On the Evaluation of a Drone-Based Delivery System on a Mixed Euclidean-Manhattan Grid

Francesco Betti Sorbelli, *Member, IEEE*, Cristina M. Pinotti, *Senior Member, IEEE*, and Giulio Rigoni

***Abstract*— In this work, we investigate the use of drones in a delivery scenario formed by two contiguous areas. In one area the drones can freely fly on straight lines between any two locations (Euclidean metric), while in the other one the drones must follow the open space above the roads (Manhattan metric). We model this delivery scenario as a Euclidean-Manhattan-Grid (EM-grid). Given a set of customers to be served in an EM-grid, the objective is to find the distribution point (DP) for the drone that minimizes the overall traveled distance, considering that the drone has to do multiple round trips to/from the DP. In our view, the DP is optimized with respect to the set of customers and its computation must be light because it needs to be recomputed every time the set of customers varies. Accordingly, we define the Single Distribution Point Problem (SDPP) and devise sub-optimal time-efficient algorithms for solving it. We numerically compare the cost of our sub-optimal solutions with that of an optimal solution computed with a brute-force approach. Finally, using the BlueSky open air simulator, we compare the cost of our best solution with the cost of a solution that serves the costumers from a fixed DP, like the location of a delivery company's depot. The fixed DP can perform very poorly for some customer instances, while our solution is highly adaptive and reduces the time and the distance covered by the drone.**

***Index Terms*— Drone, delivery, Euclidean Manhattan grids, distribution point, centroid, BlueSky.**

I. Introduction

IN THE last period, unmanned aerial vehicles (UAVs) such as drones, are extensively and widely used in many civilian applications [1]–[6]. In fact, due to their ability to perform very challenging tasks, drones are employed for localizing missing people [7]–[9] or managing the search and rescue operations [10]–[12] after disastrous events, for offloading tasks at the edge when monitoring points of interest [13], [14], or for delivering small packages in a last-mile delivery scenario [15]–[21]. In a delivery scenario, Amazon or other important big companies can efficiently and effectively use drones for many reasons, like extending their business increasing so the number of customers to be served, or for assisting customers located at hard-to-reach places which would be impossible for ground vehicles (trucks) to visit them in a timely manner.

In the current truck-based delivery system, a single ground vehicle can accomplish, on average, 110–120 deliveries in a single day in an urban area, and 40–50 deliveries in a mixed-urban area [22]. Trucks can be stuck in the traffic jam and can delay the deliveries, and in the worst case, they could even cancel many of them. Drones, instead, have the ability to fly avoiding the traffic, and in suburban areas they can also minimize the distance to travel by short-cutting the routes. Also, a recent paper [23] has proven that drones are a mean of transportation socially well accepted and considered useful. However, drones have peculiarities [24]. First of all, their payload is bounded in weight and size, and so at the moment, they cannot perform more than a single delivery at a time. In fact, they must go back to the *distribution point* (DP), often the depot (warehouse), for loading on-board the package to be delivered to the next customer. Another constraint is that the drones cannot fly beyond certain altitudes by law. In most of the countries, this limit is fixed to 120 meters [25], [26].

In this work, we focus on a drone-based delivery scenario. Given that a drone has to deliver one package at a time, it is crucial to define a suitable starting point for any subset of deliveries. Usually, in a rural area formed by low buildings and houses, a drone can easily fly on straight lines between any two locations. On the other hand, in a downtown area of a big city formed by tall buildings and possibly skyscrapers, the drone's flight is more constrained due to the presence of obstacles. Considering that a drone cannot fly beyond the maximum altitude, inside a city a drone can only fly over the roads/streets. We summarize these two different contexts as follows: in the rural scenario, the drone moves freely according to the Euclidean metric, while in the urban scenario, the drone moves according to the Manhattan metric [27]. We model this delivery area as a Euclidean-Manhattan-Grid (EM-grid), where customers reside on both the areas.

Today, one of the biggest brakes on the development of deliveries with small drones is the impossibility, related to the payload, of shipping more than one package at a time. So, shipping a parcel requires starting from the depot, usually built outside the city, with a fully charged battery, and returning back to the depot to recharge or swap the battery before the next delivery. To reduce the distance traversed for each delivery, and to allow multiple deliveries with the same

Manuscript received 29 December 2021; revised 8 June 2022; accepted 6 July 2022. Date of publication 18 July 2022; date of current version 26 January 2023. This work was supported in part by the "GNCS—INdAM" by the "HALY-ID" Project funded by the European Union's Horizon 2020 through ICT-AGRI-FOOD under Grant 862665 and Grant 862671, and in part by MIPAAF. The Associate Editor for this article was D. S. M. Ghadiri. *(Corresponding author: Francesco Betti Sorbelli.)*

Francesco Betti Sorbelli and Cristina M. Pinotti are with the Department of Computer Science and Mathematics, University of Perugia, 06123 Perugia, Italy (e-mail: francesco.bettisorbelli@unipg.it).

Giulio Rigoni is with the Department of Computer Science and Mathematics, University of Florence, 50134 Florence, Italy.

Digital Object Identifier 10.1109/TITS.2022.3189948

battery, we envision a truck-assisted drone-based delivery system where a truck carries, up to a suitably selected DP, the drone along with a *mobile pod* which stores all the packages to be delivered by the drone. This mobile pod, once it is at the DP, is inaccessible to other people for safety reasons. The drone uses the top of the pod as starting point to deliver the packages to the customers. Once the last delivery is done, the drone stops at the pod, and the pod is then ready to be picked-up by the truck.

Now, given a subset of customers inside an EM-grid to be served by a single drone, the objective is to find the DP (i.e., the pod's position) that minimizes the sum of distances between all the customers and the DP. Recall that the drone has to perform multiple round trips to/from the DP.

The contributions of this work are summarized as follows.

- We present the EM-grid model which characterizes the delivery area for the drone.
- We define the Single Distribution Point Problem (SDPP), and devise time-efficient algorithms for computing the DP. Our solution is optimized, with respect to the set of customers, and computationally light because it needs to be recomputed every time the set varies.
- We extensively and comprehensively compare the performance of our presented algorithms.
- Using a simulator called BlueSky, we compare the performance of our best solution with that of fixed DPs, emulating the depot's location of a delivery company.

The rest of the work is organized as follows. Section II reviews the relevant related work. Section III formally defines the SDPP and shows the optimal solution. Section IV presents time-efficient algorithms for solving SDPP. Section V evaluates the effectiveness of our algorithms, and compares our solution with the fixed solution through a simulator. Finally, Section VI offers conclusions and future research directions.

II. Related Work

In the literature, there are many works solving the last-mile delivery problem with drones. In the seminal work [28], the authors study the cooperation between one truck and one drone to deliver packages to customers. The authors introduced the flying sidekicks traveling salesman problem (FSTSP), a variant of the TSP, in which a drone first takes off from the truck, then proceeds to deliver goods to a customer, and finally rejoins the truck in a third location. As the drone flies, the truck makes deliveries to other customers, but has to stop waiting for the drone at the rendezvous location. An optimal mixed-integer linear programming (MILP) plus two heuristic solutions to solve problems of practical sizes are proposed. The authors in [29], instead, tackle the problem of solving the traveling salesman problem with a drone (TSP-D), by assuming a drone that rides on a truck. A branch and bound technique is applied for solving the TSP-D. Both works pose the focus on combining the truck's and the drone's movements. They identify the best route (i.e., the best order) to sequentially deliver all the goods, thus considering the truck's movements and waiting time in the process. Instead, we assume that in a pre-processing phase the deliveries are assigned to the drone and we focus on selecting the DP that minimizes the total distance traveled by the drone. The truck brings the drone at the DP and the drone will autonomously operate.

The Euclidean and Manhattan metrics have been considered in [30] to evaluate distribution systems. The delivery area is modeled as a circular region with a central warehouse and customers randomly distributed throughout the region. In the area, 13 optimization algorithms are simulated. The comparison in terms of time and traveled distance, measured with both Euclidean and Manhattan metrics, shows that different algorithms perform better in different situations, therefore authors recommend employing different algorithms depending on the neighborhood. Similarly to our work, the authors pose the focus on identifying the best algorithm based on the customers' distribution. We move a step further, by introducing a mixed Euclidean and Manhattan area, and then, selecting the best algorithm for the distribution of the customers.

The importance of a strategic planning on urban delivery services is also studied in [31]. Specifically, the preferred method and local impacts of vehicle trip may vary by neighborhood characteristics (e.g., traffic or customer demands). Instead of searching for an optimal route, the authors focus on the estimation of the vehicles miles traveled (VMT) per order, considering different types of neighborhoods, delivery scenarios, and strategies. The system is evaluated in Chicago, showing that alternative delivery strategies can largely reduce the VMT per order based on the type of neighborhood.

Although these last two papers do not consider drones, they impact our work because they underline the importance of planning different delivery strategies depending on the delivery scenario. They also suggest that distance is one of the important parameters for evaluating a delivery system.

In our work, we focus on a delivery area called EM-grid firstly introduced in [27] that is formed by two contiguous areas, i.e., rural and urban. In the former area, the drone connects two locations by following the unique line that passes through them, while in the latter area, due to the tall buildings and altitude constraints, the drone flies over the roads. Simplifying, in the rural area, the drones' movements are modeled by the Euclidean metric, while in the urban one by the Manhattan metric. In [27], the drone's mission consists of performing one delivery for each destination of the grid. The authors envisioned the drone like the mailman in days gone. That is, the authors expect that the drone will drop off something every day in each house. In that scenario, considering that the drone has to go back to the depot after each delivery and has to serve all the points of the grid, the problem is to find the location for the depot that minimizes the sum of the traveled distances. The exact solution of the problem with a delivery for each grid point is computed in logarithmic time in the length of the side of the EM-grid [27]. In this paper, we make the more realistic assumption that only an arbitrary number of customers in EM-grid grid, i.e., not necessarily all the points of the grid are customers to be served, must be served. We borrow this assumption from [32] where a drone substitutes a shopping cart inside a warehouse.

In [19], authors consider the problem of delivering goods using a drone when environmental factors are present,

specifically the wind. They propose a framework based on time-dependent cost graphs to model the problem considering that a drone can serve a single customer before returning to the warehouse. The delivery area map is modeled with a weighted graph whose costs are time-dependent (i.e., the wind can change affecting the energy consumption of the drone), and fixed topology. To solve the problem, three approaches are proposed, i.e., offline, online, and greedy online. The performance is evaluated in terms of flight missions successfully accomplished with a given battery energy budget. Differently from this paper, not all the deliveries can be performed due to the presence of the wind that can increase the drone's energy consumption. Therefore, differently from us, the objective is to maximize the number of deliveries.

Focusing on computation time and exploiting the GPU parallel computation, the work in [20] proposes a centralized system for computing energy-efficient time-varying routes in a multi-depot multi-drone delivery system. The authors devise a centralized parallel algorithm that constantly updates the drones' delivery routes avoiding the whole recomputation from scratch, becoming 4.5× faster than the state-of-the-art algorithms. Like in our model, the pursued goal is to minimize the total traveled distance, but the drones can only fly according to the Euclidean metric.

Authors in [21] investigate the collaboration among a truck and multiple drones in a last-mile package delivery scenario. Each delivery has an energy cost associated, a reward based on the priority of the delivery, and both launch and rendezvous times with the truck. The work aims at finding the optimal scheduling for the drones that maximizes the overall reward, subject to the drone's battery capacity while ensuring that the same drone performs deliveries that do not overlap. Results show that the presented problem is computationally intractable, and therefore, different time-efficient heuristics are proposed [33]. Differently from our work, not all the deliveries can be performed due to the conflicts of deliveries. Moreover, their objective function aims at maximizing the total reward regardless of the total traveled distance.

III. Problem Definition

In this section, we first introduce the system model which characterizes the delivery area and how the drone moves inside it, and then we formally describe the problem to solve. The aim of our work is to find the DP that minimizes the sum of distances between the DP and all the customers to be served by the drone. The DP is a location inside the delivery area that depends on the set of customers. So, changing the set of customers, the DP varies. We start by stating a few assumptions.

A. Assumptions

To implement our solution, the drone has to work in symbiosis with a mobile pod, located at the selected DP, that stores all the packages to be delivered for that particular set of customers. We neglect the transportation cost (additional distance) for moving the mobile pod from the main depot to the DP because such a cost is paid just once for any set of customers and does not affect the drone's resources. Such a cost is obviously "non-negligible" for the transportation company, but in our proposed model it does not impact on the drone. Namely, another vehicle like a truck is in charge of moving the pod (with the drone) inside the city. So, since the drone's cost and the transportation cost are not really comparable each other, we only took into account the drone's cost (i.e., the total flown distance to minimize).

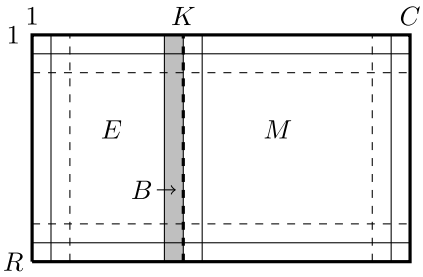

Fig. 1. The EM-grid with R rows, C columns. Each cell identified by the pair row and column, represents the "block" of a customer.

The deliveries we deal with all have the *same priority*. In fact, we believe that, currently, drone-based deliveries are offered as a premium service. Our solution is not affected by the order in which the deliveries are scheduled, so we imagine that the transportation company simply applies a *first-in-first-out order* to serve the requests assigned to the same drone. In future works, as drone-based systems become widely used, a prioritized delivery system could be studied to distinguish among the customers. In any case, our solution can be adapted for those subsets of deliveries with the same priority.

B. The System Model

The positions of the customers to be served lie inside a *Euclidean-Manhattan-Grid* (EM-grid). This area is defined as a 2-dimensional grid $G = (R, C, K)$ which consists of R rows, C columns, and a column parameter $K \in [1, C]$ (which defines the *Border* B) that separates the *Euclidean* sub-grid E (i.e., the rural area) from the *Manhattan* sub-grid M (i.e., the urban area). Specifically, $E = \{1, \ldots, R\} \times \{1, \ldots, K\}$, $B = \{1, \ldots, R\} \times \{K\} \subseteq E$, and $M = \{1, \ldots, R\} \times \{K+1, \ldots, C\}$. The border B consists of the single column K.

Conventionally, the delivery area is only Euclidean if $K = C$, and only Manhattan when $K = 1$. In an EM-grid, any internal vertex $u = (r_u, c_u)$ of G, i.e., with $1 < r_u < R$ and $1 < c_u < C$, is connected to the four adjacent vertices $(r_u, c_u \pm 1)$ and $(r_u \pm 1, c_u)$; whereas, in general, any vertex of the grid, i.e., with $1 \leq r_u \leq R$ and $1 \leq c_u \leq C$, is connected only to the existing adjacent vertices (i.e., an external vertex has only three or two neighbors). For simplicity, we assume that the distance between any two pairs of consecutive vertices on the same row or column is constant and unitary. Figure 1 shows an example of EM-grid.

The drone follows either the Euclidean metric when it moves inside the rural area, or the Manhattan metric when it moves inside the urban area. In fact, for any two vertices u, v in G, the distance $d(u, v)$ is the length of the shortest path

traversed by the drone in the EM-grid to go from $u = (r_u, c_u)$ to $v = (r_v, c_v)$. Recalling that the Euclidean and Manhattan distances are defined, respectively, as:

$$d_E(u, v) = \sqrt{(r_u - r_v)^2 + (c_u - c_v)^2}$$
$$d_M(u, v) = |r_u - r_v| + |c_u - c_v|,$$

then, the distance $d(u, v)$ for $u, v \in G$ is given by:

$$d(u, v) = \begin{cases} d_E(u, v) & \text{if } u, v \in E \\ d_M(u, v) & \text{if } u, v \in (M \cup B) \\ \min\limits_{w \in B}\{d_E(u, w) + d_M(w, v)\} & \text{if } u \in E, v \in M \\ d(v, u) & \text{if } u \in M, v \in E \end{cases}$$

where $d(u, v) = d(v, u)$ because both Euclidean and Manhattan distances are symmetric.

By Lemma 1, the shortest path $d(u, v)$ is unique and passes through the vertex in row v on the border B.

Lemma 1 [27]: Consider an EM-Grid $G = (R, C, K)$. Given $u = (r_u, c_u) \in E$ and $v = (r_v, c_v) \in M$, then $d(u, v) = d_E(u, h) + d_M(h, v)$ with $h = (r_v, K)$.

Thus, from now on, $d(u, v)$ is finally given by:

$$d(u, v) = \begin{cases} d_E(u, v) & \text{if } u, v \in E \\ d_M(u, v) & \text{if } u, v \in (M \cup B) \\ d_E(u, h) + d_M(h, v) & \text{if } u \in E, v \in M \\ & h = (r_v, K) \in B \\ d_M(u, h) + d_E(h, v) & \text{if } u \in M, v \in E \\ & h = (r_u, K) \in B \end{cases} \quad (1)$$

C. The Problem Formulation

The basic task in a delivery area is to serve a subset of customers with the help of a drone. Due to payload constraints, the drone cannot serve all the customers on the same flight. So, in a given EM-grid, a drone has to fly from the DP to all the customers. A *mission* $\mathcal{M} = \cup_{i=1}^{m}\{{u_i}^{(t_{u_i})}\}$ for a drone consists of m distinct customers (vertices of G), denoted as u_i, with $i = 1, \ldots, m$, each with *multiplicity* $t_{u_i} \geq 1$. That is, the customer u_i has ordered t_{u_i} different packages that the drone has to separately deliver. The mission $\mathcal{M}$ consists of overall $n = \sum_{i=1}^{m} t_{u_i}$ packages to be delivered. Let $\mathcal{M}_E$ be the subset of customers which reside in E, and $\mathcal{M}_M$ the subset of customers which reside in M. Clearly, $\mathcal{M} = \mathcal{M}_E \cup \mathcal{M}_M$. Let $n_E = \sum_{u \in \mathcal{M}_E} t_u$ and $n_M = \sum_{u \in \mathcal{M}_M} t_u$. Moreover, let the *cost function* for the drone applied to $\mathcal{M}$ having set the DP at $u \in G$ be:

$$\mathcal{C}(\mathcal{M}, u) = 2 \sum_{v \in \mathcal{M}} t_v d(v, u) \quad (2)$$

that is, the cost function is the sum of the distances between any item $v \in \mathcal{M}$ and the point u. Notice that the multiplicative constant 2 considers the round trip for each delivery. To ease the notation, we can write $\mathcal{C}(u)$ instead of $\mathcal{C}(\mathcal{M}, u)$ because $\mathcal{M}$ is assumed to be invariant. Our aim is to find the optimal DP $u^* = (r^*, c^*)$ in G that minimizes the cost function:

$$u^* = \operatorname*{argmin}_{u \in G} \mathcal{C}(u). \quad (3)$$

We denote this problem as the *Single Distribution Point Problem* (SDPP) because the goal is to find the *single* DP in EM-grids. The SDPP considers only a single drone, however it can be extended to a multi-drone case. Two possible scenarios can be envisioned: 1) by using multiple pods each with a single drone, or 2) by using a single pod with multiple drones. In the first scenario, the pods create a partition of customers and for each subset of customers our solution of SDPP is applied to minimize the distance traversed by the drone. In the second scenario, the customers must be partitioned among the drones to balance the distance traversed by the drones and simultaneously minimize the total (global) distance of all the drones from the DP.

D. The Optimal OPT Algorithm

The brute-force algorithm OPT optimally solves SDPP. It sequentially considers each vertex $u \in G$ as candidate to be the DP, and returns the point u^* that minimizes Eq. (3). Hence, since the grid G consists of R rows and C columns, for a given mission $\mathcal{M}$ with n packages to deliver, the time complexity for finding u^* by performing an exhaustive search is $\mathcal{O}(nRC)$.

The SDPP is strictly polynomial when applied to a fully Manhattan grid. In that case, the coordinates of the DP are just computed as the median of the x- and y-coordinates of the n deliveries in $\mathcal{O}(n)$ time [34]. Very different is the situation for the Euclidean case. In the literature, SDPP has been studied for points distributed in the continuous Euclidean space. In such a case, the DP is known as the *geometric median*. In the special case of three points, the DP is known as the *Fermat point* [35]. In general, no explicit formula, nor an exact algorithm involving only arithmetic operations and k^{th} roots, can exist for an arbitrary set of points in the Euclidean space [36]. So, only numerical or symbolic estimation of the solution of this problem are possible. However, it is possible to calculate an arbitrary good estimate of the geometric median by using an iterative procedure where each step produces a more accurate estimation. Similar procedures can be performed because the function "sum of distances to the delivery points" is a convex function. Therefore, procedures that iteratively decrease the sum of distances cannot get trapped in local optima. A common approach is relying on the Weiszfeld's algorithm [37]. This algorithm should in principle also work for full Euclidean and EM-grids, because the sum of the distances to the delivery points remains a convex function. However, the solution is not guaranteed to belong to a grid's point. For this reason, we sacrifice the accuracy of these solutions and prefer simpler and faster algorithms to be computed whenever required for EM-grids of arbitrary sizes.

Having that said, the optimal position of the DP can be exactly computed for EM-grid grids by invoking OPT. However, since both R and C can be exponentially large with respect to the number of deliveries n, the time complexity $\mathcal{O}(nRC)$ of OPT can be indeed exponential in the size of the instance. Therefore, OPT is applied only as a touchstone to evaluate the accuracy of the other proposed algorithms.

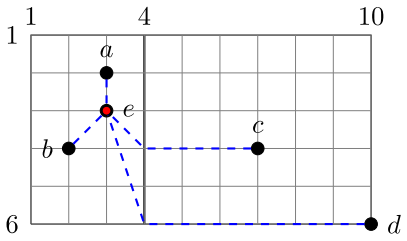

Fig. 2. An EM-grid $G = (6, 10, 4)$ with $a = (2, 3)$, $b = (4, 2)$, $c = (4, 7)$, $d = (6, 10)$, and $e = (3, 3)$. The u^* is in red in position $(3, 3)$. The dashed lines that connect all the points with u^* represent the paths of the drone.

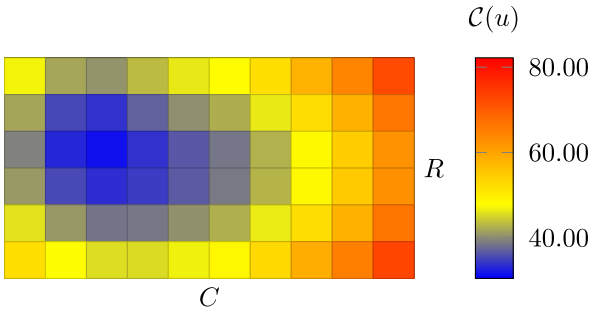

Fig. 3. Heatmap of the example in Figure 2. The colors illustrate the value $\mathcal{C}(u)$ for each $u \in G$. The best points are the ones with *cold* colors (blue), while the worst ones are in *hot* colors (red).

Finally, observe that some of the new proposed algorithms refer to the *centroid*, which is similarly defined to the geometric median, but minimizes the sum of the *squared distances* to each point. For the centroid, differently from the geometric median, a closed formula does exist, and for this reason sometimes it is used to replace the geometric median [38]. It can be found as the average (x- and y-coordinates) of the coordinates of the points.

So, in the next section we devise better time-efficient approximation and heuristic algorithms for solving SDPP.

Figure 2 highlights an example of EM-grid which will be used for describing all the presented algorithms in Section IV. Let $\mathcal{M} = \{a, b, c, d, e\}$ be the set of customers to be served. By running the optimal algorithm OPT that exhaustively tries each vertex as solution, the DP that minimizes Eq. (3) is $(3, 3)$. When using the point $u^* = (3, 3)$, the total cost by computing Eq. (2) for the drone is $2\sum_{v\in\mathcal{M}} d(v, u^*) = 2 + (2\sqrt{2}) + (6 + 2\sqrt{2}) + (12 + 2\sqrt{10}) \approx 31.98$.

Finally, Figure 3 shows the importance of choosing a good DP in the delivery area. Each *square* represents a vertex $u \in G$ whose *color* quantify the value $\mathcal{C}(u)$. From Figure 2, we know that the optimal point is $u^* = (3, 3)$, and hence, according to the adopted coloring, its square is the *coldest* one (i.e., blue). It is worth nothing that, although the example is very small, if we randomly select a DP (e.g., the top or bottom right corners) the total cost is more than twice the optimal cost. At the same time, for reasons of symmetry, there is a subset of candidate points whose cost is very close to the optimal cost. Note that the costs in Figure 3 are relative to the example in Figure 2. Changing the customers' positions, the heatmap changes and thus the optimal and the other (possible) DPs.

We now focus on two special cases of SDPP when $m = 2$, i.e., when there are only two customers. In general, finding the optimal solution with an arbitrary number of customers is not trivial. However, in the following scenario with only two customers, we can reduce the number of possible candidate DPs and hence determine the optimal solution.

Lemma 2: Given $\mathcal{M} = \{u^{(t_u)}, v^{(t_v)}\}$, then the DP u^* belongs to the shortest path between u and v.

Proof: We select a vertex $q \in G$ that does not belong to the shortest path between u and v. W.l.o.g., let $t_u \geq t_v$. Let p be the vertex of the shortest path such that $d(u, p) = d(u, q)$. Then, the cost $\mathcal{C}(q)$ of setting the DP at q yields:

$$\begin{aligned}
\mathcal{C}(q) &= t_u d(u, q) + t_v d(q, v) \\
&= t_u d(u, q) - t_v d(u, q) + t_v d(u, q) + t_v d(q, v) \\
&= (t_u - t_v) d(u, q) + t_v (d(u, q) + d(q, v)) \\
&\geq (t_u - t_v) d(u, q) + t_v d(u, v) \\
&= (t_u - t_v) d(u, p) + t_v d(u, v) \\
&= t_u d(u, p) + t_v (d(u, v) - d(u, p)) \\
&= t_u d(u, p) + t_v d(p, v) = \mathcal{C}(p)
\end{aligned}$$

So, any point q outside the shortest path increases the cost. □

When the two delivery points have the same multiplicity, not only the points of the shortest path are candidates, but all of them optimize the cost. Specifically,

Lemma 3: Given $\mathcal{M} = \{u^{(t_u)}, v^{(t_v)}\}$, if $t_u = t_v \geq 1$, then, any point p along the shortest path between u and v can be the DP.

Proof: When $t_u > t_v$, the best position of the DP is u, i.e., the vertex with the largest multiplicity. Namely, for any point $p \neq u$ along the shortest path between u and v, it yields:

$$\begin{aligned}
\mathcal{C}(p) &= t_u d(u, p) + t_v d(p, v) \\
&= t_u d(u, p) + t_v d(u, v) - t_v d(u, p) \\
&= t_v d(u, v) + (t_u - t_v) d(u, p) \\
&> t_v d(u, v) = \mathcal{C}(u)
\end{aligned}$$

□

These preliminary observations motivate our further investigation in the next section.

IV. Proposed Algorithms

In this section, we propose the time-efficient algorithm, called APX, that solves SDPP returning the best solution among the solutions of four algorithms, each exploiting a different peculiarity of the EM-grid. In particular, two algorithms fix the DP pretending that the whole area is fully Euclidean (GEC) or fully Manhattan (GMM). Then, two more algorithms are given that force the DP to reside either in the Euclidean area (ECMB) or in the Manhattan area (MMEB). It is important to note that, differently from the OPT algorithm, the new algorithms determine the DP just using the positions of the customers and the knowledge of the border's position in EM-grid. The width of the Euclidean and Manhattan area as well as the values R and C do not play any role in determining the DP (except for MMEB).

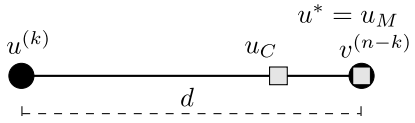

Fig. 4. A particular instance when invoking GEC.

A. The GEC Algorithm

The *Global Euclidean Centroid* (GEC) algorithm is a heuristic that solves SDPP by pretending the EM-grid as a full Euclidean grid, i.e., by selecting the *centroid* of the n customer locations. To be more precise, given a mission $\mathcal{M} \in G$, the centroid u_C is:

$$u_C = \left(\frac{\sum\limits_{u \in \mathcal{M}} t_u r_u}{n}, \frac{\sum\limits_{u \in \mathcal{M}} t_u c_u}{n} \right). \tag{4}$$

The GEC algorithm selects the centroid u_C as the DP. Note that since G is a discrete grid, we have to round each individual coordinate of u_C to the closest row and column, thus introducing a further error. Moreover, according to Eq. (4), u_C can be computed in $\mathcal{O}(n)$ time.

Observation 1: A lower bound for the approximation ratio of GEC is 2. Consider the configuration illustrated in Figure 4.

The two vertices u and v (that represent two customers) on the same row have multiplicity k and $n-k$, respectively, with $k \ll n$. It is easy to see that $u^* = v$, while, by Eq. (4), u_C is closer to v than u. The cost of setting the DP at u_C is $k\left(\frac{n-k}{n}\right)d + (n-k)\left(\frac{k}{n}\right)d = 2k\left(\frac{n-k}{n}\right)d$, while the optimal cost of setting the DP at $u^* = u_M$ is kd. For large values of n, the ratio between this solution and the optimal solution tends to 2. Note that, in this configuration we neglect the position of the border K since GEC assumes the delivery area as a full Euclidean area. Also, this illustration does not guarantee that in general the *upper bound* for the approximation ratio is 2. For this consideration, GEC is not an approximation algorithm.

If we perform the GEC algorithm on the example of Figure 2, the average row among $2, 4, 4, 6, 3$ is 3.8, while the average column among $3, 2, 7, 10, 3$ is 5. By rounding the average row, we have that $u_C = (4, 5)$, which belongs to the Euclidean area. In this case, the total cost is 35.30 and the ratio $\frac{35.30}{31.98} = 1.104$, meaning that in this example, the cost of GEC is 10% more than the optimal cost.

B. The ECMB Algorithm

The *Euclidean Centroid by projecting M to B* (ECMB) algorithm solves SDPP by forcing the DP to reside in the Euclidean area. Consider a mission $\mathcal{M} = \mathcal{M}_E \cup \mathcal{M}_M$. If the DP is selected in the Euclidean side, to serve each destination in the Manhattan area, the drone crosses the border B and then flies horizontally up to the destination (see Eq. (1)). Overall, the drone covers the distance $H = \sum_{u \in \mathcal{M}_M}(c_u - K)$ in M for serving the customers $\mathcal{M}_M$. The DP is then selected as the centroid of $\mathcal{M}' = \mathcal{M}_E \cup \mathcal{M}'_M$, where $\mathcal{M}'_M$ replaces each original customer $u = (r_u, c_u)$ in the Manhattan side with its projection $u' = (r_u, K)$ on the border B. Namely, one can see that all the points in $\mathcal{M}'$ reside in E. Hence, as explained in Section IV-A, a good DP is the vertex:

$$u_{\hat{C}} = \left(\frac{\sum\limits_{u \in \mathcal{M}} t_u r_u}{n_E + n_M}, \frac{\sum\limits_{u \in \mathcal{M}_E} t_u c_u + \sum\limits_{u \in \mathcal{M}_M} t_u K}{n_E + n_M} \right), \tag{5}$$

where $u_{\hat{C}}$ is the centroid of $\mathcal{M}'$. The total cost is given by H plus the sum of the distances traversed in E to move the points in $\mathcal{M}'$ to $u_{\hat{C}}$. The point $u_{\hat{C}}$ can be computed in $\mathcal{O}(n)$.

Notice that ECMB would not return the optimal point u^* even if we know in advance that the optimal DP resides in the Euclidean area since the centroid is not optimal for GEC, as previously discussed in Observation 1.

Observation 2: The approximation ratio of the ECMB algorithm in unbounded. In fact, consider Figure 4 and the value of the border K. Since ECMB forces the DP in the Euclidean area, if the border crosses the vertex v or it is to the right of v, then the analysis is exactly the same as before because u and v belong in E. However, the border can be also to the left of v. If the border crosses the vertex u, ECMB returns u because, after moving the $\mathcal{M}_M$ on K, all the items are in u. This solution has total cost $d(n-k)$. Recalling that the optimal solution has a cost of kd, the ratio would be $\frac{n-k}{k}$, and for large values of n this ratio tends to n. Even worse is the situation where the border is beyond u, on its left. Here, both u and v have to move because they reside in M, but the optimal cost is still kd, so the ratio can be arbitrarily large.

If we perform the ECMB algorithm on the example of Figure 2, we have initially to project c and d to the border. So, we would have to consider the projected points $c' = (4, 4)$ and $d' = (6, 4)$. Then, the arithmetic mean of the rows 2, 4, 4, 6, 3 is 3.8 (as before in GEC), while the arithmetic mean of the columns (considering the projected points) $3, 2, 4, 4, 3$, is 3.2. By rounding, we have that $u_{\hat{C}} = (4, 3)$, which obviously belongs to the Euclidean area. In this case, the total cost is 32.47 and the ratio $\frac{32.47}{31.98} = 1.015$, which is very close to the best solution (only 1.5% more than the optimal cost).

C. The Approximation GMM Algorithm

The *Global Manhattan Median* (GMM) algorithm solves SDPP by returning the *median* of all the points, pretending the whole grid as a pure Manhattan grid, i.e., $G(R, C, 1)$. In the literature, for the *Manhattan space* the optimal DP is called as the *median* [27], and it will be denoted as:

$$u_M = (\tilde{r}_{\mathcal{M}}, \tilde{c}_{\mathcal{M}}), \tag{6}$$

where $\tilde{r}_{\mathcal{M}}$ and $\tilde{c}_{\mathcal{M}}$ are the median of the rows and the columns, respectively, of the customers, up to multiplicities, in $\mathcal{M}$. We can use the median u_M as the DP of an arbitrary delivery area, regardless of the value of K.

The median u_M can be efficiently computed [39] with time complexity $\mathcal{O}(n)$. GMM guarantees an *approximation ratio* of $\sqrt{2}$, which means that the point u_M ensures that the total drone's cost is upper bounded by a factor of $\sqrt{2}$ of the optimal cost returned by OPT. Specifically,

Theorem 1: The approximation ratio performing the GMM algorithm is $\sqrt{2}$.

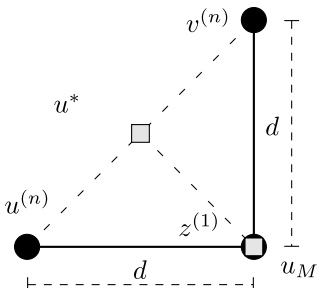

Fig. 5. A particular instance when invoking GMM.

Proof: When the DP is set to the optimal point u^*, let $\mathcal{C}_M(u^*)$ and $\mathcal{C}_E(u^*)$ be the total Manhattan and Euclidean cost for the drone, respectively, for serving u^*. Since $d_E(u,v) \leq d_M(u,v)$, then $\mathcal{C}(u_M) \leq \mathcal{C}_M(u_M)$ and $\mathcal{C}(u^*) \geq \mathcal{C}_E(u^*)$. By the optimality of u_M for a full Manhattan grid, one has $\mathcal{C}_M(u_M) \leq \mathcal{C}_M(u^*)$. Recalling the Cauchy-Schwarz inequality, that is, $a+b \leq \sqrt{2}\sqrt{a^2+b^2}$, one has $d_E(u,v) \leq d_M(u,v) \leq \sqrt{2}d_E(u,v)$ and therefore:

$$\frac{\mathcal{C}(u_M)}{\mathcal{C}(u^*)} \leq \frac{\mathcal{C}_M(u^*)}{\mathcal{C}_E(u^*)} \leq \sqrt{2}.$$

□

Observation 3: There is an instance such that the approximation ratio performing GMM tends to $\sqrt{2}$. Suppose to have a configuration like the one illustrated in Figure 5.

The three vertices $u = (0,0)$, $v = (d,d)$, and $z = (d,0)$ belong to the Euclidean part. Each of the two vertices u and v have multiplicity n, while the third vertex has a single multiplicity. For large values of n, the DP is attracted by the straight line $\overline{uv}$. In fact, if we had only u and v without the point z, by Lemma 2, any point on $\overline{uv}$ would be optimal. However, because of z, only the closest point to z on the line $\overline{uv}$, i.e., $\left(\frac{d}{2}, \frac{d}{2}\right)$, is optimal. So, selecting $u^* = (\frac{d}{2}, \frac{d}{2})$ as the DP, it holds $\mathcal{C}(u^*) = \frac{d}{2}\sqrt{2} + nd\sqrt{2}$. Instead, GMM returns $u_M = z$ for any value of n and $\mathcal{C}(z) = 2nd$. Thus, for large values of n, $\frac{\mathcal{C}(z)}{\mathcal{C}(u^*)} \to \sqrt{2}$.

If we perform the GMM algorithm on the example of Figure 2, we have to individually consider the median of rows and median of columns. The row median among the sorted sequence $2, 3, 4, 4, 6$ is 4, while the column median among the sorted sequence $2, 3, 3, 7, 10$, is 3. Hence, we have that $u_M = (4,3)$, which belongs to the Euclidean area. Note that incidentally $u_{\hat{C}}$ and u_M coincide. Hence, the total cost is 32.47 and the ratio $\frac{32.47}{31.98} = 1.015$.

D. The MMEB Algorithm

The *Manhattan Median by projecting E to B* (MMEB) algorithm solves SDPP by applying the same strategy as ECMB, only replacing Euclidean with Manhattan. In other words, MMEB forces the DP to reside in the Manhattan area. Given a mission $\mathcal{M} = \mathcal{M}_E \cup \mathcal{M}_M$, to serve all the customers of $\mathcal{M}_E$ while forcing the DP to reside in the Manhattan side, the drone must fly passing through a unique point of the border B (see Eq. (1)). To visit all the Euclidean customers of $\mathcal{M}_E$ from the border B in position $v_i = (r_{v_i}, K)$ (with $1 \leq i \leq R$), the drone travels the distance $D_{v_i} = \sum_{u \in \mathcal{M}_E} d_E(u, v_i)$. The total Euclidean distance D_{v_i} depends on the unique junction position v_i for connecting the customers of $\mathcal{M}_E$ to the border. Let $\mathcal{M}'_E(i)$ be the i^{th} set $\mathcal{M}_E$ where each original customer $u = (r_u, c_u)$ in the Euclidean side is replaced by its projection $u' = (r_{v_i}, K)$ on the border, for $1 \leq i \leq R$. One can see that to serve the customers in $\mathcal{M}$, the overall cost is D_{v_i}, plus the cost for serving the customers in $\mathcal{M}'(i) = \mathcal{M}'_E(i) \cup \mathcal{M}_M$ in a pure Manhattan grid. Hence, the MMEB algorithm selects as the DP location for the drone the vertex:

$$u_{\hat{M}} = \underset{1 \leq i \leq R}{\operatorname{argmin}} \; \mathcal{C}\left(\tilde{r}_{\mathcal{M}'(i)}, \tilde{c}_{\mathcal{M}'(i)}\right), \tag{7}$$

where $u_{\hat{M}}$ is the Manhattan median of $\mathcal{M}'(i)$, and $\tilde{r}_{\mathcal{M}'(i)}$ and $\tilde{c}_{\mathcal{M}'(i)}$ are the median of the rows and the columns, respectively, of the customers, up to multiplicities, in the i^{th} set $\mathcal{M}'(i)$. In Eq. (7), the value of $\tilde{c}_{\mathcal{M}'(i)}$ is the same regardless of the value of i, while $\tilde{r}_{\mathcal{M}'(i)}$ changes accordingly. Note that, $u_{\hat{M}}$ can be computed in $\mathcal{O}(nR)$.

If we perform MMEB on the example of Figure 2, we have to sequentially project a, b, and c to the border, starting from the first row. So, for each $1 \leq i \leq 6$ we would have to consider the projected points $a' = b' = c' = (i,4)$. Then, we compute the median in the Manhattan grid (as before in GMM) and compute the total cost. Eventually, we return the best i that minimizes Eq. (7). In this example, we have that $u_{\hat{M}} = (4,4)$, which obviously belongs to the Manhattan area (border). In this case, the total cost is 33.30 and the ratio $\frac{33.30}{31.98} = 1.041$.

Observation 4: The MMEB algorithm would return, more efficiently, the global optimal solution for SDPP if it is known that u^* resides in the Manhattan grid, in just $\mathcal{O}(nR)$ time. Namely, instead of testing all the positions of the Manhattan grid (that are $R(C-K)$), MMEB considers only R distinct positions on the border and, for each of them, it computes the median of all the customers assuming the customers in E (i.e., the subset $\mathcal{M}_E$) projected on the border. By this observation, we can decrease the time complexity of the brute-force optimal algorithm OPT to $\mathcal{O}(nRK + nR)$.

E. The Approximation APX Algorithm

All the presented algorithms have peculiarities that make them suitable for particular settings. For instance, MMEB works well when the delivery area is mostly Manhattan, whereas ECMB works well when the area is mostly Euclidean. Therefore, this suggests a new algorithm, that we call APX, which selects as DP the *best* point u_B among the four previous ones, i.e.,

$$u_B = \operatorname{argmin} \left\{\mathcal{C}(u_C), \mathcal{C}(u_{\hat{C}}), \mathcal{C}(u_M), \mathcal{C}(u_{\hat{M}})\right\}.$$

Theorem 2: The approximation ratio performing the APX algorithm is $\sqrt{2}$.

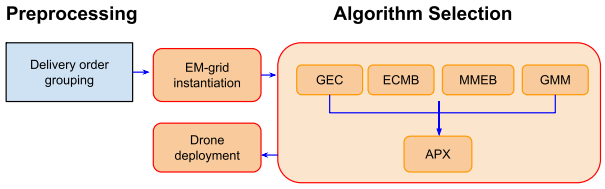

Fig. 6. Model framework of our proposed drone-based delivery system.

Proof: APX inherits the approximation ratio of Theorem 1, and trivially:

$$\frac{\mathcal{C}(u_B)}{\mathcal{C}(u^*)} \leq \frac{\mathcal{C}(u_M)}{\mathcal{C}(u^*)} \leq \frac{\mathcal{C}_M(u^*)}{\mathcal{C}_E(u^*)} \leq \sqrt{2}.$$

□

For completeness, in the example of Figure 2, the APX algorithm would compare the best DP among $u_C = (4,5)$, $u_{\hat{C}} = (4,3)$, $u_M = (4,3)$, and $u_{\hat{M}} = (4,4)$ whose costs are 35.30, 32.47, 32.47, and 33.30, respectively. So, the best DP selected by APX is $(4,3)$.

F. Discussion About the Algorithms

In Section IV, we introduced several heuristic and approximation algorithms that can deal with any EM-grid with arbitrary size. First, the centroid-based algorithms GEC and ECMB work well when the EM-grid is mainly Euclidean, and therefore obtain bad results when the EM-grid is mainly Manhattan. A symmetric trend is observed with the two median-based algorithms GMM and MMEB, where they work well when the EM-grid is mainly Manhattan (see Figures 7–8). By definition GEC prefers fully Euclidean grids, while ECMB obtains good results when the EM-grid is evenly balanced (i.e., $K = \frac{C}{2}$). Like ECMB, MMEB prefers balanced scenarios but its complexity is larger. Finally, GMM is in general the best one in terms of performance and time complexity, but its performance lacks when the EM-grid is evenly balanced.

To conclude, since our algorithms present advantages and disadvantages that depend on the current layout and number of deliveries, we devised the APX which returns the best sub-optimal solution in reasonable time, by comparing the aforementioned 4 solutions.

In Figure 6, we summarize the model framework concerning our proposed drone-based delivery system. The initial gray block refers to the pre-processing phase, not part of this work, in which a hypothetical order or grouping for the deliveries, is arranged. Next, is the EM-grid instantiation, based on the deliveries distribution from which follows the Algorithm Selection block. At this point, the presented algorithms are evaluated, and the best one (i.e., APX) is used to compute the drone's trajectories. At the end of the process, the mobile pod is deployed at the DP, and the drone performs the deliveries.

In Table I, we compare the presented algorithms evaluating their time complexities and guaranteed approximation bounds. Note that we have found two approximated solutions: GMM and APX. The former has time complexity linear in the number of the customers, while the latter has a time complexity that depends also on the size of the EM-grid. As we will see, although we can guarantee the same approximation ratio, the performance of APX is always better than that of GMM.

TABLE I
COMPARISON BETWEEN THE ALGORITHMS

Point	Type	Alg.	Time complexity	Apx. ratio
u^*	brute-force	OPT	$\mathcal{O}(nRK)$	1
u_C	centroid	GEC	$\mathcal{O}(n)$	–
$u_{\hat{C}}$	centroid	ECMB	$\mathcal{O}(n)$	–
u_M	median	GMM	$\mathcal{O}(n)$	$\sqrt{2}$
$u_{\hat{M}}$	median	MMEB	$\mathcal{O}(nR)$	–
u_B	comparison	APX	$\mathcal{O}(nR)$	$\sqrt{2}$

The time complexity of OPT depends on the size of the EM-grid, and it provides the optimal solution. The centroid-based solutions are very fast in terms of time complexity (linear in n) but they do not guarantee any upper bound in terms of accuracy of the solution. The other fast solution, i.e., GMM (median-based), has a time complexity again linear in n and it guarantees an approximation bound. It is worthy to note that, experimentally, we have observed that this upper bound is never reached, and generally it is $\leq$1.05, much better than $\sqrt{2} \approx 1.4$. The remaining median-based algorithm MMEB has a larger time complexity compared with the previous three solutions. Also, we could not provide any approximation ratio for it. Finally, the APX algorithm that selects the DP of minimum cost is very accurate in practice.

V. PERFORMANCE EVALUATION

In this section, we first numerically evaluate the goodness of our algorithms in EM-grids by varying the number of rows and columns, and the number of customers to be served. Then, with the help of the open air simulator BlueSky, we compare the cost of our best solution APX with the cost of a solution that serves the costumers from a fixed DP that emulates the depot's position of a delivery company.

A. Settings and Parameters

We implemented our algorithms in Python language version 3.5, and run all the instances on an Intel i7-10genK computer with 16 GB of RAM.

In order to evaluate our proposed algorithms for solving SDPP, we set different layouts by varying $R, C \in \{50, 100\}$ and $K \in \{1, \frac{1}{4}C, \frac{1}{2}C, \frac{3}{4}C, C\}$. We uniformly generate $n = |H|$ random positions (in which multiplicities can occur) inside the grid with $n = \{5, 10, 15, 20, 50, 100\}$, and then return the *average* ratio (along with the *standard deviation*) on 33 different random instances. Moreover, given a setting with n random points, we evaluate the algorithms when balancing the quantities n_E and n_M with respect to a certain fraction $p = \{0, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, 1\}$ on n, such that $n_E = p \cdot n$ and $n_M = (1-p) \cdot n$, with $n = n_E + n_M$.

We compare the algorithms with respect to OPT, and we plot the ratio $\rho = \frac{\mathcal{C}(H,\tilde{u})}{\mathcal{C}(H,u^*)} \geq 1$, where ρ is the ratio between the total cost for serving all the required customers assuming $\tilde{u}$ as DP returned by the tested algorithm, and the total cost outputted by the optimal algorithm assuming u^* as optimal DP, where $H \subseteq G$.

B. Numerical Analysis

Figure 7 reports the numerical evaluation of all the presented algorithms for solving SDPP with respect to the optimal

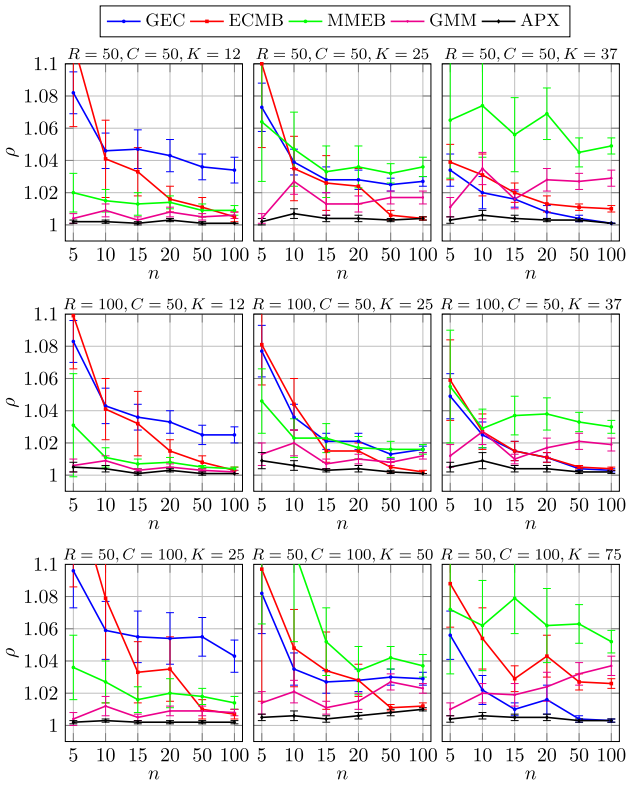

Fig. 7. All the algorithms on different layouts when varying n.

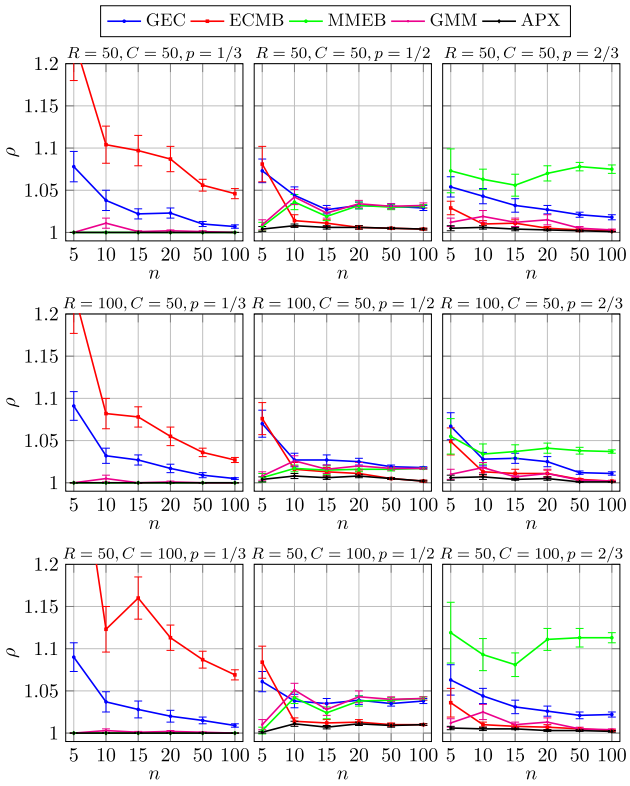

Fig. 8. All the algorithms on different layouts when varying p.

algorithm OPT. In particular, we show the performance of the algorithms when fixing a layout (R, C, K) and varying the number n of deliveries.

The first row of Figure 7 shows a squared layout, the second row a layout with $R > C$, and the third row a layout with $R < C$. Let us now focus on the first row. We initially observe a symmetric behavior between the centroid-based and the median-based algorithms. The *centroid-based* algorithms such as GEC and ECMB work well when the Euclidean side is the largest, i.e., K tends to C, while the *median-based* algorithms such as MMEB and GMM get a better performance when the Manhattan side is the largest, i.e., for smaller values of K. In general, when the number of deliveries n to be served is small, the algorithms present very variable results and hence the standard deviation is pretty large. This means that when n is small, it is easy to have instances for which the same algorithm can either work well or very poorly. Nevertheless, APX obtains a good and stable performance regardless of the values of n and K.

The best algorithm is APX whose average ratio is around 1.01, well below the guaranteed upper bound of $\sqrt{2} \approx 1.41$. For almost Manhattan grids, as expected, the worst performing algorithm is GEC, especially when n is large. For almost Euclidean grids, the worst performing algorithm is MMEB, which is worse than GMM. Concerning the second row of Figure 7, when $R > C$, all the algorithms obtain a good performance in the case of $K = \frac{C}{2}$. In principle, on this layout the algorithms behave almost the same as in the previous case with $R = C = 50$, but with much better performance. It is also interesting to note that in the opposite layout with $R < C$ (third row of Figure 7), the algorithms get a bad performance with respect to OPT. This bad trend can be seen especially with GMM, which has a counter-intuitive behavior. The poor performance when there are more columns than rows is justified by the fact that the impact of the border is higher, and there is a quite large difference between the centroid-based and the median-based algorithms. Indeed, with a small n the performance of GMM is good, but when n starts to increase, its performance degrades immediately. This is accentuated when the grid is almost Euclidean ($K = 75$).

Figure 8 compares the performance of the algorithms when fixing a layout $G = (R, C, \frac{C}{2})$ and varying the quantity p of deliveries to each side of the EM-grid. As before, the first row depicts the squared layout, while the second and the third rows show the two different rectangular layouts. On each plot we fixed the value of the border K as $\frac{C}{2}$, i.e., we zoom in the central column of Figure 7. In the first column of Figure 8, a third of the deliveries are on the Euclidean area, i.e., $p = \frac{1}{3}$, in the second column the deliveries are equally halved to each sub-area, i.e., $p = \frac{1}{2}$, and finally in the third column a third of the deliveries are on the Manhattan area, i.e., $p = \frac{2}{3}$.

In general, the performance are more variable than those already seen in Figure 7, so here we have a different scale on the y-axis for ρ up to 1.2. On these plots, ECMB performs poorly, especially for small values of p. This is due to the fact that ECMB forces the DP in the Euclidean area, and having two thirds of the deliveries in the Manhattan area, the total cost for the drone blows up accordingly. It is interesting to see how all the algorithms decently perform when the n deliveries are equally distributed on both the areas. In these cases, the best algorithms are GMM and, obviously, APX. As expected, the centroid-based algorithms work well when most of the points belong to the Euclidean grid, and the median-based ones work well when most of the points belong to the Manhattan grid.

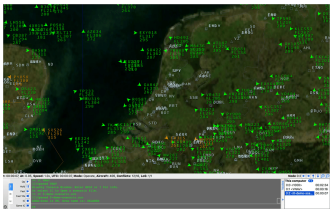

Fig. 9. A screenshot of the drone simulator BlueSky.

In the next section we move towards a much realistic environment, evaluating the performance of APX by using an open air simulator.

C. Simulation Results

For evaluating our algorithms on a simulated environment, in this work we rely on BlueSky. BlueSky is an open Air Traffic Simulator (ATS), and is meant as a tool to perform research on Air Traffic Management and Air Traffic Flows. It is distributed under the GNU General Public License v3. The goal of BlueSky is to provide a free tool in order to visualize, analyze, or simulate air traffic without any restrictions, licenses, or limitations [40]. Although BlueSky is an open air simulator, in this work we simply refer to it as drone simulator. A screenshot is shown in Figure 9.

In the simulated environment, we set $R = C = 50$ and $K = \{12, 25, 37\}$ which represent a grid with a third, a half, and two thirds of Euclidean columns, respectively. Overall, we fix a reasonable number of deliveries $n = 50$. Moreover, we set the speed of the drone to 10 m/s and its altitude 100 m above the ground. Finally, we set the unitary distance between two grid points as to 100 m, so the whole area is $5 \times 5\,\text{km}^2$. In the simulations, we evaluate the total drone's covered distance and elapsed time for accomplishing all the deliveries. Namely, the focus of our research is to minimize the mission's cost by selecting a single starting point from which all the deliveries are served. In order to expedite the simulations, we virtually set a drone for each individual delivery, and then we run in parallel all the flights and sum all the distance and time values.

We also compare our solution, that uses an ad-hoc DP for the current set of deliveries, with the solution that uses a fixed predefined DP, independent of the current deliveries. A fixed DP simply models a fixed warehouse built in a specific location which is the standard (current) solution for a delivery company. As already said, a DP that varies with the given set of deliveries, instead, implies to have a truck that moves the mobile pod at a certain position of the grid. The (non fixed) DP, that depends on the set of deliveries, introduces the additional cost of moving the pod, paid just once for all the deliveries of the same set. At the moment, we neglect such a cost because our goal is to see if it is advantageous to use the ad-hoc DP over the fixed one in terms of distance covered and time spent by the drone to serve the given set of deliveries. For this reason, we compare the cost of the DP returned by the best algorithm APX presented in this work with OPT, and with the following three fixed points:

- u_E (denoted as FIXE) is the center of the Euclidean area, i.e., $u_E = (\lfloor \frac{R}{2} \rfloor, \lfloor \frac{K}{2} \rfloor)$;
- u_B (denoted as FIXB) is the center of the border, i.e., $u_B = (\lfloor \frac{R}{2} \rfloor, K)$;

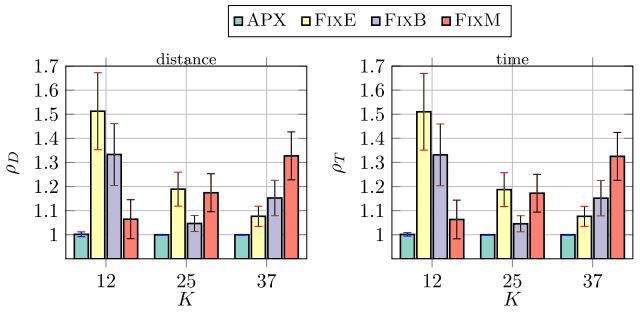

Fig. 10. Results by using BlueSky simulator comparing the distance and time with respect to the optimal one.

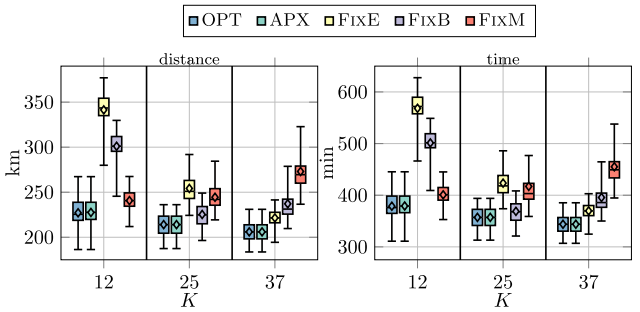

Fig. 11. Results by using BlueSky simulator evaluating the distribution (distance and time) of all the deliveries.

- u_M (denoted as FIXM) is the center of the Manhattan area, i.e., $u_M = (\lfloor \frac{R}{2} \rfloor, \lfloor \frac{C+K}{2} \rfloor)$.

In this section, in addition to the distance covered by the drone which was evaluated with the analytic experiments in Section V, we also evaluate the time spent by the drone during the missions. First, Figure 10 illustrates the qualitative assessment of our new solution by plotting the ratio between the distance (and time) cost of APX, and the distance cost (and time) of the fixed points FIXB, FIXB, and FIXM with respect to that of OPT. Then, Figure 11 gives the absolute distance and time values to quantitatively assess our new solution.

Figure 10 plots in the y-axis the ratio $\rho_D = \frac{\mathcal{C}(H,\tilde{u})}{\mathcal{C}(H,u^*)} \geq 1$ (on the left) and the ratio $\rho_T = \frac{\mathcal{T}(H,\tilde{u})}{\mathcal{T}(H,u^*)} \geq 1$ (on the right) where $\mathcal{T}$ represents the time required for performing all the round trips from the DP to the customers. Basically, on the 33 random simulated scenarios with BlueSky, APX and OPT almost always coincide on both distance and time. The ratio is thus very close (slightly better) to that of the analytic experiments in Figure 7 (first row, $n = 50$). Hence, fixed the source, destination, and intermediate points, in absence of external agents (e.g., wind[1]), the drones are routed in BlueSky along straight lines, without deviations and noises, as we assumed in Section V-B. The differences between ρ_D and ρ_T are pretty negligible. As one can now evaluate in Figure 11, the time obeys the physical law, that is, it is equal to the distance divided by the speed. Hence, without external wind, the drones in BlueSky constantly move at the specified speed. Although this aspect was not so obvious at the beginning when we started to consider this simulator, it is quite important because

[1]BlueSky allows to set a global wind speed and direction parameter when simulating the missions, influencing so the flight of drones.

we can now reasonably assume that not only the distance but also the time can be analytically estimated.

The main aim of Figure 10 is to evaluate the performance of the mobile DPs versus that of the fixed DPs. As expected, when K is small FIXM is preferable because the Manhattan side dominates in EM-grid, and when K is large FIXE is preferable because the Euclidean side dominates in EM-grid. In both cases, FIXB is in between. When $K = \frac{C}{2}$, the worst result occurs when applying FIXE or FIXM (middle of Euclidean or Manhattan grid, respectively) whose DPs are $u_E = (25, 12)$ and $u_M = (25, 37)$, respectively, while the other fixed point FIXB with $u_B = (25, 25)$ (middle of border) gets a quite good performance, just a bit worse than the optimal. This is reasonably expected since u_E and u_M are specular each other with respect to the border that halves the EM-grid, and both are equidistant from the border. The point u_B is closer to the optimal point u^* than the other two points, and therefore the cost when applying FIXE or FIXM is larger.

Finally, in Figure 11 we plot the simulated results in a much more thorough manner. In these plots, we make use of *box plots* [41] which report the *minimum*, the *first quartile*, the *median*, the *third quartile*, and the *maximum*. Also the *mean* is reported as the diamond. In particular, in the first plot we highlight the total covered distance by the drone (kilometers, km), while in the second plot we show the total makespan of the drone for performing all the deliveries (minutes, min).

We initially observe that the optimal solution OPT requires approximately 210–230 km as total distance flown for 50 deliveries. Therefore, each delivery requires an average of 4.2–4.6 km for a complete round trip to/from the DP. Obviously, the total covered distance increases when the value of K decreases due to the fact that the drone has to connect more intermediate points provided by the Manhattan metric. Concerning APX, its performance almost matches the one of OPT on these random instances. It is interesting to note that the mean and the median coincide for APX and OPT.

The three static points show the same behavior as before (see Figure 10), i.e., FIXE is better when K is large, while FIXM is better when K is small. Moreover, note that the mean of FIXE is smaller than its median when $K = 12$. Similarly, the mean of FIXM is larger than its median when $K = 37$. The span of values of FIXE is larger than that of FIXM. The worst performance can be observed for FIXE when $K = 12$ in which the drone has to fly for almost 350 km (7 km for a single delivery). The overall duration of the mission is about 600 min, i.e., about 10 hours (12 min for a single delivery). With respect to the worst case, the length of the missions in APX save up to 100 km and 150 min. Finally, in OPT and APX the drone flies for approximately 350 min (7 min for a single delivery), thus halving the delivery time with respect to the worst scenario. As expected, the missions in OPT are shorter when the Euclidean side dominates in EM-grid, and longer when the Manhattan side dominates in EM-grid. The OPT and APX missions saves from a minimum of 10 to a maximum of 150 km, and from 10 to 150 min with respect to the fixed DPs. In conclusion, FIXM (respectively, FIXE) has a reasonable performance if it is applied when the Euclidean side is small (respectively, large), while its performance is very poor if it is applied when the Euclidean side is large (respectively, small). That is, the behavior of FIXM and FIXE strongly depends on which side dominates in EM-grid. Therefore, both FIXM and FIXE can perform very poorly depending on the distribution of customers among both the sides. So, when the width of the Manhattan and Euclidean sides is not defined a priori, APX substantially outperforms all the other algorithms.

VI. CONCLUSION

In this work, we considered a drone which is in charge of delivering small packages to customers in a delivery area, modeled as an EM-grid. We proposed an optimal as well as approximation and heuristic algorithms for computing the DP that solves SDPP. Although we can guarantee the same approximation ratio for the two approximation algorithms, the performance of APX is always better than that of GMM. By using the BlueSky simulator, we compare our best solution APX with fixed DPs, selected as a function of the size RC of the EM-grid. Our solution saves from a minimum of 10 to a maximum of 100 km of traveled distance, and from 10 to 150 min of elapsed time with respect to the fixed DPs.

As a future work, we intend to extend our solution to multiple drones. This requires to partition the customers among the drones, and then apply our proposed solutions to each partition. We plan to further investigate the symbiotic cooperation between a truck, pods, and drones, by setting different DPs for subsets of customers, and then use a truck that sequentially visits each of these. Other possible extensions would include a hierarchy of customers with different priorities, so that the drones have to serve first the users with higher priority, and then the regular ones. Our solution will apply for subsets of customers with the same priority.

REFERENCES

[1] J.-P. Yaacoub, H. Noura, O. Salman, and A. Chehab, "Security analysis of drones systems: Attacks, limitations, and recommendations," *Internet Things*, vol. 11, Sep. 2020, Art. no. 100218.

[2] M. Ayamga, S. Akaba, and A. A. Nyaaba, "Multifaceted applicability of drones: A review," *Technol. Forecasting Social Change*, vol. 167, Jun. 2021, Art. no. 120677.

[3] R. Kellermann, T. Biehle, and L. Fischer, "Drones for parcel and passenger transportation: A literature review," *Transp. Res. Interdiscipl. Perspect.*, vol. 4, Mar. 2020, Art. no. 100088.

[4] J. Burgués and S. Marco, "Environmental chemical sensing using small drones: A review," *Sci. Total Environ.*, vol. 748, Dec. 2020, Art. no. 141172.

[5] A. Yazdinejad, R. M. Parizi, A. Dehghantanha, H. Karimipour, G. Srivastava, and M. Aledhari, "Enabling drones in the Internet of Things with decentralized blockchain-based security," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6406–6415, Apr. 2021.

[6] B. Skorup and C. Haaland, "How drones can help fight the coronavirus," Mercatus Center Res. Paper Ser., Special Ed. Policy Brief, Mar. 2020. [Online]. Available: https://ssrn.com/abstract=3564671, doi: 10.2139/ssrn.3564671.

[7] X. Chang, C. Yang, J. Wu, X. Shi, and Z. Shi, "A surveillance system for drone localization and tracking using acoustic arrays," in *Proc. IEEE 10th Sensor Array Multichannel Signal Process. Workshop (SAM)*, Jul. 2018, pp. 573–577.

[8] S. O. Al-Jazzar and Y. Jaradat, "AOA-based drone localization using wireless sensor-doublets," *Phys. Commun.*, vol. 42, Oct. 2020, Art. no. 101160.

[9] F. Betti Sorbelli, S. K. Das, C. M. Pinotti, and G. Rigoni, "A comprehensive investigation on range-free localization algorithms with mobile anchors at different altitudes," *Pervas. Mobile Comput.*, vol. 73, Jun. 2021, Art. no. 101383.

[10] B. Mishra, D. Garg, P. Narang, and V. Mishra, "Drone-surveillance for search and rescue in natural disaster," *Comput. Commun.*, vol. 156, pp. 1–10, Apr. 2020.

[11] S. Hayat, E. Yanmaz, C. Bettstetter, and T. X. Brown, "Multi-objective drone path planning for search and rescue with quality-of-service requirements," *Auto. Robots*, vol. 44, no. 7, pp. 1183–1198, Sep. 2020.
[12] C. Qu, R. Singh, A. E. Morel, F. B. Sorbelli, P. Calyam, and S. K. Das, "Obstacle-aware and energy-efficient multi-drone coordination and networking for disaster response," in *Proc. 17th Int. Conf. Netw. Service Manage. (CNSM)*, Oct. 2021, pp. 1–9.
[13] E. Barmpounakis and N. Geroliminis, "On the new era of urban traffic monitoring with massive drone data: The pNEUMA large-scale field experiment," *Transp. Res. C, Emerg. Technol.*, vol. 111, pp. 50–71, Feb. 2020.
[14] A. Khochare, Y. Simmhan, F. B. Sorbelli, and S. K. Das, "Heuristic algorithms for co-scheduling of edge analytics and routes for UAV fleet missions," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, May 2021, pp. 1–10.
[15] W. D. Rosamond *et al.*, "Drone delivery of an automated external defibrillator," *New England J. Med.*, vol. 383, no. 12, pp. 1186–1188, Sep. 2020.
[16] S. Cheskes *et al.*, "Improving access to automated external defibrillators in rural and remote settings: A drone delivery feasibility study," *J. Amer. Heart Assoc.*, vol. 9, no. 14, Jul. 2020, Art. no. e016687.
[17] H. Huang, A. V. Savkin, and C. Huang, "Reliable path planning for drone delivery using a stochastic time-dependent public transportation network," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 8, pp. 4941–4950, Aug. 2021.
[18] T. Cokyasar, W. Dong, M. Jin, and İ. Ö. Verbas, "Designing a drone delivery network with automated battery swapping machines," *Comput. Oper. Res.*, vol. 129, May 2021, Art. no. 105177.
[19] F. B. Sorbelli, F. Coro, S. K. Das, and C. M. Pinotti, "Energy-constrained delivery of goods with drones under varying wind conditions," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 9, pp. 6048–6060, Sep. 2021.
[20] A. Khanda, F. Coro, F. B. Sorbelli, C. M. Pinotti, and S. K. Das, "Efficient route selection for drone-based delivery under time-varying dynamics," in *Proc. IEEE 18th Int. Conf. Mobile Ad Hoc Smart Syst. (MASS)*, Oct. 2021, pp. 437–445.
[21] F. B. Sorbelli, F. Corò, S. K. Das, L. Palazzetti, and C. M. Pinotti, "Greedy algorithms for scheduling package delivery with multiple drones," in *Proc. 23rd Int. Conf. Distrib. Comput. Netw.*, Jan. 2022, pp. 31–39.
[22] T. Information. (2021). *How Many Packages Are Moved by Fedex and Ups Everyday?*. Accessed: Nov. 13, 2021. [Online]. Available: http://www.shorturl.at/gpEPW
[23] S. Anggraeni *et al.*, "The deployment of drones in sending drugs and patienblood samples COVID-19," *Indonesian J. Sci. Technol.*, vol. 5, no. 2, pp. 18–25, 2020.
[24] Y. Liu, "An optimization-driven dynamic vehicle routing algorithm for on-demand meal delivery using drones," *Comput. Oper. Res.*, vol. 111, pp. 1–20, Nov. 2019.
[25] A. E. McKellar, N. G. Shephard, and D. Chabot, "Dual visible-thermal camera approach facilitates drone surveys of colonial marshbirds," *Remote Sens. Ecol. Conservation*, vol. 7, no. 2, pp. 214–226, Jun. 2021.
[26] I. Duporge *et al.*, "Determination of optimal flight altitude to minimise acoustic drone disturbance to wildlife using species audiograms," *Methods Ecol. Evol.*, vol. 12, no. 11, pp. 2196–2207, Nov. 2021.
[27] L. Bartoli, F. B. Sorbelli, F. Coro, C. M. Pinotti, and A. Shende, "Exact and approximate drone warehouse for a mixed landscape delivery system," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, Jun. 2019, pp. 266–273.
[28] C. C. Murray and A. G. Chu, "The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery," *Transp. Res. C, Emerg. Technol.*, vol. 54, pp. 86–109, May 2015.
[29] S. Poikonen, B. L. Golden, and E. Wasil, "A branch-and-bound approach to the traveling salesman problem with a drone," *Informs J. Comput.*, vol. 31, no. 2, pp. 335–346, 2019.
[30] J. R. Brown, M. A. Bushuev, and A. L. Guiffrida, "Distance metrics matter: Analysing optimisation algorithms for the last mile problem," *Int. J. Logistics Syst. Manage.*, vol. 38, no. 2, pp. 151–174, 2021.
[31] N. Ai, J. Zheng, X. Chen, and K. Kawamura, "Neighborhood-specific traffic impact analysis of restaurant meal delivery trips: Planning implications and case studies in Chicago," *J. Urban Planning Develop.*, vol. 147, no. 2, Jun. 2021, Art. no. 05021013.
[32] F. B. Sorbelli, F. Coro, C. M. Pinotti, and A. Shende, "Automated picking system employing a drone," in *Proc. 15th Int. Conf. Distrib. Comput. Sensor Syst. (DCOSS)*, May 2019, pp. 633–640.
[33] M. R. Garey and D. S. Johnson, *Computers and Intractability*, vol. 174. San Francisco, CA, USA: Freeman, 1979.
[34] G. Wesolowsky, "The weber problem: History and perspectives," *Location Sci.*, vol. 1, pp. 5–23, 1993. [Online]. Available: https://www.scirp.org/(S(351jmbntvnsjt1aadkozje))/reference/referencespapers.aspx?referenceid=687435
[35] J. Krarup, "On Torricelli's geometrical solution to a problem of fermat," *IMA J. Manage. Math.*, vol. 8, no. 3, pp. 215–224, Mar. 1997.
[36] C. Bajaj, "Proving geometric algorithm non-solvability: An application of factoring polynomials," *J. Symbolic Comput.*, vol. 2, no. 1, pp. 99–102, Mar. 1986.
[37] E. Weiszfeld and F. Plastria, "On the point for which the sum of the distances to n given points is minimum," *Ann. Oper. Res.*, vol. 167, no. 1, pp. 7–41, Mar. 2009.
[38] M. H. Protter and C. B. Morrey, *College Calculus With Analytic Geometry*. Reading, MA, USA: Addison-Wesley, 1977.
[39] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, 2009.
[40] J. M. Hoekstra and J. Ellerbroek, "Bluesky ATC simulator project: An open data and open source approach," in *Proc. 7th Int. Conf. Res. Air Transp.*, vol. 131, 2016, p. 132.
[41] M. Frigge, D. C. Hoaglin, and B. Iglewicz, "Some implementations of the boxplot," *Amer. Statist.*, vol. 43, no. 1, pp. 50–54, Feb. 1989.

Francesco Betti Sorbelli (Member, IEEE) received the master's degree in computer science from the University of Perugia, Italy, in 2007, and the Ph.D. degree in computer science from the University of Florence, Italy, in 2018. He was a Post-Doctoral Researcher at the Missouri University of Science and Technology University, USA, under the supervision of Prof. Sajal K. Das in 2020. Currently, he is a Post-Doctoral Researcher at the University of Perugia under the supervision of Prof. Cristina M. Pinotti. His research interests include algorithms design, combinatorial optimization, and unmanned vehicles.

Cristina M. Pinotti (Senior Member, IEEE) received the master's degree *(cum laude)* in computer science from the University of Pisa, Italy, in 1986. In 1987 and 1999, she was a Researcher with the National Council of Research, Pisa. In 2000 and 2003, she was an Associate Professor at the University of Trento. Since 2004, she has been a Full Professor at the University of Perugia. Her current research interests include the design and analysis of algorithms for wireless sensor networks and communication networks.

Giulio Rigoni received the bachelor's and master's degrees in computer science from the University of Padua, Italy, in 2012 and 2017, respectively, and the Ph.D. degree in computer science from the University of Florence under the supervision of Prof. Cristina M. Pinotti in 2022. His research interests include unmanned vehicles, security and machine learning. He has been a member of the SPRITZ Security and Privacy Research Group since 2018.

Open Access funding provided by 'Università degli Studi di Perugia' within the CRUI-CARE Agreement