# Drones in the Big City: Autonomous Collision Avoidance for Aerial Delivery Services

Fabíola M. C. de Oliveira, Luiz F. Bittencourt, *Senior Member, IEEE*, Reinaldo A. C. Bianchi, *Member, IEEE*, and Carlos A. Kamienski, *Senior Member, IEEE*

***Abstract*— As delivery companies continue to explore the use of drones, the need for efficient and safe operation in urban environments becomes increasingly critical. Market-wide versions of drone delivery services will necessarily spread many drones, especially in big cities. In this scenario, avoiding collisions with other drones or typical obstacles in urban spaces is fundamental. This paper proposes and evaluates, via simulation and analytical modeling, an aerial delivery service scenario and three autonomous geometric approaches for collision avoidance. We compare our approaches with three simple methods – DoNothing (not detouring), Random, and aviation-like Rightward – and two state-of-the-art geometric approaches. Simulation experiments consider different fleet sizes with constant and Poisson drone arrival rates and drones randomly choosing one of different altitudes for the cruise flight. Contrary to our expectations, the Random and Rightward approaches increase the collisions compared with DoNothing, making the latter our baseline. Our approaches significantly reduce collisions in all experiments and deal with more drones within the detection radius, showing that collisions are more complex to avoid. Comparing the collision rate, successful trips, and the number of flying drones reveals that the efficiency in avoiding collisions reduces the number of successful trips by increasing the number of active drones. Regardless of the expected reduction in collisions, more altitudes do not eliminate them. These results indicate the need for more sophisticated approaches to reduce or eliminate collisions. The analytical modeling using Markov Chains corroborates the simulation results by shedding some light on and helping explain the simulation results.**

***Index Terms*— Smart drone delivery service, collision avoidance, geometric collision avoidance, logistics, smart cities, complex systems.**

## I. Introduction

A SMART aerial delivery service uses Unmanned Aerial Vehicles (UAVs), simply drones, to deliver packages in urban areas replacing terrestrial vehicles [1], [2]. We expect drones to be pervasive in urban areas, offering citizens services such as food and parcel delivery, fire detection, firefighting, medical equipment transport, and drone computing [3], [4]. Intelligent drone delivery is vital in reducing the environmental impact, delivery time, and costs, calling the attention of delivery companies that started to perform small-scale tests with drones[1,2] and providers offering this service, such as Alphabet's Wing.[3]

In big cities, a drone delivery service will necessarily employ many drones, increasing drone density and bringing the fundamental challenge of avoiding collisions with other drones or typical obstacles of urban spaces. Nevertheless, the main challenge is showing successful experiences of an operational delivery service [5], [6]. Drones fly at low altitudes, increasing the probability of collision with buildings, trees, mountains, birds, or even other drones [7]. Thus, the first step to deploying a multidrone system is avoiding collisions [3], [8], [9]. In a scenario with a high density of obstacles, drones may need a set of collision avoidance strategies to apply the most appropriate technique in each situation [10]. Different techniques have been proposed for this challenge, varying from non-cooperative purely autonomous decisions that do not involve communication with other drones or exterior computing support [11] to cooperative approaches involving communication and collaborative learning [12], [13].

Current studies in collision avoidance suffer from some limitations that make them inadequate for a high-density drone delivery service. First, complex cooperative strategies requiring real-time communication and resource-intensive training (*e.g.*, Deep Reinforcement Learning [13], [14], [15]) are not compared with simpler non-cooperative strategies (*e.g.*, geometric ones). Second, even complex solutions still suffer from higher collision rates, ranging from 10% to 40%. With such poor anti-collision performance using sophisticated techniques, there is no rationale for not including more straightforward solutions in the comparison. Third, they simulate a small number of drones, which does not adequately represent a drone delivery service in an urban setting. Fourth, many studies consider 2D scenarios in the evaluation for simplicity purposes, even though what makes a difference between aerial and terrestrial robots is the open 3D airspace.

Manuscript received 30 May 2023; revised 22 September 2023; accepted 24 October 2023. Date of publication 14 November 2023; date of current version 13 May 2024. This work was supported in part by the National Science and Technology Institute (INCT) of the Future Internet for Smart Cities funded by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) under Grant 465446/2014-0, in part by the Coordenacao de Aperfeiçoamento de Pessoal de Nível Superior—Brazil (CAPES)—under Grant Finance Code 001, in part by the Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) under Grant 14/50937-1 and Grant 15/24485-9, in part by FAPESP under Grant 2020/14771-2, and in part by CNPq under Grant 159565/2022-7. The Associate Editor for this article was L. Wang. *(Corresponding author: Fabíola M. C. de Oliveira.)*

Fabíola M. C. de Oliveira and Carlos A. Kamienski are with the Center of Mathematics Computing and Cognition, Federal University of ABC, São Paulo 09210-580, Brazil (e-mail: fabiola.oliveira@ufabc.edu.br).

Luiz F. Bittencourt is with the Institute of Computing, Universidade Estadual de Campinas (UNICAMP), São Paulo 13038-852, Brazil.

Reinaldo A. C. Bianchi is with the Department of Electrical Engineering, FEI University Center, São Paulo 09715-140, Brazil.

Digital Object Identifier 10.1109/TITS.2023.3329029

[1] https://www.insiderintelligence.com/insights/drone-delivery-services

[2] https://www.bloomberglinea.com/2022/01/25/flying-off-the-shelves-ifood-cleared-for-delivery-drone-take-off/

[3] https://wing.com

This paper proposes and evaluates a multidrone logistics scenario and three geometric approaches for collision avoidance: SingleDrone, MultiDrone, and SpeedDrone. We focus our analyses on non-collaborative strategies, where drones do not communicate and cooperate, which will always be needed for communication unavailability or emergency detours. Within non-cooperative strategies, geometric ones play a very important role due to their simplicity and suitability for dynamic, 3D, and outdoor environments. Other strategies, such as force-field ones [16], are appropriate for flight path planning in static environments as they rely on the mobility and geometry of drones and obstacles, which are unknown in advance in dynamic environments [10].

We conduct a performance analysis study using two techniques: first, we simulate different scenarios using the UTSim simulator; second, from the simulations, we obtain transition matrices for a Markov model. We consider using drones for individual missions and focus on individual autonomous (non-cooperative) decisions, assuming there will be a multitude of drones flying for different professional missions soon, including drone swarms, human transport, filming, surveillance, leisure, and delivery services. As our goal is to understand the fundamental tradeoffs in collision avoidance, we assume that drones do not communicate nor cooperate with each other. We compare our approaches with more straightforward baseline approaches, namely, DoNothing (let drones collide), Random (random detour), and Rightward (detour to the right, as in aviation [17]). Also, we compare all of them with the geometric approach proposed by Seo et al. [11], indicated by *SeoKimKimTsourdos* hereafter. We consider a 3D scenario and simulate hundreds of drones taking off due to delivery missions generated by constant and Poisson arrival rates. We also simulate drones randomly choosing among different altitudes for the cruise flight and include the Vector Sharing Resolution (VSR) approach [18].

Different simulation experiments consider seven fleet sizes on a constant drone arrival rate and four Poisson arrival rates. Our first counter-intuitive result is that the Random and Rightward approaches increase collisions compared with DoNothing because they make bad detouring decisions that shortly increase collision likelihood. Thus, we adopt the letting-drones-collide approach (DoNothing) as our baseline. Our approaches significantly reduce collisions in all situations. However, they decrease collisions but do not increase the number of successful trips for large numbers of active drones. As time passes, avoiding collisions without finishing trips increases the density, although the number of collisions remains constant. To improve our understanding of the asymptotic behavior of the strategies, which the simulations take time to achieve, we perform a numerical evaluation of a Markov chain model. The analytical modeling shows that our algorithms outperform the other ones in the stationary state. Also, since our algorithms are efficient in avoiding collisions, they take more time (*i.e.*, more transitions of the Markov chain) to achieve the stationary state. Finally, we consider cruise flights in two and five different altitudes to show that using more altitudes reduces collisions but does not eliminate them.

The main contribution of this paper is to propose and evaluate non-cooperative collision avoidance strategies for a high-density drone delivery service expected to be used as a baseline to analyze the improvement brought by more complex strategies. Non-cooperative strategies, even with 5G/6G and edge computing support, will always be needed when communication is unavailable or for emergency detours. The collision-free target needed for the viability of high-density drone delivery services is not yet achieved, neither by our strategies nor the strategies proposed in the literature. Our paper shows that our strategies result in a collision rate of 5% to 10%, which is better than the literature but still unacceptable for allowing a real-world high-density smart drone delivery service in large urban areas.

Our study deepens the understanding of the tradeoffs and limitations of more straightforward non-cooperative geometric approaches, which do not depend on heavy computational training and real-time communication. It is not worth using complex collision avoidance approaches, such as deep learning, when one does not understand if they perform better than simpler ones. Also, one must learn which success factors play a significant role in designing more sophisticated approaches that eliminate collisions. Before collision-free strategies are available, a drone delivery service will not thrive in big cities.

We organize this paper as follows. Section II presents the related work, and the proposed scenario is introduced in Section III. Section IV introduces our geometric approaches for collision avoidance. Section V presents the simulation and analytical modeling methodologies, whose results are presented in Section VI and Section VII, respectively. Finally, Section VIII discusses our findings, and Section IX concludes our work and draws possibilities for future work.

## II. Related Work

A challenge that hinders the existence of smart drone delivery services is the possibility of collision with stationary and moving obstacles [19]. Stationary obstacles are environmental objects such as trees, buildings, and mountains, whereas moving obstacles include flying objects like drones, birds, and other aircraft. Thus, the successful deployment of any delivery service based on drones depends on anti-collision strategies for population safety as the first and foremost priority.

Different classifications exist for anti-collision strategies, but typically they are considered a three-step process [10], [19], [20], [21] consisting of obstacle sensing, collision prediction (or detection), and collision avoidance (or resolution). The first step is sensing the obstacles, which may be performed individually by each drone or cooperatively by multiple drones [21]. In the latter case, drones must be able to exchange data via wireless technologies. Each drone senses the environment using different sensors, which include conventional ultrasonic sensors and cameras, but also Light Detection and Ranging (Lidar) [22]. Lidar is a technology and method to determine precise distances by emitting a pulsed laser beam on a surface and measuring the time the reflected light return to a laser detector, *i.e.*, the receiver [23]. The second step is collision prediction, where drones predict the likelihood of a collision after receiving information from obstacle sensing.

The prediction methods may be deterministic or nondeterministic. Deterministic methods use equations or algorithms previously defined, which, given a particular input, always produce the same output. They include certainty for trajectory fitting using fixed functions with a low computational burden but limited prediction precision [19]. On the other hand, nondeterministic methods are probabilistic or based on machine learning techniques. They continuously learn the obstacle movement pattern and can predict their future trajectory more precisely at the cost of higher computational overhead.

Finally, collision avoidance is the critical step for allowing drones for different applications in the future. As of the beginning of 2023, the literature contains classical and heuristic approaches for allowing drones to conclude their missions safely [19]. Classical approaches include geometric methods that use obstacle location and velocity and force-field methods that set an attraction force in the destination and a repulsion force in the obstacles to obtain an optimized path [10]. On the other hand, heuristic approaches include a variety of methods using techniques such as genetic algorithms, ant colony optimization, reinforcement learning, and deep learning. A recent trend is using deep reinforcement learning drones to learn on demand how to avoid obstacles [13], [24].

Our study differs from the literature in different aspects. The existing anti-collision technologies are mostly for ground vehicles [19]. For drones, most studies focus on drone swarms [25], [26], where multiple drones cooperate to overcome the limitations of single drones, dividing large tasks into smaller ones and carrying them out by a swarm. In this case, the flight paths of multiple cooperating drones can intersect during their mission and generate collisions, which must be avoided [24], [27], [28], [29]. Path planning is also a common strategy for avoiding collisions suitable for applications with static environment scenarios, which differ from our scenario that considers a dynamic, intelligent drone delivery service [30]. Many studies consider a constant altitude for simplicity [11], [13], [16], [31], *i.e.*, a 2D scenario, which we deem an oversimplification, as many collisions occur during takeoff, landing, and vertical detours. Finally, although many studies propose collision avoidance methods, they only evaluate scenarios with one or only a few drones. As considered in our paper, these limitations do not represent a real scenario of a future intelligent drone delivery service.

Geometric strategies compute the distance between the drone and the obstacle utilizing data such as the location and velocity of drones and obstacles. They typically have acceptable real-time performance and work in static and dynamic 3D environments, considering the dynamics of drones and obstacles and being an adequate choice for outdoor environments. Park et al. [18] proposed a geometric collision resolution logic called Vector Sharing Resolution (VSR). This logic calculates the closest point of approach to evaluate the worst condition between two drones and, by considering a minimum separation distance between them, decides the direction for both drones, which maneuver cooperatively. Nevertheless, this approach presents some limitations. Drones using VSR always detour vertically, which may be impossible due to drone altitude limits or when drones fly at different altitudes in cruise flights. Additionally, VSR considers that drones fly at constant velocities, and the authors evaluate the approach with only two drones.

Similarly, Seo et al. [11] proposed a geometric anti-collision strategy for single or multiple drones (swarms), using a collision prevention envelope that considers the drone angular rate and object detection range limits to indicate whether a detour is possible. Also, they proposed a collision avoidance strategy that considers multiple obstacles by defining a detour angle concerning all obstacles risking collision with a particular drone. However, this study has critical limitations, such as a constant speed magnitude and direction of drones and obstacles, evaluation in a 2D scenario only, with both drones and obstacles flying at the same altitude, and with a maximum of two drones. Chen et al. [1] proposed DroneTalk, an IoT-based drone delivery system for mail delivery in internal settings. To avoid collisions, DroneTalk uses a geometric strategy based on the simultaneous calculation of communication latencies, processing, and control. The system defines a maximum safe speed according to latencies and distance between the drone and the obstacles ahead in the path. However, the simulations considered only one drone.

Force-field strategies are inspired by attractive or repulsive electric forces among charged objects. Choi et al. [16] proposed a force-field approach for static and dynamic environments, replacing the obstacle axial repulsion force with circular repulsion. This change removes the possibility of aircraft getting caught in local minima and generates more efficient paths than similar strategies. However, their evaluation only considers three drones in a 2D scenario. Sun et al. [32] proposed a specific force-field strategy for drone swarms, where only drones in the swarm are dynamic obstacles. Force-field strategies are appropriate for flight path planning in static environments as they rely on the mobility and geometry of drones and obstacles.

In machine learning, deep reinforcement learning (DRL) has been successfully used for applications such as anti-collision strategies, where individual drones interact with the environment and learn how to make optimized decisions based on historical experience [12]. Usually, such systems need: a) long-term instead of one-time performance optimization; b) real-time decision-making and learning without prior knowledge instead of predetermined knowledge gathering; and c) high scalability. Wang et al. [14] proposed a decentralized DRL approach for drone collision avoidance based on a two-stage training method and considering imperfect sensing, *i.e.*, they added noise in the obstacle sensing data. The authors simulated simple 2D and 3D scenarios with up to 200 drones and obtained a minimum collision rate of 5%. Ouahouah et al. [13] proposed a DRL strategy for collision avoidance considered unaware of the scenario scale. However, their simulation scenario is simple, considering only one observable drone with constant velocity and altitude and a few mobile obstacles using random walk mobility. Their best results of collision rate are 10%. Thumiger and Deghat [15] also proposed a deep reinforcement learning approach for

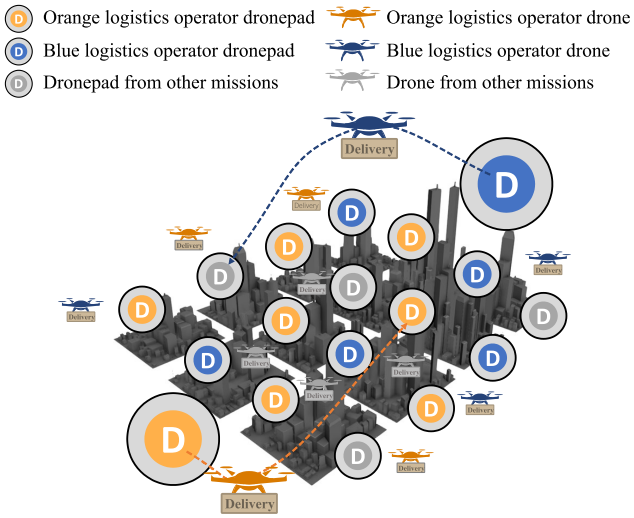



Fig. 1. Smart drone delivery service scenario.

drone collision avoidance, achieving a minimum collision rate of 7.5% for 12 simultaneous drones. Ourari et al. [24] proposed a specific reinforcement learning approach for drone swarms, which differs from our approach, which considers the decisions of individual drones.

## III. Aerial Delivery Services

The scenario in Fig. 1 illustrates an aerial order delivery service using drones, considering two delivery logistics operators (orange and blue), besides drones that fulfill other unidentified missions (gray). There are two merchandise distribution centers (DC) for the orange and blue operators, where takeoffs to initiate deliveries and landings in return to the DC occur in dronepads, identified by the letter D. Each logistics operator may have one or more merchandise DCs and performs the deliveries in smaller dronepads scattered throughout the city. A real scenario may include several delivery service operators with drones [33], drones carrying out sporadic missions [34], besides drones for the transport of people [35].

The main challenge in a drone-based delivery service is to avoid collisions [36], which may involve other drones, flying objects, birds, or static obstacles such as buildings and trees. Like aircraft, the highest risks concentrate near dronepads during landings and takeoffs [37]. However, contrary to the centralized air traffic control system, a delivery scenario containing several logistics operators lacks global centralized coordination. For this reason, the operators can implement optimized landing and takeoff sequencing systems at their DC dronepads. Nevertheless, drones must be able to avoid cruising collisions, that is, outside the landing and takeoff regions, in the operator DC or the delivery dronepads.

Fig. 2 illustrates four typical situations where collisions between drones may occur. These situations may often occur if no drone takes action to avoid collisions, even in a scenario with few drones. In Fig. 2a, a drone taking off from a dronepad is on a collision course with a drone that is flying over the dronepad, and, in Fig. 2b, a drone landing on the dronepad is on a collision course with another drone that flies over

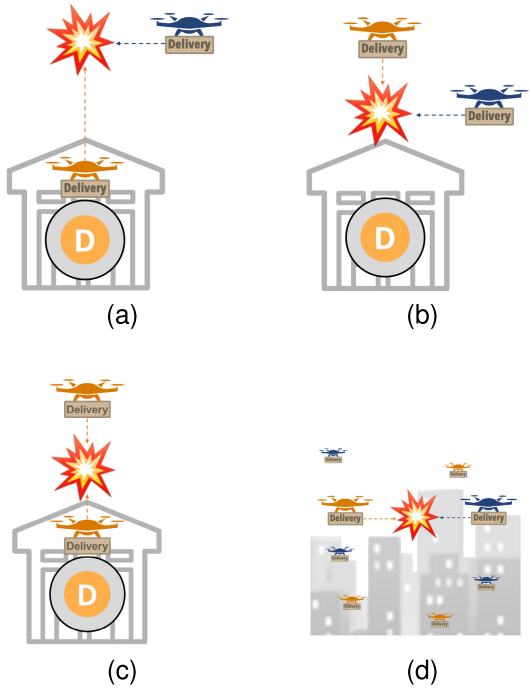



Fig. 2. Different collision situations between drones. (a) Takeoff: different operators. (b) Landing: different operators. (c) Landing and takeoff: same operator. (d) Cruise: different operators.

the dronepad. In both pictures, the drones belong to different operators, as the same operator could avoid such situations through centralized control. A solution to collisions between drones from different operators may be establishing forbidden regions, like airports in aviation. Fig. 2c represents a situation in which two drones from the same operator may collide while they take off from and land on the same dronepad due to a lack of dronepad sequencing control. Finally, the scenario requiring more sophisticated collision avoidance approaches is illustrated by Fig. 2d, where two drones from different operators fly on a collision course while on a cruise flight. Although this picture contains drones from different operators, without centralized control or communication, this situation may also happen to drones from the same operator. Here we assume no centralized control for collision avoidance, no takeoff and landing sequencing in the dronepads, or communication between drones. The drones make individual detour decisions based on data provided by their sensors.

## IV. GeoDrone Approaches

This paper proposes three geometric collision avoidance approaches inspired by the strategy that considers multiple obstacles proposed by Seo et al. [11] (SeoKimKimTsourdos algorithm). First, we explain the characteristics common to the three approaches and turn to the individual features afterward.

### A. General Characteristics

Our approaches work for static and dynamic obstacles and are divided into two stages: collision detection and collision avoidance, so the drone must detour for every detected collision. The drone detection radius $d_r > 0$ indicates the size of a sphere around the drone inside which the drone

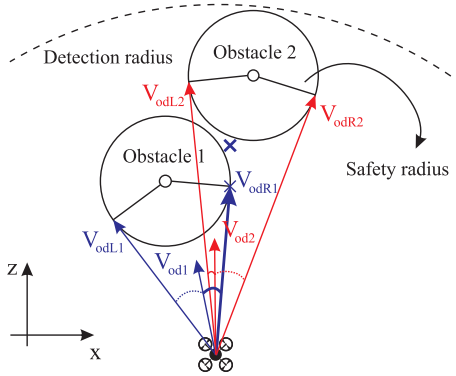



Fig. 3. Detour considering multiple obstacles in our proposals (MultiDrone and SpeedDrone).

sensors identify any obstacle. A drone considers an obstacle for collision detection when the distance between an object and a drone is less or equal to the drone detection radius, and the angle $\gamma$ between the relative velocity vector between the drone and the obstacle ($\mathbf{V_{od}}$) and the line-of-sight vector ($\mathbf{v_{los}}$), *i.e.*, the vector between the drone and the obstacle, is between $-90$ and 90 degrees:

$$||p_d - p_o|| \leq d_r$$
$$-\pi/2 \leq \gamma \leq \pi/2,$$

in which $p_d$ is the drone position, and $p_o$ is the obstacle position. Our approaches use the obstacle safety radius $s_r > 0$, which indicates the size of a sphere around the obstacle, to detect a possible collision. Fig. 3 shows the safety radius for two obstacles. A possible collision is detected if the relative velocity vector between the drone and the obstacle points toward the safety radius:

$$|\mathbf{v_{los}}| \sin\phi < s_r,$$

$$\phi = \begin{cases} \pi/2, & \text{if } \mathbf{v_{los}} = 0 \\ & \text{and } \mathbf{V_{od}} = 0 \\ \arccos\left(\dfrac{\mathbf{v_{los}}.\mathbf{V_{od}}}{|\mathbf{v_{los}}|.|\mathbf{V_{od}}|}\right), & \text{otherwise.} \end{cases}$$

In the detour step, our approaches draw a cone whose radius equals the safety radius around each obstacle under the collision route; this is the collision cone in Fig. 3. This figure shows two moving obstacles under a collision route to a drone. The drone must detour along one of the drawn cone generators (blue $V_{odL1}$, $V_{odR1}$, red $V_{odL2}$, or $V_{odR2}$). Our approaches calculate the detour angle between the possible directions for a detour and the relative velocity between the drone and each obstacle ($V_{od1}$ and $V_{od2}$). The chosen direction depends on the approach, and we explain it ahead. Our approaches calculate if it is possible to detour horizontally depending on the drone angular rate, or maneuverability ($h_{ang}$), according to

$$d_r \cos\phi - \sqrt{s_r^2 - d_r^2 \sin^2\phi} - |\mathbf{V_{od}}| \frac{\arcsin\left(\frac{s_r}{d_r}\right) - \left|\arccos\left(\frac{\mathbf{v_{los}}.\mathbf{V_{od}}}{|\mathbf{v_{los}}|.|\mathbf{V_{od}}|}\right)\right|}{h_{ang}} > 0,$$

and if not, the drone detours vertically. If it is impossible to detour in any plane, the drone will collide while detouring vertically. The detour ends when the relative velocity vector is orthogonal to the line-of-sight vector of the tangent obstacle, *i.e.*, when the projection of the relative velocity over the line-of-sight vector is lower than 0.12:

$$\text{Proj}_{\mathbf{V_{od}}}\mathbf{v_{los}} = \frac{||\mathbf{v_{los}}||.||\mathbf{V_{od}}|| \cos\phi}{||\mathbf{V_{od}}||^2}.\mathbf{V_{od}} < 0.12,$$

which results in the thinner cross in Fig. 3. We include a tolerance for the detour termination condition due to the simulation sampling frequency. While, in a time step, the projection magnitude is slightly greater than zero, in the next step, this magnitude may reach zero and be positive again. In this situation, the detour would never end. Based on the simulations, we use the Unity standard sampling frequency of 0.02 and 0.12 for the tolerance in the detour termination condition. We apply these characteristics to any detour approach implemented in UTSim, including SeoKimKimTsourdos.

The detours are performed by including new temporary destinations in the drone flight plan instead of a detour angle. Our approaches include an emergency detour, defining a detour angle whenever an obstacle invades the safety sphere. The drone detours 180°, decreasing its velocity to zero and flying in the opposite direction. We choose this value based on the simulations, in which Unity considers the drone dynamics. SeoKimKimTsourdos does not include an emergency detour, so we apply this characteristic in its implementation.

UTSim drones have different angular rates according to the destination. Thus, we fix 150°/s, which is a typical angular rate in commercial drones [38]. We set the longitudinal axis destination ahead along the collision cone generator to guarantee the 150°/s angular rate. Fig. 3 shows the corresponding destination as the thicker cross, although the drone terminates the detour when it arrives at the thinner one.

### B. SingleDrone, MultiDrone, and SpeedDrone

Three characteristics differentiate our proposals. The first one is considering one or multiple obstacles when detouring. While SingleDrone detours from the obstacle that first gets on a collision course with the drone, MultiDrone and SpeedDrone detour from all colliding obstacles. SingleDrone chooses the smallest detour angle so the drone spends less energy detouring. In Fig. 3, if Obstacle 1 is the first obstacle that enters a collision route to the drone, SingleDrone chooses $V_{odR1}$. MultiDrone and SpeedDrone first choose the smallest detour angle for each obstacle, represented by the full line angles in Fig. 3, and the most significant detour angle among the angles chosen in the previous step ($V_{odR1}$).

The case of Fig. 3 rarely happens in our scenario simulations because the obstacles should enter the drone detection radius at the same time step. Therefore, MultiDrone and SpeedDrone detour from the obstacle that first enters a collision route. If there are still colliding obstacles after finishing this detour, the drone takes another detour related to the closest obstacle under a collision route. As this situation is rare, we also prioritize the closest colliding obstacle in the

SeoKimKimTsourdos implementation, aiming to standardize the simulation comparisons.

The other differences between our proposals are the initial velocity and the velocity when a drone detects a collision. SeoKimKimTsourdos uses constant and equal velocity for all drones, producing deadlocks in our scenario simulations for a constant drone arrival rate. This result motivates us to change the velocities in our approaches to remove the synchrony among the drones and avoid deadlocks. We also propose to change velocities to reduce the number of collisions. On the one hand, MultiDrone starts with the maximum drone velocity and does not reduce it if the drone detects a collision. SingleDrone also starts with the maximum drone velocity but reduces it by half if a collision can occur. On the other hand, SpeedDrone starts with a random velocity between the maximum and half this value and reduces it to a random value between its current velocity and half this value when the drone detects a collision. Our approaches work for any positive maximum velocity of drones.

### C. Algorithms

Algorithms 1, 3, and 4 show the pseudocode for SingleDrone, MultiDrone, and SpeedDrone, respectively. They call the functions *GenerateDrones()*, *DefineDetourAngle()*, *Detour()*, and *TerminateDetour()* in Algorithm 2, which are common to the three approaches.

Algorithm 1 implements *GenerateSingleDrone()* (lines 2–5) and *SingleDrone()* (lines 7–15). *GenerateSingleDrone()* sets the drone velocity as its maximum velocity (line 3) and calls *GenerateDrones()* (line 4) in Algorithm 2, which creates a drone in the simulation according to the drone arrival rate (line 3 in Algorithm 2). UTSim calls *SingleDrone()* whenever an obstacle is within a drone detection radius. *SingleDrone()* checks if a collision is detected and the drone is not detouring (line 8). If both conditions are satisfied, the algorithm sets the velocity to half the maximum velocity (line 9) and calls *DefineDetourAngle()* (line 10). *DefineDetourAngle()* checks if the distance between the drone and the obstacle is smaller than the safety radius (line 7 in Algorithm 2). If so, the algorithm sets the detour angle to 180° (line 8); else, it sets the detour angle to touch the obstacle safety radius (lines 9–10). Next, *SingleDrone()* picks the smallest angle of the first obstacle, choosing to detour to its left or right (line 11 in Algorithm 1), and calls *Detour()* (line 12). *Detour()* checks if the drone can detour horizontally (line 15 in Algorithm 2), defining the detour in the transversal axis as the SeoKimKimTsourdos transversal detour (line 16) and the detour in the longitudinal axis as the transversal detour divided by the angular rate to guarantee the drone angular rate (line 17). Suppose the drone cannot detour horizontally (line 18). In that case, the algorithm defines a vertical detour choosing the SeoKimKimTsourdos detour in the vertical axis (line 20) and the detour in the longitudinal axis as the vertical detour divided by the angular rate (line 21). Finally, *SingleDrone()* calls *TerminateDetour()* (line 14 in Algorithm 1), which checks if the drone is detouring (line 26 in Algorithm 2). If so, the algorithm calculates the relative velocity between the drone and the obstacle (line 27) and checks if the projection between the relative velocity vector and the obstacle line of sight is smaller than 0.12 (line 28). If so, the detour terminates (line 29), and the drone velocity returns to the initial value (line 30).

Algorithm 3 also implements two functions, *GenerateMultiDrone()* (lines 2–5) and *MultiDrone()* (lines 7–15). *GenerateMultiDrone()* sets the drone velocity as its maximum velocity (line 3) and calls *GenerateDrones()* (line 4). *MultiDrone()* checks for a detected collision and if the drone is not detouring (lines 8–13) and, if so, calls *DefineDetourAngle()* (line 9). After that, the algorithm picks the smallest angle for each colliding obstacle (line 10) and the largest angle among the ones chosen in the previous step (line 11). Finally,

**Algorithm 1** SingleDrone

```
1: /* Drone generation and initialization */
2: function GenerateSingleDrone(arrivRate, maxV)
3:     velocity ← maxV;
4:     GenerateDrones(arrivRate);
5: end function
6: /* SingleDrone() initiates or terminates a detour */
7: function SingleDrone(Obstacle, maxV, safeR, angRate)
8:     if collision is detected AND drone is not detouring then
9:         velocity ← maxV/2;
10:        DefineDetourAngle(Obstacle, safeR);
11:         Pick the smallest angle of the first obstacle;
12:        Detour(Obstacle, safeR, angRate);
13:    end if
14:    TerminateDetour(Obstacle, maxV);
15: end function
```

**Algorithm 2** Code in Common for Collision Avoidance

```
1: /* Drone generation */
2: function GenerateDrones(arrivalRate)
3:     Generates a drone according to arrivalRate;
4: end function
5: /* Detour angle definition */
6: function DefineDetourAngle(Obstacle, safeRadius)
7:     if Distance(drone, Obstacle) < safeRadius then
8:         detourAngle ← 180°;
9:     else
10:        detourAngle ← angle to touch safeRadius;
11:    end if
12: end function
13: /* Detour start */
14: function Detour(Obstacle, safeRadius, angRate)
15:    if a horizontal detour is possible then
16:        det.pitch ← SeoKimKimTsourdos pitch detour;
17:        det.roll ← det.pitch / angRate;
18:    else
19:        /* Vertical detour */
20:        det.yaw ← SeoKimKimTsourdos yaw detour;
21:        det.roll ← det.yaw / angRate;
22:    end if
23: end function
24: /* Detour termination */
25: function TerminateDetour(Obstacle, initialVel)
26:    if the drone is detouring then
27:        relVel ← RelativeVelocity (drone, Obstacle);
28:        if Projection(relVel, lineOfSight) < 0.12 then
29:            Terminates detour;
30:            velocity ← initialVel;
31:        end if
32:    end if
33: end function
```

*MultiDrone()* calls *Detour()* (line 12) and *TerminateDetour()* (line 14).

**Algorithm 3** MultiDrone

```
1: /* Drone generation and initialization */
2: function GENERATEMULTIDRONE(arrivRate, maxV)
3:     velocity ← maxV;
4:     GENERATEDRONES(arrivRate);
5: end function
6: /* MultiDrone() initiates or terminates a detour */
7: function MULTIDRONE(Obstacle, maxV, safeR, angRate)
8:     if collision is detected AND drone is not detouring then
9:         DEFINEDETOURANGLE(Obstacle, safeR);
10:        Pick the smallest angle for each colliding obstacle;
11:        Pick the largest angle chosen in the previous step;
12:        DETOUR(Obstacle, safeR, angRate);
13:    end if
14:    TERMINATEDETOUR(Obstacle, maxV);
15: end function
```

Algorithm 4 implements *GenerateSpeedDrone()* (lines 2–5) and *SpeedDrone()* (lines 7–16). *GenerateSpeedDrone()* sets the velocity and initial velocity as a random number between the maximum velocity and half of it (line 3) and calls *GenerateDrones()* (line 4). If the algorithm detects a collision and the drone is not detouring (line 8), *SpeedDrone()* chooses another random velocity for the drone between the current velocity and half of it (line 9). *SpeedDrone()* calls *DefineDetourAngle()* (line 10), picks the smallest angle for each colliding obstacle (line 11), picks the largest angle chosen in the previous step (line 12), and calls *Detour()* (line 13). Finally, *SpeedDrone()* calls *TerminateDetour()* (line 15).

**Algorithm 4** SpeedDrone

```
1: /* Drone generation and initialization */
2: function GENERATESPEEDDRONE(arrivRate, maxV)
3:     velocity ← initVel ← random(maxV/2, maxV);
4:     GENERATEDRONES(arrivRate);
5: end function
6: /* SpeedDrone() initiates or terminates a detour */
7: function SPEEDDRONE(Obstacle, initVel, safeR, angRate)
8:     if collision is detected AND drone is not detouring then
9:         velocity ← random(velocity/2, velocity);
10:        DEFINEDETOURANGLE(Obstacle, safeR);
11:        Pick the smallest angle for each colliding obstacle;
12:        Pick the largest angle chosen in the previous step;
13:        DETOUR(Obstacle, safeR, angRate);
14:    end if
15:    TERMINATEDETOUR(Obstacle, initVel);
16: end function
```

## V. METHODOLOGY

Our performance analysis adopts a methodology initially based on simulations, which we extend by analytical results using Markov Chains.

### A. Simulation Methodology

We execute the simulations in our modified version of UTSim. The scenarios aim to assess the performance of the algorithms under a high density of drones, representing a more critical scenario than Fig. 1. Fig. 4 shows the superior view of the simulated scenario. The orange circles represent 100 distribution centers (DC), arranged every 40 m along a square of 1 km and from where drones take off and land. Drones deliver in an area of 0.81 km$^2$ concentric to the square where the DCs are located so that all delivery points have a safe distance of at least 50 m from all DCs. The blue region represents the delivery area. There are no pre-established airways; drones can fly freely at different altitudes, latitudes, and longitudes.

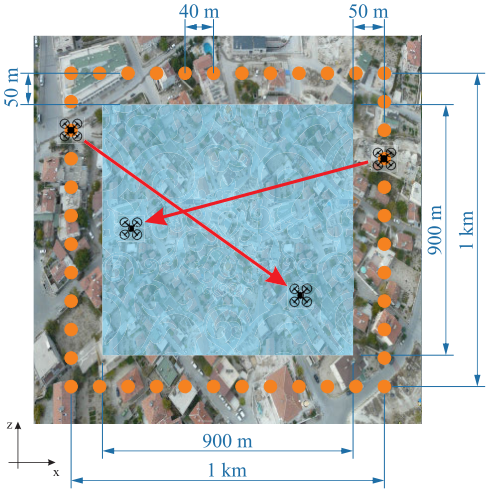



Fig. 4. Superior view of the simulated scenario.

We use 100 DCs spread over the 1 km x 1 km area to represent a small neighborhood with many small vendors (such as restaurants). For evaluating the high-density drone delivery service with geometric collision avoidance strategies, we intend to distribute the drones over the area to the greatest extent with multiple source and destination places to keep the drone density fairly constant. In preliminary simulations, we identified that the most influence on the ability to make good detour decisions comes from drone density and not the size of the area. We identified that large distribution centers (*e.g.*, e-commerce) where many drones take off and land simultaneously cause collisions that could be avoided with a centralized drone sequencing strategy. Therefore, we aim to isolate the collisions when drones make individual autonomous decisions from those in crowded hotspots managed by a single organization [39].

Drones in the simulation are 2 m × 2 m × 0.5 m so that they can transport large loads and do not just perform the last-mile delivery, that is, the end route in the delivery of an order. They present a maximum velocity of 20 m/s, a safety radius of 30 m, and a detection radius of 100 m. We consider Lidar sensors, which can present high precision and accuracy up to 100 m. Each drone delivers only one order and returns to the takeoff DC. The trip contains initially 10 steps: 1) initialization; 2) definition of the initial velocity; 3) takeoff at an altitude of 30 m; 4) trip to a randomly chosen destination within the area of 0.81 km$^2$ indicated in Fig. 4; 5) landing at an altitude of 1 m; 6) wait of 10 s for the delivery; 7) takeoff at an altitude of 30 m; 8) trip to the takeoff DC; 9) landing at an altitude of 1 m; and 10) finalization.

When a drone decides to detour, it includes temporary destinations to the initial trip. The drones fly at an altitude of 30 m due to the less time spent in the landings and takeoffs, which decreases the total simulation time. Moreover, using only one altitude increases the density of drones on cruise flights, making the scenario more challenging for collision avoidance. Although the drones fly at only one altitude, they can detour vertically to a higher or lower altitude, returning to the original altitude of 30 m after the detour. We simulate scenarios where drones randomly choose one from two and one from five different altitudes for the cruise flight, starting from 30 m and increasing 10 m for each altitude. These simulations tend to lead to fewer collisions for all approaches since drones have fewer opportunities to collide as they are at different altitudes. Although we can increase the drone density with more altitude levels, this paper aims to simulate the worst possible scenario for cruising collisions, which is the one with all drones flying at the same altitude and increasing the opportunities for collisions.

We simulate two types of drone arrival rates: constant and Poisson. Initially, we study the behavior of the approaches by increasing the number of drones at a constant rate. As more drones take off, their density increases until the system reaches the stationary time, *i.e.*, an interval whose properties do not change over time. In the stationary time, the accumulated number of collisions increases proportionally to the simulation time, which takes longer due to the higher number of drones. Further, we change the simulations to Poisson arrival rates to achieve higher control over the drone takeoffs and compare all approaches with the same density. We use a low, medium, high, and excessively high rate, which increases the density (*i.e.*, the system load). To measure the collisions at Poisson rates, we wait for the simulation to reach a stationary time and simulate each approach for a fixed time. Thus, the experiments are complementary since, in the first one, we fix the rate and simulate for a variable time, and in the second, we vary the rate and simulate for a fixed time.

For the constant rate, we simulate seven fleets with increasing sizes, with 2, 5, 10, 20, 50, 100, and 200 drones, which take off sequentially every 10 s from a randomly chosen DC. The simulation ends when one of the following conditions is satisfied: 1) all drones end their trips or collide, or 2) the three-hour simulation time is reached. We accelerate the simulation time scale three times related to the wall-clock time. Thus, one clock hour equals three hours of simulation. If the simulation time is reached, we interrupt the simulation and consider that at least one deadlock occurred since the slowest simulations end in 12.7 min on average (wall time). A deadlock occurs when two or more drones get stuck, flying away from the delivery area or rotating around an axis indefinitely. In the Poisson arrival rate, drones take off at 2, 4, 6, and 12 drones/min. We simulate 1000 s for the system to reach the stationary time and 30 min in the stationary time.

We simulate eight collision avoidance strategies: DoNothing, Random, Rightward, SeoKimKimTsourdos, VSR, SingleDrone, MultiDrone, and SpeedDrone. DoNothing does not change the initial trip. Random and Rightward detour when an obstacle is 100 m or less from a drone. The Random strategy detours between −40 m and 40 m horizontally related to the drone position in the x and z axes. Rightward horizontally detours 20 m related to its position in the x and z axes, *i.e.*, 28.3 m at 45° to the right. While we simulate DoNothing to know the scenario collision conditions, Random is a naive strategy, and Rightward is inspired by the collision avoidance rules in aviation [17]. For the other strategies, when an obstacle is 100 m or less from a drone, the drone calculates if there can be a collision and detours if so. SeoKimKimTsourdos employs the strategy that considers multiple obstacles proposed by Seo et al. [11]. As VSR always makes vertical detours and does not limit the altitude, we only simulate it when evaluating the impact of multiple altitudes. In this case, we limit VSR detour altitudes to 120 m and 80 m, since 120 m is a common maximum altitude for drones in regulation policies [40], and 80 m leads to a mean altitude closer to the other approaches, besides allowing 10 m beyond the maximum cruise altitude of 70 m when using five altitudes. Finally, SingleDrone, MultiDrone, and SpeedDrone are our proposals. We execute each scenario and collision avoidance strategy 30 times and report the mean of each measure and the 95% confidence interval.

### B. Analytical Modeling Methodology

After the simulations, we collect the number of drones transitioning to different flight stages and model it as a Markov chain. In this model, each node represents a state, and each edge represents a transition to another state that can occur with a certain probability. Each state can have one or more transitions to other states. Therefore, we can formally define a Markov chain as a stochastic model with state space $\chi$, where transitions from any state $x$ to any state $y$ occur according to a $\chi\chi$ matrix $P$ [41]. Markov Chains present the memoryless Markov property meaning that the probability of proceeding from state $x$ to state $y$ is the same, no matter what sequence $x_0, x_1, \cdots, x_{t-1}$ of states precedes the current state $x$.

Fig. 5 depicts the Markov chain for our drone delivery service, with descriptions in Table I. It has 14 states representing the possible flight phases, starting with drones waiting for delivery trips in the distribution center (S0 - D-DC) and ending with successful trips where drones return safely and land in the distribution center (S12 - R-DC). The states are symmetrical for the delivery (D) and return (R) trips involving takeoff (TO), cruise (CR), landing (LD), detour (DT), deadlock (DL), and collision (S13). The latter can happen in all states where drones are flying. After the drones land in the delivery dronepads, the deliveries are performed (S6 - D-CT), and the drones initiate the return trip by taking off again.

The state transition probabilities for our model are extracted from the UTSim simulations shown in Section VI. We create a transition matrix for each algorithm and drone Poisson arrival rate to generate numerical results. The performance analysis based on the Markov chain is performed by multiplying the initial state matrix (probability 1 for $S0$ and 0 for all other states) by the transition matrix, varying the number of state transitions. This process yields two types of results: a) the stationary state, *i.e.*, the asymptotic final probabilities

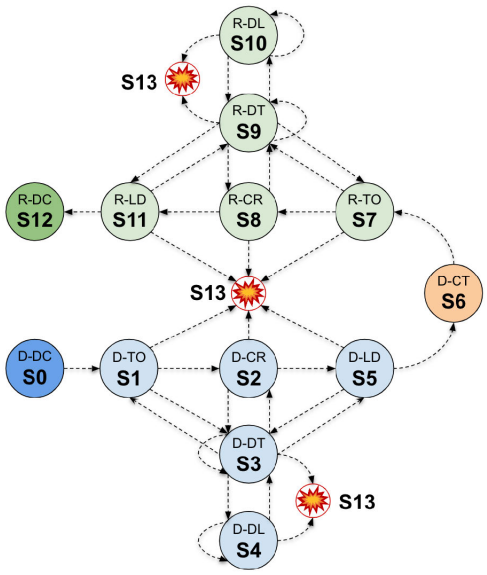



Fig. 5. Drone delivery service modeled as a Markov chain.

TABLE I
STATES IN THE DRONE DELIVERY MODELING

| State | Acronym | Description |
|---|---|---|
| S0 | D-DC | Delivery Trip - Distribution Center |
| S1 | D-TO | Delivery Trip - Takeoff |
| S2 | D-CR | Delivery Trip - Cruise |
| S3 | D-DT | Delivery Trip - Detour |
| S4 | D-DL | Delivery Trip - Deadlock |
| S5 | D-LD | Delivery Trip - Landing |
| S6 | D-CT | Delivery Trip - Customer |
| S7 | R-TO | Return Trip - Takeoff |
| S8 | R-CR | Return Trip - Cruise |
| S9 | R-DT | Return Trip - Detour |
| S10 | R-DL | Return Trip - Deadlock |
| S11 | R-LD | Return Trip - Landing |
| S12 | R-DC | Return Trip - Distribution Center |
| S13 | | Collision |

for all states (the final probability matrix), where a new multiplication generates the same transition matrix (we execute 700 state transitions); and b) the evolution of the probabilities with increasing numbers of state transitions (up to 700), *i.e.*, the intermediate state matrices before the stationary state is reached. Section VII presents the analysis of the results provided by this model.

## VI. Simulation Results

This section presents the results for the constant and Poisson drone arrival rates and for several altitudes in the cruise flight.

### A. Constant Drone Arrival Rate

Fig. 6 shows the mean collision rate and 95% confidence interval for seven fleet sizes. Although SeoKimKimTsourdos is a sophisticated geometric collision avoidance approach, there are no statistical differences between it and DoNothing, and they present collision rates between 0% and 9.17%. Overall, Random and Rightward yield the highest collision rates in the simulations. Random presents higher collision rates than DoNothing for 50 or more drones, between 19.47% and 21.40%, while Rightward presents higher collision rates than DoNothing for 20 or more drones, between 12.33% and 25.03%. These results show that simple approaches are ineffective in our high-density scenario and increase the collision rate compared with a not detouring approach.

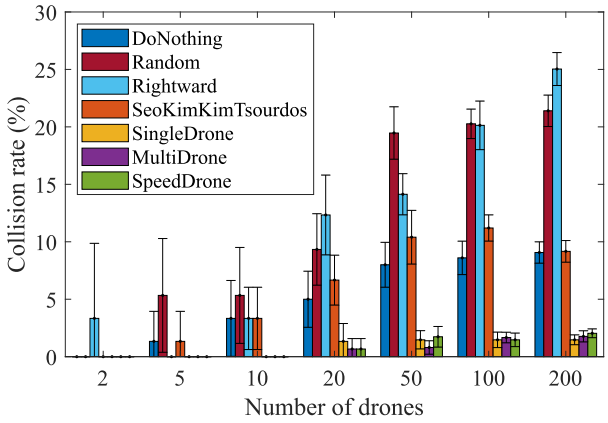



Fig. 6. Collision rate at constant arrival rate.

In aviation, we expect a small number of aircraft under collision routes due to their low density and the use of airways. Thus, Rightward can work in aviation but may not work for a scenario with a high density of drones without airways. A similar situation occurs with SeoKimKimTsourdos. Seo et al. [11] presented results without collisions in scenarios with a few drones. Therefore, there are no statistical differences in the collision rate between SeoKimKimTsourdos and DoNothing, since it was not designed for a high-density scenario. As Random and Rightward substantially increase the collisions compared with DoNothing, DoNothing is our baseline, and the results for the collision details do not include these strategies.

Our approaches eliminate the collisions of up to 10 drones. They present collision rates between 0 and 2.03% for all fleets, being the approaches with the lowest collision rates among the tested approaches. The result in Fig. 6 shows the efficacy of our approaches and that they are the best approaches for collision avoidance in the scenario of aerial deliveries among the tested approaches. However, our approaches have no statistical difference in the constant drone arrival rate.

Fig. 7 shows the number of collisions in the four collision situations of Fig. 2 for the 200-drone fleet. Most collisions in DoNothing and SeoKimKimTsourdos occur when both drones are on the cruise flight, as seen in the collision situation in Fig. 2d. These approaches have no statistical difference, although SeoKimKimTsourdos presents more collisions on average. Our approaches significantly reduce the collisions in this situation, between 4.58 and 11 times compared with DoNothing and 5.97 and 14.33 times compared with SeoKimKimTsourdos. This result shows that, as designed, our approaches present a significant reduction in cruise collisions compared with DoNothing and SeoKimKimTsourdos for the proposed scenario.

Our approaches eliminate collisions when one drone takes off and the other is on a cruise flight, as represented in Fig. 2a. On the other hand, SeoKimKimTsourdos significantly

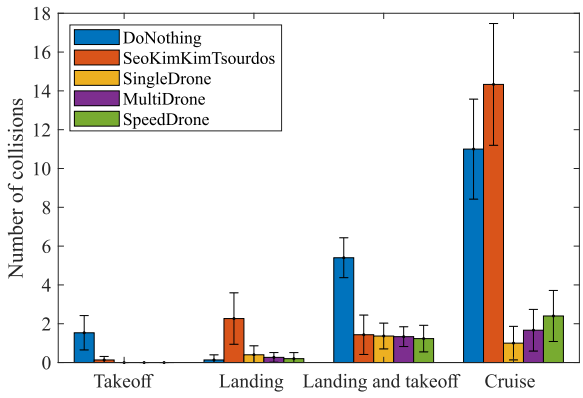



Fig. 7. Number of colliding drones per collision situation for 200 drones.

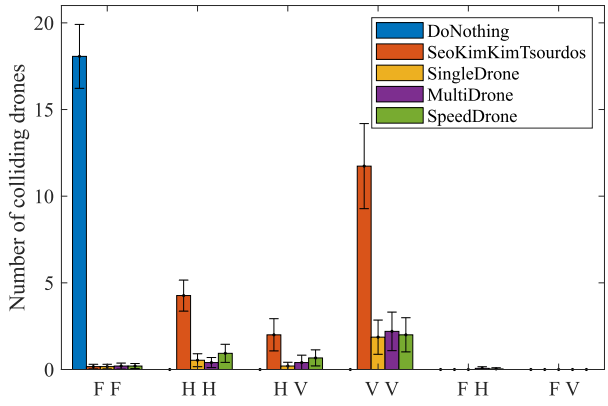



Fig. 8. Number of colliding drones per detour state for 200 drones.

increases collisions when one drone is landing and the other is on a cruise flight (Fig. 2b). DoNothing and our approaches do not present statistical differences in this situation. All the approaches that detour significantly reduce collisions compared with DoNothing when a drone lands and the other takes off (Fig. 2c). However, there is no significant statistical difference between the detour approaches. Although we do not design our approaches to reduce collisions in the situations with landings and takeoffs (the cases of Fig. 2a, 2b, and 2c), they reduce collisions in some of these situations. This result shows another advantage for the adoption of our approaches.

Fig. 8 shows the colliding drone detour state for the fleet with 200 drones. F means flying, *i.e.*, not detouring, H means detouring horizontally, and V means detouring vertically. All collisions in DoNothing occur when both drones are not detouring, as expected. Most collisions occur when both drones detour vertically for all detour approaches. These drones probably try to avoid a collision between themselves. In this state, our approaches reduce the collisions between 5.33 and 6.29 times compared with SeoKimKimTsourdos, although our approaches do not present statistical differences. The drone angular rates may explain the collisions since the horizontal angular rate equals the vertical angular rate. If the angular rates are equal and a horizontal detour is impossible, a vertical detour is probably also impossible.

SeoKimKimTsourdos presents significant collisions when both drones detour horizontally. In this state, drones also probably try to avoid a collision between themselves.

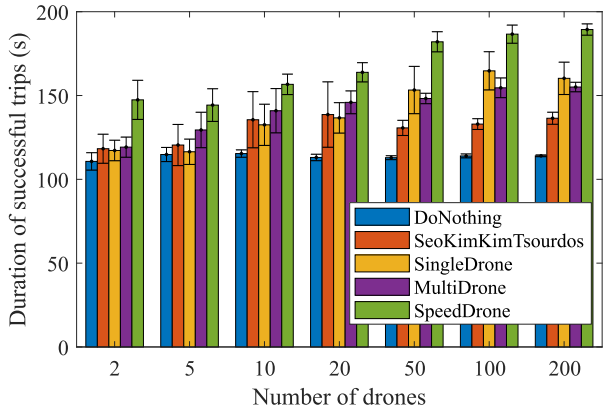



Fig. 9. Duration of successful trips.

SeoKimKimTsourdos also collides significantly when one drone detours horizontally and the other detours vertically. This state indicates that the drones collide when trying to avoid a collision with other drones, that is, there is at least a third drone in the collision scenario. These states show the limitation of the algorithm proposed by Seo et al. [11]. There are almost no collisions when one drone is not detouring and the other is detouring horizontally or vertically. These results indicate that collisions occur when all drones involved in the collision detour or all drones do not detour, *i.e.*, either all drones detect a collision or no drone detects it.

Fig. 9 shows the duration of the successful trips, *i.e.*, the trip duration of the drones that performed a delivery and returned to the original DC. The duration of successful trips shows the efficiency of the collision avoidance approaches since detouring increases the trip duration and impacts the drone battery time, which may prevent the drone from delivering or returning to the DC. DoNothing presents the lowest duration of 113.5 s on average since it does not detour. Our approaches increase the duration of successful trips for almost all fleets, between 1.01 and 1.45 times longer than DoNothing. SpeedDrone increases this duration the most, between 1.11 and 1.26 times longer than SingleDrone and MultiDrone, causing them to be preferable among our approaches. SingleDrone and MultiDrone do not present statistical differences.

The simulation execution time for the fleet with 200 drones takes 819.33 $\pm$ 11.59 s for DoNothing, 1504.23 $\pm$ 513.24 s for SeoKimKimTsourdos, 1083.27 $\pm$ 195.16 s for SingleDrone, 836.9 $\pm$ 187.52 s for MultiDrone, and 981.97 $\pm$ 320.15 s for SpeedDrone. While DoNothing, MultiDrone, and SpeedDrone are statistically equal for the fleet with 200 drones, SingleDrone takes 1.32 times longer than DoNothing and is statistically equal to SeoKimKimTsourdos. The higher execution times and confidence intervals for SeoKimKimTsourdos and our approaches are due to deadlocks. SeoKimKimTsourdos produces the most deadlocks, starting from four deadlocks in the fleet with 20 drones and reaching seven deadlocks in the 100- and 200-drone fleets. SingleDrone produces one deadlock in the 50-drone fleet, four deadlocks in the 100-drone fleet, and five deadlocks in the 200-drone fleet. MultiDrone produces one deadlock in the 100- and 200-drone fleets, and SpeedDrone produces two deadlocks in the 200-drone fleet. As deadlocks

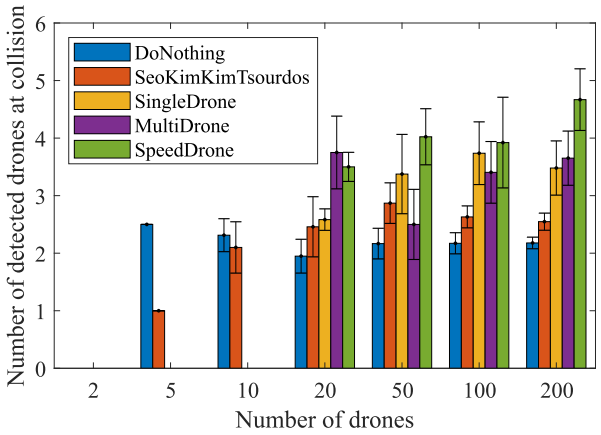



Fig. 10. Number of detected drones at collision.

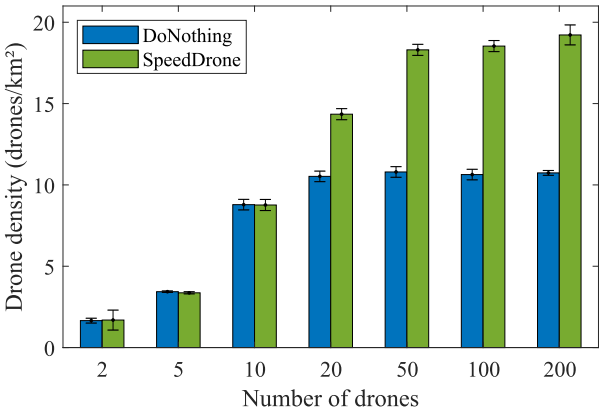



Fig. 11. Drone density for DoNothing and SpeedDrone.

are critical to the safety and battery of drones, SpeedDrone is the best collision avoidance approach to reduce deadlocks in the constant arrival rate scenario.

Fig. 10 shows the number of detected drones at collision. SeoKimKimTsourdos presents between 1.17 and 1.32 times more detected drones than DoNothing for the fleets with 50 or more drones. SingleDrone and MultiDrone are statistically equal and detect between 1.29 and 1.43 times more drones than SeoKimKimTsourdos for the 100- and 200-drone fleets. The collisions in SpeedDrone happen with 4.7 drones in the 200-drone fleet, representing 1.83 more drones than SeoKimKimTsourdos and indicating that SpeedDrone can avoid collisions in the presence of more drones in a more complex scenario.

In the constant drone arrival rate, we do not control the density of drones in the simulation. Fig. 11 shows the density of drones in DoNothing and SpeedDrone. The density of drones in DoNothing gradually increases and reaches 10.7 drones/km$^2$ in the fleets with 20 or more drones. On the other hand, SpeedDrone reaches higher densities in these fleets, between 14.3 and 18.5 drones/km$^2$.

This result shows the difference in the density of drones for the same fleet size in different collision avoidance approaches. The baseline is 10.7 drones/km$^2$ in DoNothing, indicating that including more drones in the fleets with 20 drones or more does not increase the load in the simulation. Collision avoidance approaches, such as SpeedDrone, increase the density of

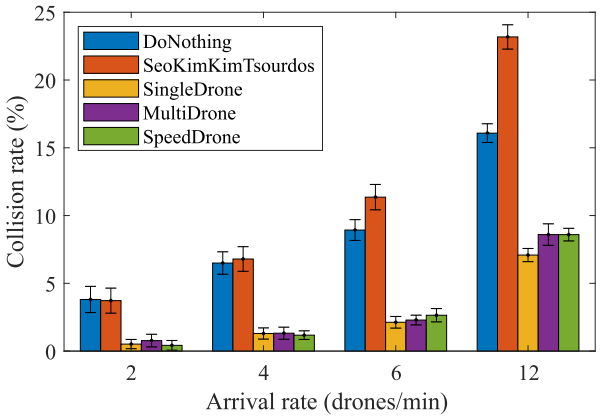



Fig. 12. Collision rate at Poisson arrival rates.

drones, although there is a trend to reach a constant value if the fleets present 50 or more drones. The simulation load reaches a constant value when using 50 or more drones for all approaches. Thus, we must control the load by changing a constant arrival rate to a Poisson drone arrival rate.

### B. Poisson Drone Arrival Rate

Fig. 12 shows the collision rate for four Poisson arrival rates of drones. The collision rate considers the total number of collisions and launched drones, even those that happened outside the stationary time. SeoKimKimTsourdos does not present statistical differences compared with DoNothing for the first two arrival rates, showing that it cannot work well even in low densities. While Seo et al. [11] successfully avoided collisions with one moving obstacle in their paper, our low-density scenario can impose more than one obstacle at a time for a drone to detour. Additionally, it increases the collision rates by 1.27 and 1.44 times compared with DoNothing for 6 and 12 drones/min, respectively. Our approaches significantly reduce the collisions compared with DoNothing at all arrival rates, presenting no statistical differences between them, except between SingleDrone and Multidrone or SpeedDrone for 12 drones/min. They reduce the collision rates between 1.87 and 8.99 times compared with DoNothing, with MultiDrone and SpeedDrone presenting a collision rate of 1.21 times larger than SingleDrone at 12 drones/min. This result shows that our approaches effectively reduce collisions under the more realistic scenario of a Poisson arrival rate of drones.

All approaches significantly increase the collision rates for 12 drones/min compared with 6 drones/min. DoNothing and SeoKimKimTsourdos increase them 1.8 and 2.0 times, respectively, while our approaches increase them from 3.25 to 3.75 times. This result shows that the arrival rate of 12 drones/min represents a critical scenario. Therefore, we separately analyze the details of the collisions that happen in the stationary time for 6 and 12 drones/min.

Fig. 13 shows the number of collisions per collision situation for 6 drones/min. Most collisions occur when both drones are on a cruise flight, in which SeoKimKimTsourdos produces more collisions than DoNothing, although these approaches do not present statistical differences. Our approaches significantly

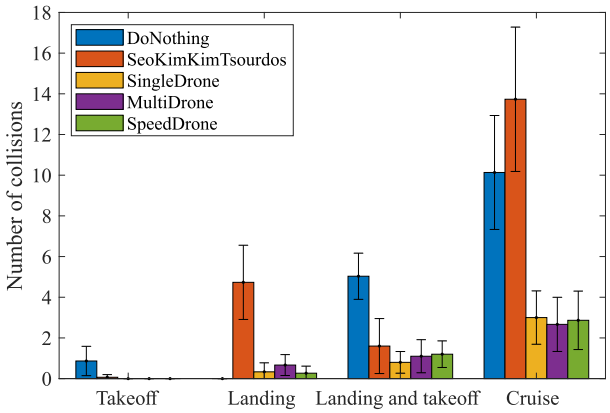



Fig. 13. Number of colliding drones per situation for 6 drones/min.

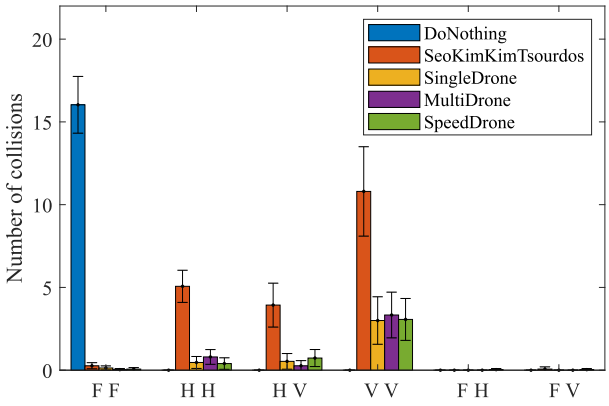



Fig. 14. Number of colliding drones per detour state for 6 drones/min.

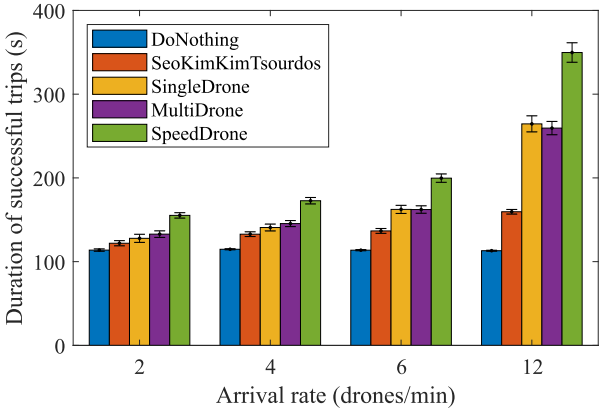



Fig. 15. Duration of successful trips at Poisson arrival rates.

reduce cruise collisions, with 4 to 4.94 times fewer collisions than DoNothing, confirming their effectiveness in reducing cruising collisions.

Most collisions when one drone is landing and another is taking off occur in DoNothing. In this situation, the detour approaches significantly reduce the collisions, between 3.70 and 4.72 times, with no significant statistical differences between ours and SeoKimKimTsourdos. SeoKimKimTsourdos generates 4.7 collisions when one drone lands and the other is on a cruise flight, while DoNothing generates none. Our approaches do not present statistical differences compared with DoNothing in this situation, although they present between 0.27 and 0.67 collisions. Finally, DoNothing presents the most collisions when one drone takes off and the other is on a cruise flight. SeoKimKimTsourdos and DoNothing do not have statistical differences in this situation, while our approaches eliminate the collisions. These results unveil some positive side effects of our approaches, which can reduce or eliminate collisions even in situations not initially considered at algorithm design time.

Fig. 14 shows the detour state of the colliding drones for 6 drones/min with the same labels as the constant arrival rate. Again, all collisions happen when both drones are not detouring in DoNothing (flying state), as expected. The detour approaches virtually do not cause collisions in this state. SeoKimKimTsourdos causes the most collisions in all states with both drones detouring. When both drones detour vertically, our approaches reduce the collisions between 3.24 and 3.6 times compared with SeoKimKimTsourdos. Our approaches also reduce the collisions between 6.33 and 12.67 times compared with SeoKimKimTsourdos when both drones detour horizontally. When one drone detours horizontally and another vertically, our approaches reduce the collisions between 5.36 and 14.75 times compared with SeoKimKimTsourdos. Finally, all approaches practically present no collisions when one drone is not detouring. Our approaches present no statistical differences in any case. This chart is similar to the constant arrival rate, with the same conclusions.

Fig. 15 shows the duration of successful trips for all Poisson arrival rates. DoNothing presents an average of 113.85 s, similar to the constant arrival rate, and SeoKimKimTsourdos increases it between 1.07 and 1.41 times. SingleDrone and MultiDrone take longer than SeoKimKimTsourdos, virtually with no statistical differences, and increase the time of successful trips between 1.12 and 2.34 times compared with DoNothing. SpeedDrone causes the most extended trip duration, between 1.36 and 3.09 times longer than DoNothing.

The execution time is the same for all approaches since they wait for 1000 s to enter the stationary time, and the simulation continues for 1800 s. Only SpeedDrone presents $0.03 \pm 0.07$ drones in deadlocks for 12 drones/min, showing no statistical differences between the other approaches. Thus, we consider that there are no deadlocks in the Poisson arrival rates.

Fig. 16 shows the number of detected drones at collision. SpeedDrone presents the most significant number of detected drones, with 12.79 drones in 12 drones/min, while SingleDrone and MultiDrone are statistically equal at this rate, with 9.03 drones. There are between 1.89 and 2.68 times more drones detected at a collision in our approaches than SeoKimKimTsourdos, which presents 4.77 drones and 1.38 times more drones than DoNothing. Our approaches are statistically equal in 6 drones/min and detect more drones than SeoKimKimTsourdos and DoNothing, which are also statistically equal. SpeedDrone detects 1.37 more drones than DoNothing in 4 drones/min, while the other approaches are statistically equal. MultiDrone, DoNothing, and SeoKimKimTsourdos are statistically equal

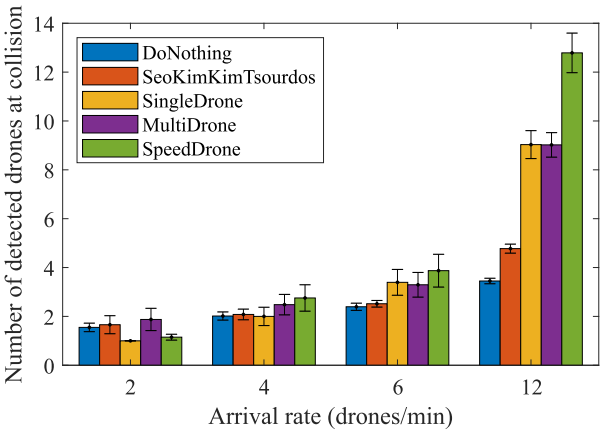



Fig. 16. Number of detected drones at collision at Poisson arrival rates.

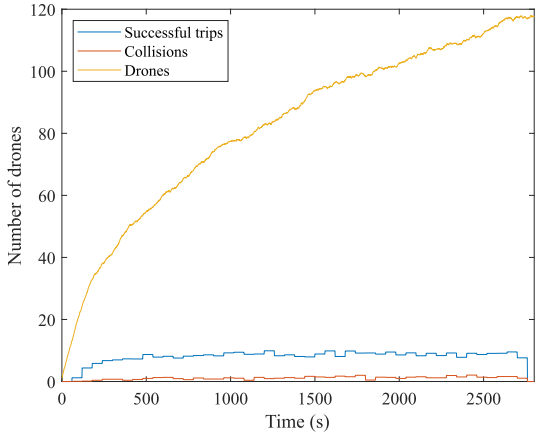



Fig. 17. Number of drones, successful trips, and collisions over time for 12 drones/min and SpeedDrone.

for 2 drones/min, detecting more drones than SingleDrone and SpeedDrone.

As the number of detected drones at collision indicates the complexity of the collisions and the scenario, we analyze the number of drones, successful trips, and collisions over time in SpeedDrone for 12 drones/min in Fig. 17. SpeedDrone maintains an approximately constant low value of collisions – 0.53 collisions/min on average – and an approximately constant value of successful trips – 3.83 trips/min. As the drones do not collide but also do not finish their trips, the density of drones increases, reaching almost 120 drones at the end of the simulation. SingleDrone and MultiDrone produce a similar chart, but the number of drones is smaller, reaching 76.43 and 65 drones at the end of the simulation, respectively. The simulations suggest that SeoKimKimTsourdos and DoNothing achieve a constant number of drones, *i.e.*, the stationary time, like all approaches in the lower arrival rates. This result confirms that this arrival rate produces a critical scenario for our approaches, in which the drones do not collide as designed but still need to finish their trips.

### C. The Impact of Several Altitudes in 3D Scenarios

Unlike the current literature, we consider a true 3D scenario, and thus, this section presents results for drones flying at multiple altitudes. Also, we compare the VSR strategy, as it always

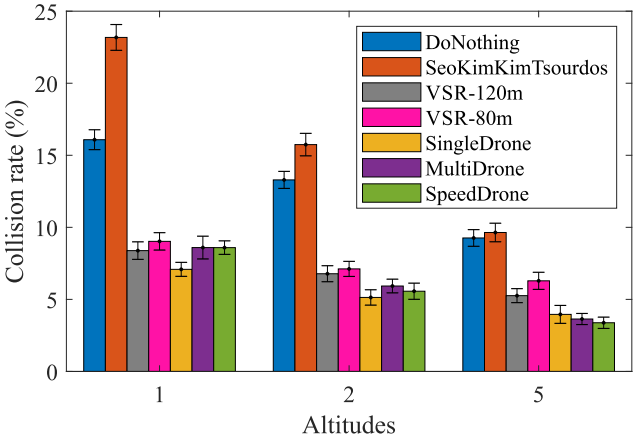



Fig. 18. Collision rate using several altitudes for 12 drones/min.

makes vertical detours and does not limit the altitudes. Fig. 18 depicts the collision rate when the drones fly at the same altitude, like the previous subsections, and randomly choose one of two and one of five different altitudes for the cruise flight and the highest Poisson arrival rate of 12 drones/min. As expected, as we include more altitudes, the collision rate drops for all approaches, between 1.21 and 1.54 times fewer collisions for the simulations with two altitudes and 1.44 and 2.54 times fewer collisions for five altitudes than those with only one altitude. The exception is VSR-80m, which does not present statistical differences between two and five altitudes. Nonetheless, our approaches still significantly reduce the collision rate compared to DoNothing and SeoKimKimTsourdos and do not present statistical differences, except for one altitude already discussed in the previous subsection. DoNothing and SeoKimKimTsourdos are statistically equal for five altitudes, like the lower Poisson rates in the previous subsection. Compared with VSR, our approaches significantly reduce the collision rate for five altitudes, and SingleDrone significantly reduces it in all cases. SpeedDrone also significantly reduces the collision rate for two altitudes, and MultiDrone reduces it for two altitudes compared with VSR-80m.

Fig. 19 shows the achieved altitudes. SeoKimKimTsourdos and our approaches increase the mean altitudes between 1.13 and 1.51 times compared with DoNothing and are statically equal, except for SeoKimKimTsourdos and SpeedDrone for five altitudes. On the other hand, VSR increases the altitudes even more, between 1.27 and 2.36 times compared with DoNothing. This result confirms the VSR strategy of avoiding collisions by flying to higher altitudes, thus escaping to areas with lower drone density. On the other hand, this strategy consumes more energy. Additionally, as the collision rate of VSR presents the worst improvements when using more altitudes, between 1.24 and 1.59 times, VSR may not be adequate when using different altitudes for the cruise flight.

Although we can reduce collisions using different altitudes for cruise flights, we cannot eliminate them. The decrease in the collisions is due to the lower density at each altitude. However, this situation is only valid while drones are at the cruise flight stage, so the improvements are nonlinear. The collisions during takeoff, landing, and vertical detouring still occur. Thus, even though any real-world solution must rely

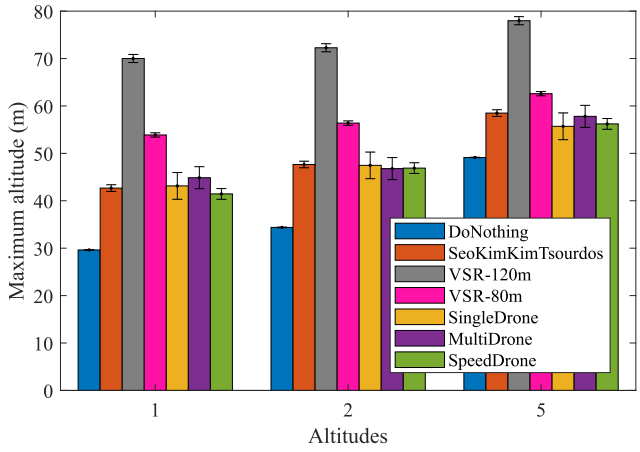



Fig. 19. Achieved altitudes using several altitudes for 12 drones/min.

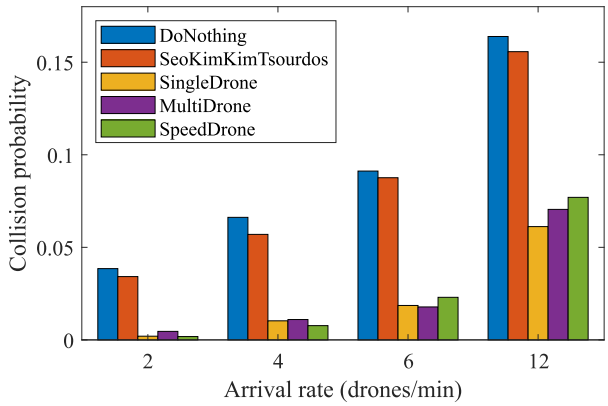



Fig. 20. Collision (S13) probability for the stationary states.

on multiple altitudes to allow higher-density services, different collision-avoidance strategies will still be needed.

## VII. Analytical Results

As simulations take a long time, partly because of the number of replications, we need help to understand what occurs with the drones after the experiment finishes. Fig. 17 provides insights into the simulation behavior, given that the number of drones increases, whereas the number of successful trips and collisions reaches a steady state. To improve our understanding of the asymptotic behavior of the strategies, we model the drone delivery scenario as a Markov chain, as described in Subsection V-B.

Fig. 20 depicts the collision probabilities, *i.e.*, the stationary state for S13, for the five algorithms and four Poisson arrival rates of Subsection VI-B. As expected, the collision probability increases with higher arrival rates since the drone density is higher. Also, our algorithms (SingleDrone, MultiDrone, and SpeedDrone) have lower collision probabilities than DoNothing and SeoKimKimTsourdos. One can observe that the results resemble the collision rates depicted in Fig. 12, with some significant differences. The key differences are because the UTSim simulations generate collision rates, *i.e.*, the number of colliding drones by the total number of drone trips. In contrast, here, we compute the collision probability of the Markov chain in the stationary state. UTSim keeps track of all collisions when computing the collision rates, whereas the results from Fig. 20 are general state probabilities to represent all drones at once. Thus, the collision probabilities differ from the collision rates since all the drones that keep flying at the end of the UTSim simulations manage to land in the stationary state. Specifically, when comparing DoNothing to SeoKimKimTsourdos, we can see that the latter generates higher collision rates than the former because DoNothing does not detour and has very few flying drones at the end of the simulations, matching approximately the collision probability values. On the other hand, the collision probabilities for SeoKimKimTsourdos are lower than DoNothing because all the flying drones arrived at the stationary state.

Please notice that, as an analytical tool, the Markov Chain does not account for practical issues. For example, drones can fly for an extended period unaffected by the limited battery capacity. Even though this is a purely theoretical consideration, these results help us to understand the behavior of drone delivery services, as shown in the following results.

TABLE II
Collision Probability Breakdown

| Algo | Rate (d/m) | Flight Phase | | | |
|---|---|---|---|---|---|
| | | Takeoff (S1+S7) | Cruise (S2+S8) | Detour (S3+S9) | Landing (S5+S11) |
| Do Nothing | 2 | 0.006 | 0.027 | 0.000 | 0.006 |
| | 4 | 0.012 | 0.044 | 0.000 | 0.010 |
| | 6 | 0.017 | 0.060 | 0.000 | 0.015 |
| | 12 | 0.030 | 0.109 | 0.000 | 0.025 |
| Seo KimKim Tsourdos | 2 | 0.000 | 0.003 | 0.027 | 0.004 |
| | 4 | 0.000 | 0.007 | 0.044 | 0.007 |
| | 6 | 0.001 | 0.010 | 0.067 | 0.010 |
| | 12 | 0.001 | 0.017 | 0.122 | 0.016 |
| Single Drone | 2 | 0.000 | 0.001 | 0.000 | 0.001 |
| | 4 | 0.001 | 0.002 | 0.007 | 0.001 |
| | 6 | 0.000 | 0.004 | 0.014 | 0.001 |
| | 12 | 0.001 | 0.009 | 0.049 | 0.002 |
| Multi Drone | 2 | 0.000 | 0.000 | 0.003 | 0.001 |
| | 4 | 0.001 | 0.001 | 0.009 | 0.000 |
| | 6 | 0.001 | 0.003 | 0.013 | 0.001 |
| | 12 | 0.001 | 0.009 | 0.058 | 0.002 |
| Speed Drone | 2 | 0.000 | 0.000 | 0.001 | 0.000 |
| | 4 | 0.000 | 0.002 | 0.005 | 0.001 |
| | 6 | 0.001 | 0.003 | 0.018 | 0.001 |
| | 12 | 0.001 | 0.009 | 0.064 | 0.003 |

Table II breaks down the results of Fig. 20, showing the collision probability for four flight phases (Takeoff, Cruise, Detour, and Landing). We add the results for the phases with equivalent states in the delivery and return trips for improved visibility. For example, the results for Cruise are the sum of S2 (cruising on the delivery trip) and S8 (cruising on the return trip). DoNothing has a higher probability of collisions during cruising, whereas for the other algorithms, this happens when detouring.

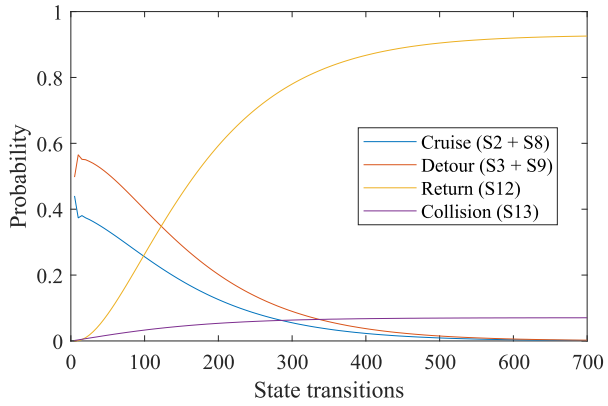



Fig. 21. Probability of flight phases for MultiDrone with 12 drones/min with increasing state transitions.

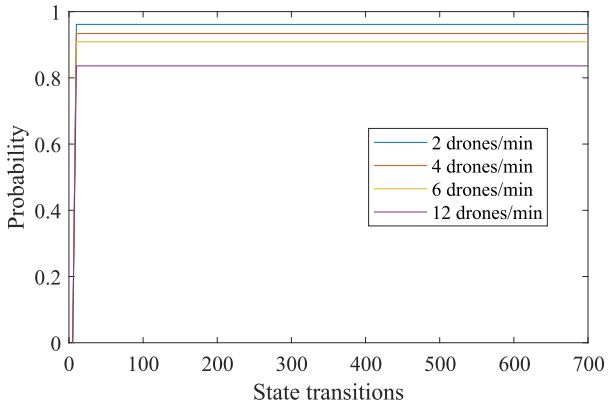



Fig. 22. Probability of S12 (Return to Distribution Center) for DoNothing for 2, 4, 6, and 12 drones/min with increasing state transitions.

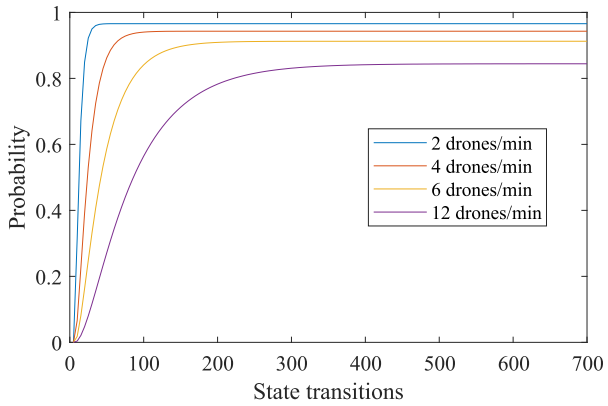



Fig. 23. Probability of S12 (Return to Distribution Center) for SeoKimKimTsourdos for 2, 4, 6, and 12 drones/min with increasing state transitions.

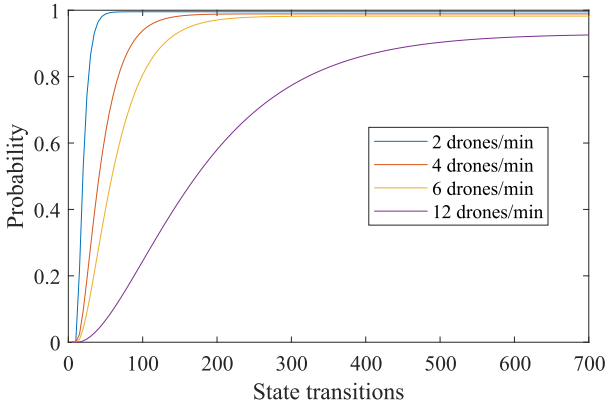



Fig. 24. Probability of S12 (Return to Distribution Center) for MultiDrone for 2, 4, 6, and 12 drones/min with increasing state transitions.

Next, we dive into the evolution of state probabilities as we increase the number of transitions. Fig. 21 shows the final state probabilities for MultiDrone with 12 drones/min, varying the number of state transitions from 5 to 700. In the beginning (5 transitions), most drones are detouring (probability 0.50), some are cruising (probability 0.44), no one successfully returns to the distribution center (probability 0.00), and a small fraction collides (probability 0.02). Probabilities for takeoff and landing account for the remaining 0.04 but are not shown in the chart for improved visibility. As we keep increasing the number of transitions, the probability of successful trips (S12) increases at the same rate that the probabilities of detour and cruise decrease, revealing that, at some point, many detouring drones complete the trip. Also, another fraction of drones collide, as the probability of state S13 reaches 0.07. These results reveal the asymptotic behavior for the simulations (which cannot run indefinitely), given that the probabilities fed by the UTSim simulations do not change.

We also compare the probabilities of successful trips (*i.e.*, state S12) for different arrival rates and our MultiDrone algorithm, DoNothing, and SeoKimKimTsourdos. Fig. 22 shows that DoNothing reaches the final state before 10 transitions, as from S0 to S12, there are only eight transitions without detouring. On the other hand, Fig. 23 shows for SeoKimKimTsourdos that, as the drone density increases (with increased arrival rates), the drones take more time to reach the final state, *i.e.*, keep more time cruising and detouring. These results help explain why the collision rate depicted in Fig. 12 is higher for SeoKimKimTsourdos than DoNothing, but the stationary probability is lower. Finally, Fig. 24 reveals that MultiDrone decreases the collisions by keeping more drones detouring and cruising compared with SeoKimKimTsourdos, as it takes more transitions to reach the stationary probability. However, more drones achieve S12.

## VIII. Discussion

Our experience yields important discussions coming from the lessons we learned.

### A. Drone Density in the Airspace

Excessive delays in evaluating flight risks may lead to catastrophic events when not effectively accommodated, as rigorous collision avoidance is a time-critical mission [21]. Geometric approaches have low computation overhead and cost but low accuracy when the number of obstacles is high with a large turning radius [19]. The increased airspace usage can induce a secondary threat due to the conflict resolution, which may issue an improper maneuver that resolves a one-on-one encounter with a first threat [21], [42]. The major challenge is ensuring that local decisions also satisfy the coupling collision avoidance constraints [43] of multiple drones.

The simulation results reveal that the 12 drones/min arrival rate generates a density limit to our approaches since they do

not achieve a stationary time. Although SeoKimKimTsourdos does achieve the stationary time at this arrival rate, its effectiveness is low due to the increased collisions compared with DoNothing. The increased duration of successful trips generated by our approaches indicates that they make many detours along the way. While it becomes clear that our approaches effectively reduce collisions, the increased number of detours may hinder the success of conflict resolution with multiple drones. With an arrival rate of 12 drones/min, the results indicate the efficiency of our approaches in avoiding collisions that can negatively affect aerial delivery services. Additionally, the increase in the altitude levels that drones can fly in cruise flights reduces the density per level, where most collisions occur. While we reduce the collisions with this strategy, we do not eliminate them, suggesting we need more sophisticated approaches.

### B. Communication and Cooperation

Further techniques must be developed or tested against our baseline and geometric approaches necessary to reduce collisions, including communication, coordination, and intelligence. Drones can communicate with each other, exchanging data on drones in the neighborhood to avoid collisions. They can also communicate with base stations, sharing their data and collecting data on a larger region. An intelligent component should either be deployed onboard the drone or at a multiaccess-edge computing (MEC) node that can read the gathered data from different drones, process them, and then make the right decision to detect and avoid physical collision [13]. We can also integrate these techniques into a collision management strategy based on communication.

In the literature, complex solutions, *e.g.*, deep learning [13], are not compared with simpler ones, so there are no simpler baselines to compare with. The typical collision rate is in the 10%-40% range, which is too high and does not justify not including a simpler baseline, *e.g.*, a geometric approach, in the comparison. This paper compares the simplest baseline, not detouring at all, to simple solutions, Random and Rightward, and medium-complexity solutions, as geometric approaches, showing that they present collision rates between 0% and 25.03%. We show the effectiveness of our approaches in the collision rates between 0% and 2.64%. As a next step, we can propose more complex solutions involving intelligence, such as deep learning, to eliminate collisions in aerial delivery services.

### C. Scalability

The numerical results obtained from the Markov chain are valuable to the simulation results, as they extend the limited simulation time to capture the asymptotic behavior in the stationary state. However, it is essential to note that the probabilities for the transition matrices are extracted from the simulations, representing a much shorter period of the drone delivery service. Therefore, the analytical results assume that the transition matrix, *i.e.*, all state transitions, remains the same after the simulation ends. It is worth emphasizing that analytical modeling involves abstractions and simplifications since real-world data from drone delivery services is often unavailable, and longer simulations are not always feasible due to their unpredictable length. Nonetheless, these analytical results remain valuable and informative for understanding the behavior of the drone delivery service under study.

### D. Limitations

We analyze our results in light of non-cooperative collision avoidance techniques, which highlight a limitation of our approaches. Namely, they do not make up a complete solution by themselves. Rather, deploying a high-density real-world drone delivery service will likely require a combination of different cooperative and non-cooperative strategies to be used by drones at the most appropriate time as a toolset to provide a collision-free delivery service.

As a scenario with collisions is safety-critical, we need to ensure that no collisions will happen in the experiments since they could cause injuries and a considerable waste of money. Therefore, real-world experiments are impossible before a collision-free system is in place. However, a first step toward real-world experiments may be setting up an indoor prototype with miniatures as soon as there is evidence of the effectiveness of a solution to provide safe deliveries.

Our geometric approaches are deterministic in that the decisions are always the same for a given situation. In preliminary studies, we identified that this characteristic may lead to some unexpected biases regarding the direction the drones take and the coordinates where the collisions occur. However, this issue requires more in-depth investigation to be completely understood.

## IX. Conclusion

This paper proposes and evaluates an aerial delivery service scenario and three geometric approaches for collision avoidance: SingleDrone, MultiDrone, and SpeedDrone. Our performance analysis employs simulation and analytical modeling. We compare the proposed approaches to not detouring, simple collision avoidance strategies, and two geometric algorithms from the literature.

We simulate the proposed scenario with seven fleets on a constant drone arrival rate and four Poisson arrival rates, showing the number of collisions and collision details. The results show that our approaches significantly reduce the collisions compared with all other approaches, especially the cruise collisions, which are more challenging to avoid. The collisions generated by our approaches present more drones within the colliding drone detection radius, indicating that these collisions are more complex to avoid than the ones of the other strategies. Simple approaches and the algorithm proposed by Seo et al. [11] increase the collisions under the simulated scenario compared with the DoNothing baseline, showing the ineffectiveness of the approaches not designed for a high-density scenario.

We also assess the limitations of the proposed approaches, which mostly collide vertically and make more detours, generating longer trips. We evaluate the number of collisions,

successful trips, and drones over time for SpeedDrone in the 12 drones/min arrival rate. The number of collisions and successful trips is constant and low while the number of drones increases. This result indicates that our approaches effectively reduce collisions, but drones do not finish their trips, which increases the density of drones. The detours in our approaches are effective for lower arrival rates, but they compromise the increase in successful trips with higher arrival rates. The analytical modeling using Markov Chains corroborates the simulation results by shedding some light on and helping to explain the simulation results. Finally, we simulate drones randomly choosing one of two and five altitudes for the cruise flight. These simulations reveal that more altitudes lead to lower densities and collision rates, although more altitudes do not eliminate collisions. Approaches that always solve conflicts vertically, such as VSR [18], may not be adequate in this situation, while our GeoDrone approaches significantly reduce collisions when the drones use more altitudes.

For future work, we may employ communication, coordination, and intelligence to reduce further or eliminate collisions. The solutions include communication between drones and edge infrastructure, machine learning techniques, and integrating these solutions to achieve a management system based on communication and intelligence. We can limit the arrival rate of drones when the density of drones reaches a threshold or create precise routes with airways. Machine learning techniques can be applied to define optimal arrival rates according to traffic conditions or calculate each drone collision risk, taking the proper action for each risk range.

## REFERENCES


[1] K.-W. Chen, M.-R. Xie, Y.-M. Chen, T.-T. Chu, and Y.-B. Lin, "DroneTalk: An Internet-of-Things-based drone system for last-mile drone delivery," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 15204–15217, Sep. 2022.

[2] F. Kong, J. Li, B. Jiang, H. Wang, and H. Song, "Trajectory optimization for drone logistics delivery via attention-based pointer network," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 4, pp. 4519–4531, Apr. 2023.

[3] P. Sánchez, R. Casado, and A. Bermúdez, "Real-time collision-free navigation of multiple UAVs based on bounding boxes," *Electronics*, vol. 9, no. 10, pp. 16–32, 2020.

[4] F. B. Sorbelli, F. Corò, L. Palazzetti, C. M. Pinotti, and G. Rigoni, "How the wind can be leveraged for saving energy in a truck-drone delivery system," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 4, pp. 4038–4049, Apr. 2023.

[5] A. Jeon, J. Kang, B. Choi, N. Kim, J. Eun, and T. Cheong, "Unmanned aerial vehicle last-mile delivery considering backhauls," *IEEE Access*, vol. 9, pp. 85017–85033, 2021.

[6] G. Brunner, B. Szebedy, S. Tanner, and R. Wattenhofer, "The urban last mile problem: Autonomous drone delivery to your balcony," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2019, pp. 1005–1012.

[7] A. Al-Mousa, B. H. Sababha, N. Al-Madi, A. Barghouthi, and R. Younisse, "UTSim: A framework and simulator for UAV air traffic integration, control, and communication," *Int. J. Adv. Robot. Syst.*, vol. 16, no. 5, pp. 1–19, 2019.

[8] B. H. Sababha, A. Al-Mousa, R. Baniyounisse, and J. Bdour, "Sampling-based unmanned aerial vehicle air traffic integration, path planning, and collision avoidance," *Int. J. Adv. Robot. Syst.*, vol. 19, no. 2, p. 17298806221086431, 2022.

[9] L. Lu, G. Fasano, A. Carrio, M. Lei, H. Bavle, and P. Campoy, "A comprehensive survey on non-cooperative collision avoidance for micro aerial vehicles: Sensing and obstacle detection," *J. Field Robot.*, vol. 40, no. 6, pp. 697–1720, May 2023.

[10] J. N. Yasin, S. A. S. Mohamed, M.-H. Haghbayan, J. Heikkonen, H. Tenhunen, and J. Plosila, "Unmanned aerial vehicles (UAVs): Collision avoidance systems and approaches," *IEEE Access*, vol. 8, pp. 105139–105155, 2020.

[11] J. Seo, Y. Kim, S. Kim, and A. Tsourdos, "Collision avoidance strategies for unmanned aerial vehicles in formation flight," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 53, no. 6, pp. 2718–2734, Dec. 2017.

[12] W. Chen, X. Qiu, T. Cai, H.-N. Dai, Z. Zheng, and Y. Zhang, "Deep reinforcement learning for Internet of Things: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 1659–1692, 3rd Quart., 2021.

[13] S. Ouahouah, M. Bagaa, J. Prados-Garzon, and T. Taleb, "Deep-reinforcement-learning-based collision avoidance in UAV environment," *IEEE Internet Things J.*, vol. 9, no. 6, pp. 4015–4030, Mar. 2022.

[14] D. Wang, T. Fan, T. Han, and J. Pan, "A two-stage reinforcement learning approach for multi-UAV collision avoidance under imperfect sensing," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3098–3105, Apr. 2020.

[15] N. Thumiger and M. Deghat, "A multi-agent deep reinforcement learning approach for practical decentralized UAV collision avoidance," *IEEE Control Syst. Lett.*, vol. 6, pp. 2174–2179, 2022.

[16] D. Choi, K. Lee, and D. Kim, "Enhanced potential field-based collision avoidance for unmanned aerial vehicles in a dynamic environment," in *Proc. AIAA Scitech Forum*, Jan. 2020, pp 1–7.

[17] X. Guan, R. Lyu, H. Shi, and J. Chen, "A survey of safety separation management and collision avoidance approaches of civil UAS operating in integration national airspace system," *Chin. J. Aeronaut.*, vol. 33, no. 11, pp. 2851–2863, 2020.

[18] J.-W. Park, H.-D. Oh, and M.-J. Tahk, "UAV collision avoidance based on geometric approach," in *Proc. SICE Annu. Conf.*, 2008, pp. 2122–2126.

[19] Z. Wei, Z. Meng, M. Lai, H. Wu, J. Han, and Z. Feng, "Anti-collision technologies for unmanned aerial vehicles: Recent advances and future trends," *IEEE Internet Things J.*, vol. 9, no. 10, pp. 7619–7638, May 2022.

[20] K. Dushime, L. Nkenyereye, S. K. Yoo, and J. Song, "A review on collision avoidance systems for unmanned aerial vehicles," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2021, pp. 1150–1155.

[21] J. Tang, S. Lao, and Y. Wan, "Systematic review of collision-avoidance approaches for unmanned aerial vehicles," *IEEE Syst. J.*, vol. 16, no. 3, pp. 4356–4367, Sep. 2022.

[22] J. R. Kellner et al., "New opportunities for forest remote sensing through ultra-high-density drone LiDAR," *Surv. Geophys.*, vol. 40, no. 4, pp. 959–977, 2019.

[23] T. S. Taylor, *Introduction to Laser Science and Engineering*. Boca Raton, FL, USA: CRC Press, 2019.

[24] R. Ourari, K. Cui, A. Elshamanhory, and H. Koeppl, "Nearest-neighbor-based collision avoidance for quadrotors via reinforcement learning," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 293–300.

[25] S.-J. Chung, A. A. Paranjape, P. Dames, S. Shen, and V. Kumar, "A survey on aerial swarm robotics," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 837–855, Aug. 2018.

[26] A. Tahir, J. Böling, M.-H. Haghbayan, H. T. Toivonen, and J. Plosila, "Swarms of unmanned aerial vehicles—A survey," *J. Ind. Inf. Integr.*, vol. 16, Jan. 2019, Art. no. 100106.

[27] I. Mahjri, A. Dhraief, and A. Belghith, "A review on collision avoidance systems for unmanned aerial vehicles," in *Proc. Int. Workshop Commun. Technol. Vehicles*. Cham, Switzerland: Springer, 2015, pp. 203–214.

[28] K. Shen, R. Shivgan, J. Medina, Z. Dong, and R. Rojas-Cessa, "Multidepot drone path planning with collision avoidance," *IEEE Internet Things J.*, vol. 9, no. 17, pp. 16297–16307, Sep. 2022.

[29] M. Abdelkader, S. Güler, H. Jaleel, and J. S. Shamma, "Aerial swarms: Recent applications and challenges," *Curr. Robot. Rep.*, vol. 2, no. 3, pp. 309–320, 2021.

[30] S. Aggarwal and N. Kumar, "Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges," *Comput. Commun.*, vol. 149, pp. 270–299, Jan. 2020.

[31] N. Gageik, P. Benz, and S. Montenegro, "Obstacle detection and collision avoidance for a UAV with complementary low-cost sensors," *IEEE Access*, vol. 3, pp. 599–609, 2015.

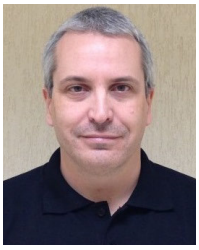
[32] J. Sun, J. Tang, and S. Lao, "Collision avoidance for cooperative UAVs with optimized artificial potential field algorithm," *IEEE Access*, vol. 5, pp. 18382–18390, 2017.

[33] E. Frachtenberg, "Practical drone delivery," *Computer*, vol. 52, no. 12, pp. 53–57, Dec. 2019.

[34] M. Ayamga, S. Akaba, and A. A. Nyaaba, "Multifaceted applicability of drones: A review," *Technol. Forecast. Soc. Change*, vol. 167, Jun. 2021, Art. no. 120677.
[35] A. Kasliwal et al., "Role of flying cars in sustainable mobility," *Nature Commun.*, vol. 10, no. 1, pp. 1–9, 2019.
[36] X. Yang and P. Wei, "Autonomous free flight operations in urban air mobility with computational guidance and collision avoidance," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 9, pp. 5962–5975, Sep. 2021.
[37] I. C. Kleinbekman, M. A. Mitici, and P. Wei, "EVTOL arrival sequencing and scheduling for on-demand urban air mobility," in *Proc. IEEE/AIAA 37th Digit. Avionics Syst. Conf. (DASC)*, Sep. 2018, pp. 1–7.
[38] DJI. (2023). *Consumer Drones Comparison*. Accessed: Apr. 30, 2023. [Online]. Available: https://www.dji.com/products/compare-consumer-drones
[39] L. P. Soares, F. M. C. de Oliveira, C. A. Kamienski, and L. F. Bittencourt, "Drone edge management system (DREMS): Sequencing drone takeoff and landing," in *Proc. 10th Int. Conf. Future Internet Things Cloud*, Aug. 2023.
[40] Federal Aviation Administration. (2020). *Small Unmanned Aircraft Systems (UAS) Regulations (Part 107)*. Accessed: Sep. 13, 2023. [Online]. Available: https://www.faa.gov/newsroom/small-unmanned-aircraft-systems-uas-regulations-part-107
[41] D. A. Levin and Y. Peres, *Markov Chains and Mixing Times*, vol. 107. Providence, RI, USA: Amer. Math. Soc., 2017.
[42] Z. M. Fadlullah, D. Takaishi, H. Nishiyama, N. Kato, and R. Miura, "A dynamic trajectory control algorithm for improving the communication throughput and delay in UAV-aided networks," *IEEE Netw.*, vol. 30, no. 1, pp. 100–105, Jan. 2016.
[43] B. Sabetghadam, R. Cunha, and A. Pascoal, "A distributed algorithm for real-time multi-drone collision-free trajectory replanning," *Sensors*, vol. 22, no. 5, pp. 1–12, 2022.

**Fabíola M. C. de Oliveira** received the B.Sc. degree in control and automation engineering, the M.Sc. degree in mechanical engineering, and the Ph.D. degree in computer science from the University of Campinas, Campinas, Brazil, in 2012, 2015, and 2020, respectively. Currently, she is a Post-Doctoral Researcher with the Federal University of ABC, Santo André, Brazil. Her research interests include federated learning, intelligent transportation systems, graph partitioning, distributed deep learning, fog computing, the Internet of Things, distributed computing, parallel computing, and high-performance computing.

**Luiz F. Bittencourt** (Senior Member, IEEE) received the degree in computer science from the Federal University of Parana and the M.Sc. and Ph.D. degrees from the University of Campinas (Unicamp). He is currently an Associate Professor with Unicamp. He has been a Visiting Researcher with The University of Manchester, Cardiff University, and Rutgers University, USA. He was the TPC Co-Chair of IEEE/ACM UCC2018 and the Track Chair of CloudCom (2018–2019) and FiCloud (2017–2023). He serves as an Associate Editor for the *Computers and Electrical Engineering*, *Internet of Things Journal* (Elsevier), *Journal of Network and Systems Management*, and IEEE NETWORKING LETTERS. He was awarded with the IEEE Communications Society Latin America Young Professional Award in 2013.



**Reinaldo A. C. Bianchi** (Member, IEEE) received the B.Sc. degree in physics and the master's and Ph.D. degrees in engineering from the University of São Paulo. He is currently a Full Professor with the FEI University Center, São Bernardo do Campo. He is a reviewer of several conferences and scientific journals and has organized several conferences. As the main objective of his research work is the creation of autonomous and intelligent systems, he works mainly in the following areas: intelligent robotics, artificial intelligence, machine learning, computer vision, robotics, artificial intelligence applied in finance, and also in the teaching and dissemination of engineering.

**Carlos A. Kamienski** (Senior Member, IEEE) received the B.S. degree in computer science from the Federal University of Santa Catarina, Florianópolis, Brazil, in 1989, the M.S. degree from the State University of Campinas, Campinas, Brazil, in 1994, and the Ph.D. degree in computer science from the Federal University of Pernambuco, Recife, Brazil, in 2003. He is currently a Full Professor in computer science with the Federal University of ABC (UFABC), Santo André, São Paulo, Brazil. His current research interests include the Internet of Things, smart agriculture, smart cities, network softwarization, and online social networks.