance of all the tested classification methods. In particular, a subset of features consisting of new and old features results shows the best results. Then, we investigate the impact of the resulting classification on the *FS-TSP* solution in terms of the quality of the solution and running times. On the one hand, the results confirm that the new features can improve the performance of a combinatorial optimization method for the *FS-TSP* across various benchmark instances, and that this hybrid approach can also be effectively employed as a heuristic method.

Table 7
Running time with $Dtl = 40$.

Features	Running time					TL%				
	Lit	*New*	*All*	*Sel*	*Tun*	*Lit*	*New*	*All*	*Sel*	*Tun*
Center	692.26	1.15	777.80	1110.40	711.43	15.00	0.00	15.00	22.50	15.00
Edge	774.04	1.02	1898.86	1061.93	609.93	20.00	0.00	27.50	27.50	12.50
Origin	1018.28	1.24	816.48	955.14	730.10	25.00	0.00	15.00	20.00	15.00
Average	2484.58	1.14	3493.14	3127.47	683.82	20.00	0.00	19.17	23.33	14.17

Table 8
Results on the large-sized instances.

Instances	Upper bound		Diff%		Running time	
	BMS21	DMN21	BMS21	DMN21	DMN21	Tun
berlin52	746.25	199.75	−72.03	4.28	2.90	1836.97
bier127	12 806.40	3505.40	−71.38	4.37	569.60	780.34
ch130	1508.49	184.52	−78.43	43.29	387.40	3600
d198	838.44	461.23	0.00	44.99	482.60	3600
eil51	43.80	13.45	−69.29	0.00	0.00	57.92
eil76	61.30	16.90	−72.43	0.00	0.00	137.50
kroA100	5928.00	540.81	−89.91	9.59	225.40	393.92
kroA150	8970.75	717.44	−91.60	4.76	546.00	3600
kroA200	11 528.90	832.10	−80.48	63.02	139.80	3600
kroB150	8464.25	694.06	−90.26	15.84	670.20	3600
kroB200	10 328.90	813.49	−82.91	53.92	153.00	3600
kroC100	5706.70	564.28	−88.77	11.93	519.20	475.19
kroD100	4482.75	560.14	−86.21	9.41	674.20	368.66
kroE100	5734.20	589.02	−88.97	6.85	211.00	364.99
lin105	1069.20	387.62	−48.97	28.96	217.70	3600
pr107	1743.57	1044.11	−34.70	8.29	249.00	1014.35
pr124	2361.20	1620.23	−30.26	1.61	8.70	1030.98
pr136	7736.80	2525.62	−64.00	9.32	234.6	3600
pr144	2628.61	1676.75	−36.21	0.00	28.8	598.35
pr152	4493.50	1981.94	−55.20	1.54	10.2	1540.34
rat99	76.10	37.45	−50.79	0.00	0.00	390.52
rat195	126.90	71.50	0.00	43.66	0.00	3600
rd100	1605.44	221.53	−85.64	3.92	163.00	3590.75
st70	105.45	21.00	−80.09	0.00	0.00	249.53
Average			−64.52	15.40	228.90	1884.60

On the other hand, the results show that there is not a straight relationship between classification and combinatorial optimization results. Future extensions of this study could consider using combinatorial optimization methods to select the subset of features for the customer classification (Boccia, Masone, Sforza, & Sterle, 2017). Moreover, new features can be developed to account for other *FS-TSP* aspects that are still unexplored. Finally, it would be interesting to investigate the use of similar hybrid methods based on machine learning techniques in combination with other approaches to evaluate the resulting benefits in different application fields (Capocasale, Gotta, & Perboli, 2023; Capocasale & Perboli, 2022).

CRediT authorship contribution statement

Maurizio Boccia: Conceptualization, Methodology, Investigation, Software, Writing – original draft, Writing – review & editing, Software, Visualization, Validation, Data curation, Supervision. **Andrea Mancuso:** Conceptualization, Methodology, Investigation, Software, Writing – original draft, Writing – review & editing, Software, Visualization, Validation, Data curation, Supervision. **Adriano Masone:** Conceptualization, Methodology, Investigation, Software, Writing – original draft, Writing – review & editing, Software, Visualization, Validation, Data curation, Supervision. **Teresa Murino:** Conceptualization, Methodology, Investigation, Software, Writing – original draft, Writing – review & editing, Software, Visualization, Validation, Data curation, Supervision. **Claudio Sterle:** Conceptualization, Methodology, Investigation, Software, Writing – original draft, Writing – review & editing, Visualization, Validation, Data curation, Supervision.

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.

Acknowledgments

Research presented in the paper has been partially funded by the National Centre for Sustainable Mobility “CN MOST” (MUR code CN00000023 - CUP UNINA code: E63C22000930007) and by the MUR research programs PRIN 2022 founded by the European Union – Next Generation EU (Project “ACHILLES, eco-sustAinable effiCient tecHdrIven Last miLE logiStics”, Prot. 2022BC2CW7, CUP: E53D23005630006) and PRIN 2022 PNRR founded by the European Union – Next Generation EU (Project “COSMO, Co-Opetitive Sustainable Mobility Optimization”, Prot. P20224YN23 CUP: F53D23009980001). Lastly, the authors wish to express their appreciation to Prof. Roberto Montemanni for the valuable discussions concerning the instances derived from the TSPLIB.

References

Agatz, N., Bouman, P., & Schmidt, M. (2018). Optimization approaches for the traveling salesman problem with drone. *Transportation Science*, *52*(4), 965–981.

Amari, S.-i., & Wu, S. (1999). Improving support vector machine classifiers by modifying kernel functions. *Neural Networks*, *12*(6), 783–789.

Bengio, Y., Lodi, A., & Prouvost, A. (2021). Machine learning for combinatorial optimization: a methodological tour d’horizon. *European Journal of Operational Research*, *290*(2), 405–421.

Boccia, M., Mancuso, A., Masone, A., Sforza, A., & Sterle, C. (2021). A two-echelon truck-and-drone distribution system: Formulation and heuristic approach. In *Optimization and decision science: ODS, virtual conference, November 19, 2020* (pp. 153–163). Springer.

Boccia, M., Mancuso, A., Masone, A., & Sterle, C. (2021). A feature based solution approach for the flying sidekick traveling salesman problem. In *Mathematical optimization theory and operations research: Recent trends: 20th International conference, MOTOR 2021, Irkutsk, Russia, July 5–10, 2021, Revised selected papers 20* (pp. 131–146). Springer.

Boccia, M., Mancuso, A., Masone, A., & Sterle, C. (2023). A new MILP formulation for the flying sidekick traveling salesman problem. *Networks*, *82*(3), 254–276.

Boccia, M., Masone, A., Sforza, A., & Sterle, C. (2017). A partitioning based heuristic for a variant of the simple pattern minimality problem. In *Optimization and decision science: Methodologies and applications: ODS, Sorrento, Italy, September 4-7, 2017 47* (pp. 93–102). Springer.

Boccia, M., Masone, A., Sforza, A., & Sterle, C. (2021c). A column-and-row generation approach for the flying sidekick travelling salesman problem. *Transportation Research Part C (Emerging Technologies)*, *124*, Article 102913.

Boccia, M., Masone, A., Sforza, A., & Sterle, C. (2021d). An exact approach for a variant of the FS-TSP. *Transportation Research Procedia*, *52*, 51–58.

Capocasale, V., Gotta, D., & Perboli, G. (2023). Comparative analysis of permissioned blockchain frameworks for industrial applications. *Blockchain: Research and Applications*, *4*(1), Article 100113.

Capocasale, V., & Perboli, G. (2022). Standardizing smart contracts. *IEEE Access*, *10*, 91203–91212.

Chang, Y. S., & Lee, H. J. (2018). Optimal delivery routing with wider drone-delivery areas along a shorter truck-route. *Expert Systems with Applications*, *104*, 307–317.

Chung, S. H., Sah, B., & Lee, J. (2020). Optimization for drone and drone-truck combined operations: A review of the state of the art and future directions. *Computers & Operations Research*, *123*, Article 105004.

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.

de Holanda Maia, M. R., Plastino, A., & Penna, P. H. V. (2018). Hybrid data mining heuristics for the heterogeneous fleet vehicle routing problem. *RAIRO-Operations Research*, *52*(3), 661–690.

de Holanda Maia, M. R., Plastino, A., & Penna, P. H. V. (2020). MineReduce: An approach based on data mining for problem size reduction. *Computers & Operations Research*, *122*, Article 104995.

Dell'Amico, M., Montemanni, R., & Novellani, S. (2021a). Algorithms based on branch and bound for the flying sidekick traveling salesman problem. *Omega*, *104*, Article 102493.

Dell'Amico, M., Montemanni, R., & Novellani, S. (2021b). A random restart local search matheuristic for the flying sidekick traveling salesman problem. In *Proceedings of the 2021 8th international conference on industrial engineering and applications (Europe)* (pp. 205–209).

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.

Ferri, F. J., Pudil, P., Hatef, M., & Kittler, J. (1994). Comparative study of techniques for large-scale feature selection. *volume 16*, In *Machine intelligence and pattern recognition* (pp. 403–413). Elsevier.

Freitas, J. C., Penna, P. H. V., & Toffolo, T. A. (2023). Exact and heuristic approaches to truck–drone delivery problems. *EURO Journal on Transportation and Logistics*, *12*, Article 100094.

Freund, Y., Schapire, R., & Abe, N. (1999). A short introduction to boosting. *Journal-Japanese Society for Artificial Intelligence*, *14*(771–780), 1612.

Friedl, M. A., & Brodley, C. E. (1997). Decision tree classification of land cover from remotely sensed data. *Remote Sensing of Environment*, *61*(3), 399–409.

Goldberger, J., Hinton, G. E., Roweis, S., & Salakhutdinov, R. R. (2004). Neighbourhood components analysis. *vol. 17*, In *Advances in neural information processing systems*.

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.

Hua, J., Xiong, Z., Lowey, J., Suh, E., & Dougherty, E. R. (2005). Optimal number of features as a function of sample size for various classification rules. *Bioinformatics*, *21*(8), 1509–1515.

Karimi-Mamaghan, M., Mohammadi, M., Meyer, P., Karimi-Mamaghan, A. M., & Talbi, E.-G. (2022). Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. *European Journal of Operational Research*, *296*(2), 393–422.

Kuo, R., Lu, S.-H., Lai, P.-Y., & Mara, S. T. W. (2022). Vehicle routing problem with drones considering time windows. *Expert Systems with Applications*, *191*, Article 116264.

Lever, J. (2016). Classification evaluation: It is important to understand both what a classification metric expresses and what it hides. *Nature Methods*, *13*(8), 603–605.

Madani, B., & Ndiaye, M. (2022). Hybrid truck-drone delivery systems: A systematic literature review. *IEEE Access*.

Mara, S. T. W., Rifai, A. P., & Sopha, B. M. (2022). An adaptive large neighborhood search heuristic for the flying sidekick traveling salesman problem with multiple drops. *Expert Systems with Applications*, *205*, Article 117647.

Martins, D., Vianna, G. M., Rosseti, I., Martins, S. L., & Plastino, A. (2018). Making a state-of-the-art heuristic faster with data mining. *Annals of Operations Research*, *263*(1–2), 141–162.

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*, *80*(2), 193–215.

Mazyavkina, N., Sviridov, S., Ivanov, S., & Burnaev, E. (2021). Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, *134*, Article 105400.

Moshref-Javadi, M., & Winkenbach, M. (2021). Applications and Research avenues for drone-based models in logistics: A classification and review. *Expert Systems with Applications*, *177*, Article 114854.

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.

Pal, M. (2005). Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, *26*(1), 217–222.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, *12*, 2825–2830.

Poikonen, S., Golden, B., & Wasil, E. A. (2019). A branch-and-bound approach to the traveling salesman problem with a drone. *INFORMS Journal on Computing*, *31*(2), 335–346.

Reinelt, G. (1991). TSPLIB—A traveling salesman problem library. *ORSA Journal on Computing*, *3*(4), 376–384.

Richard, M. D., & Lippmann, R. P. (1991). Neural network classifiers estimate Bayesian a posteriori probabilities. *Neural Computation*, *3*(4), 461–483.

Roberti, R., & Ruthmair, M. (2021). Exact methods for the traveling salesman problem with drone. *Transportation Science*, *55*(2), 315–335.

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.

Sacramento, D., Pisinger, D., & Ropke, S. (2019). An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones. *Transportation Research Part C (Emerging Technologies)*, *102*, 289–315.

Salama, M. R., & Srinivas, S. (2022). Collaborative truck multi-drone routing and scheduling problem: Package delivery with flexible launch and recovery sites. *Transportation Research Part E: Logistics and Transportation Review*, *164*, Article 102788.

Schermer, D., Moeini, M., & Wendt, O. (2020). A b ranch-and-cut approach and alternative formulations for the traveling salesman problem with drone. *Networks*, *76*(2), 164–186.

Tamke, F., & Buscher, U. (2023). The vehicle routing problem with drones and drone speed selection. *Computers & Operations Research*, *152*, Article 106112.

Tang, J., Alelyani, S., & Liu, H. (2014). Feature selection for classification: A review. In *Data classification: Algorithms and applications* (p. 37). CRC Press.

Tiniç, G. O., Karasan, O. E., Kara, B. Y., Campbell, J. F., & Ozel, A. (2023). Exact solution approaches for the minimum total cost traveling salesman problem with multiple drones. *Transportation Research Part B: Methodological*, *168*, 81–123.

Wang, L. (2005). *Support vector machines: Theory and applications*: *volume 177*, Springer Science & Business Media.

Yu, V. F., Lin, S.-W., Jodiawan, P., & Lai, Y.-C. (2023). Solving the flying sidekick traveling salesman problem by a simulated annealing heuristic. *Mathematics*, *11*(20), 4305.

Zhang, S., Li, X., Zong, M., Zhu, X., & Wang, R. (2017). Efficient kNN classification with different numbers of nearest neighbors. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(5), 1774–1785.