**Accepted for Publication in the International Conference on Robotics and Automation (ICRA), 2024.**

# Wind Field Modeling for Formation Planning in Multi-Drone Systems

Minhyuk Park[1] and Tsz-Chiu Au[2]

***Abstract*— In multi-drone systems such as drone light shows, drones move in formation while avoiding collisions. However, few existing formation planning algorithms consider the wind fields of drones during planning. Since the wind field effect is prominent when drones have to fly close to each other, we cannot ignore the effect during planning. In this paper, we extend the reservation system in autonomous intersection management for grid-based formation planning by including a new type of reservation called non-exclusive reservations specifically for handling wind fields. We trained a deep learning model to predict the deviation of a drone's trajectory when the drone enters the wind field of another drone, and then use the reservation grid to prevent collision. Based on the reservation system, we develop a new formation planning algorithm that focuses on adjusting the start times of motion plans to avoid collision. Our experimental results show that trajectory prediction can help make better decisions in task assignment in terms of minimizing the makespan.**

## I. INTRODUCTION

In multirobot systems, robots often move in formation while avoiding collisions. A good example is drone light shows, in which a group of drones, each equipped with a light source, forms a sequence of light patterns in a night sky (see Fig. 1). The performance of a drone light show depends on how quickly the drones can change from one formation to another. There are many works on controlling a team of robots to form and maintain a formation while the robots move together (e.g., [24]). We consider the *formation planning problem* in which a planner generates formation motion plans for drones. The formation plan ensures all drones can reach their destinations while the makespan (i.e., the longest execution time of the motion plans) is minimized.

A unique challenge in multi-drone systems is that drones can generate strong wind below and around the drones, affecting other drones. Although a drone would not crash immediately when it enters another drone's wind field, it can be pushed away, causing the drone to deviate from its intended trajectory. Since the region affected by a drone's wind field can be quite large, a multi-drone planning system cannot ignore the effect of wind fields during planning, especially when drones have to fly close to each other. This is essentially true when we want to achieve a high pixel density in drone light shows.

In this paper, we propose utilizing a *reservation grid* to handle wind fields during formation planning. The idea is similar to the reservation system in autonomous intersection management (AIM), which is used to coordinate autonomous

Fig. 1: A drone formation in a drone light show.[3]

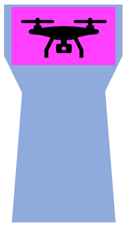
Fig. 2: Cross sections of the safety buffer (the magenta region) and the wind buffer (the light blue region).

vehicles at intersections to achieve an ultra-low traffic delay [4], [8]. In AIM, a vehicle has to reserve the set of space-time tiles on its trajectory in an intersection before it can enter the intersection. Collision avoidance can be achieved by ensuring that no two vehicles reserve the same tile. However, this reservation system is too restrictive for handling the wind fields since it is acceptable for the wind fields of two drones to overlap. Moreover, we should allow drones to enter the wind fields of other drones when it is safe to do so. Since the reservation system in AIM cannot properly handle wind fields, we propose an extension of the reservation system that allows *non-exclusive* reservations for the cells in the wind fields, such that drones can fly closer to each other and more drones can fit into the same space.

The reservation grid will be used to check when a drone enters the wind field of another drone. Instead of forbidding drones from entering any wind field, we trained a neural network model to predict drones' trajectories under the influence of wind fields. Then, the deviated trajectories are used to make reservations in the reservation grid such that collision can be avoided. This approach allows drones to fly even closer to each other temporarily, and the makespan can be reduced further.

To illustrate the necessity of wind field modeling, we devised a new formation planning algorithm that uses goal swapping as in [24]. We show that the goal swapping rule can avoid making wrong decisions that increase the makespan if it takes the deviated trajectories into account.

In summary, the contributions of this paper are:

- We extend the reservation system for AIM with a new type of reservation called non-exclusive reservations for handling wind buffers during the formation planning of multiple drones.
- We develop a formation planning algorithm that avoids collision by adding time delays to the start times of the predefined motion plans. To the best of our knowledge, we are the first to propose adjusting the start times

[1][2]Department of Computer Science and Engineering, Ulsan National Institute of Science and Technology, South Korea. {sqmemory13,chiu}@unist.ac.kr

[3]https://pixisdrones.com/paris-hilton

of motion plans *only* to avoid collision. We state the sufficient condition under which this adjustment exists.

- We trained a neural network model to predict the deviated trajectories under the influence of the wind fields of other drones. We show that trajectory prediction can improve goal swapping in formation planning.

This paper is organized as follows. After presenting the related work in Sec. II, we define our formation planning problem in Sec. III. Then, we discuss what safety buffers and wind buffers in Sec. IV, and how to utilize them in the reservation grid in Sec. V. After that, we discuss how we use deep learning to model deviated trajectories in Sec. VI. Then, we describe our formation planning algorithm with goal swapping in Sec. VII. Finally, we present our experimental results in Sec. VIII and conclude this paper in Sec. IX.

## II. RELATED WORK

Our multi-drone formation planning problem is a *formation reconfiguration* problem, which aims to find a formation plan that transforms an initial formation configuration into a final configuration [9]. A popular approach for formation reconfiguration is the artificial potential field (APF) method [12], [14], [20]. Zhou et al. used buffered Voronoi cells formed with locations of all peer drones to effectively intercept and box all control commands within the cells [29], an approach used by the Crazyflie platform [18]. Reciprocal n-body collision avoidance selects a safe velocity for an agent from observed peer velocities [11], [22]. Other collision avoidance approaches are chance-constrained collision avoidance for MAVs [30], bayesian policy optimization with model predictive control [3], and particle swarm optimization [26]. However, these approaches have no long-term planning for more complicated formation reconfiguration.

Some systems use decentralized approaches for formation reconfiguration. Alonso-Mora et al. presented a method based on the optimal reciprocal collision avoidance (ORCA), which uses a global cost function minimization to assign target velocity and goal points to drones [2]. Bajaj and Rao presented a divide and conquer approach along with an edge following heuristic to fill the space in to display a shape while preventing oscillatory behavior and gridlock [5]. Meng et al. described gene regulatory networks with virtual DNA encoding target formation that diffuses information to other members of the swarm [15]. Wang and Rubenstein proposed a decentralized multi-agent algorithm called Walk, Stop, Count, and Swap that relies on swaps of goals between peers to resolve gridlocks [24]. They also used a local pairwise goal swap technique among the members of a swarm to reduce a cost function. Decentralized approaches typically use less computational resources and are robust against incomplete modeling of agents' behavior. However, they could be less efficient when compared with centralized approaches.

Some multiagent planning algorithms can overcome the weakness of reactive control and decentralized approaches. Yu and LaValle proposed planning optimal paths for multiple robots using integer linear programming models that optimize for makespan or distance travelled [28]. They also proposed a vertex ordering structure with a scheduling algorithm for distributed distance optimal formation path planning [27]. Nar and Kotecha proposed an optimal waypoint assignment algorithm for drone light show formations.

A feature of the reservation grid is the discretization of space and time. This feature is similar to occupancy grids [21], but our work is closely related to the grid-based reservation system in AIM [4], [8]. Wu et al. proposed a data structure called stacked reservation grid (SRG) for motion planning [25]. Lattice-based motion planners go one step further to discretize the state of robots [7], [6].

## III. THE FORMATION PLANNING PROBLEM

Let $\mathcal{V}$ be a set of $n$ drones. A pose $\rho$ for a drone $\nu \in \mathcal{V}$ is the position and the orientation of $\nu$ in the workspace. In a 3D workspace, $\rho$ is $(x, y, z, \theta_x, \theta_y, \theta_z)$, where $(x, y, z)$ is the coordinate of the center of $\nu$ and $\theta_x$, $\theta_y$, and $\theta_z$ are the pitch, yaw and roll rotations of $\nu$. A *formation* $F$ for $\mathcal{V}$ is $\{\rho_1, \rho_2, \ldots, \rho_n\}$, where $\rho_i$ is a pose for $\nu_i \in \mathcal{V}$, for $1 \leq i \leq n$. A *motion plan* $\pi$ for a drone $\nu$ is a sequence of control commands for controlling $\nu$ to move along a trajectory $\mathsf{Traj}[\pi]$. A *formation plan* is $\Pi = \{\pi_i\}_{1 \leq i \leq n}$, where $\pi_i$ is a motion plan for $\nu_i \in \mathcal{V}$. The execution of the motion plans in $\Pi$ does not have to start at the same time. A *schedule* $\Gamma$ for $\Pi$ is $\{t_i\}_{1 \leq i \leq n}$, where $t_i \geq 0$ is the start time of the execution of $\pi_i$. A *timed formation plan* for $\mathcal{V}$ is a pair $(\Pi, \Gamma)$. Let $|\mathsf{Traj}[\pi]|$ be the length of the trajectory according to $\pi$, which is the time difference between the start time and the end time of the execution of $\pi$. The *makespan* of $(\Pi, \Gamma)$ is $\mathsf{makespan}(\Pi, \Gamma) = \max_{1 \leq i \leq n}\{t_i + |\mathsf{Traj}[\pi_i]|\}$, which is the duration of the execution of $\Pi$ given $\Gamma$.

At any time, a drone $\nu$ has a *safety buffer* and a *wind buffer* (see Fig. 2). A safety buffer is a 3D region around $\nu$ that is used to keep $\nu$ from flying too close to other drones. A wind buffer is a 3D region in the wind field generated by $\nu$ such that the wind speed at any point in the wind buffer is larger than a given threshold. The shapes of these buffers depend on the speed and the heading of a drone. We will discuss the modeling of these buffers in Sec. IV.

Given a timed formation plan $(\Pi, \Gamma)$, let $\partial_i$ and $\xi_i$ be the safety buffers and the wind buffer of $\nu_i \in \mathcal{V}$, respectively, when $\nu_i$ flies along the trajectory according to the motion plan $\pi_i$ starting at time $t_i$. The timespans of the buffers begin at time 0 and end at time $t_{\mathsf{end}} = \mathsf{makespan}(\Pi, \Gamma)$. We say two buffers *overlap* if and only if there exists a time $t$ at which the buffers overlap. We define the *buffer profile* of $\nu_i$ given $\pi_i$ and $t_i$ as a pair $\mathsf{Buf}_i = (\partial_i, \xi_i)$. We say $\mathsf{Buf}_i$ and $\mathsf{Buf}_j$ are *fully compatible* if and only if 1) $\partial_i$ and $\partial_j$ do not overlap, 2) $\partial_i$ and $\xi_j$ do not overlap, and 3) $\partial_j$ and $\xi_i$ do not overlap. These conditions state that the safety buffer of a drone cannot overlap with the safety buffer and the wind buffer of another drone at any time. However, they allow the overlap of the wind buffers $\xi_i$ and $\xi_j$. We say $\mathsf{Buf}_i$ and $\mathsf{Buf}_j$ are *partially compatible* if and only if $\partial_i$ and $\partial_j$ do not overlap. In this work, it is sufficient to consider partial compatibility since we allow drones to enter the wind fields.

A timed formation plan $(\Pi, \Gamma)$ is *valid* if and only if the buffer profiles of every pair of drones are partially compatible. We define the drone formation planning problem as follows. Given 1) a set $\mathcal{V}$ of $n$ drones, 2) an initial formation $F_0$, and 3) a goal formation $F_{\mathsf{goal}}$, find a valid timed formation plan $(\Pi, \Gamma)$ such that $t_{\mathsf{end}} = \mathsf{makespan}(\Pi, \Gamma)$ is minimized while the formation at time $t_{\mathsf{end}}$ is $F_{\mathsf{goal}}$.

## IV. Safety Buffers and Wind Buffers

The shapes of safety buffers and wind buffers depend on the physical characteristics, the velocity, and the heading of drones. Safety buffers should generally be large enough for emergency stops when a drone senses an imminent crash. One way to determine the minimum stopping distance is to measure the overshooting distances of a drone by conducting experiments in which we issue a stop command to a drone flying at high speed. Fig. 3 shows the overshooting distance of our drone when we issued a stop command to stop our drone flying at speed $v$ and at angle $\theta$ relative to the horizontal plane. Thus, the minimum stopping distance is the overshooting distance. The safety buffer can be defined using a list of cylindrical shapes, each slightly larger than the drone. More precisely, the safety buffer is the union of all cylindrical shapes along a straight-line path whose length is the minimum stopping distance and whose direction is $\theta$

The wind buffer is defined in the same way, except that the cylindrical shape is replaced by a different shape. We used a handheld anemometer to measure the maximum wind speed at locations around and below our drone in a stationary pose. At each location, we rotated the anemometer to measure the wind speed at different angles and then identified the maximum wind speed at the location. The set of anemometer readings constitutes an estimation of the wind field around the drone. Fig. 4 shows the wind field of our drone at 1.5 m above the ground. Then, we define a shape larger than all locations with a maximum wind speed larger than a certain threshold $V_{\mathsf{wind}}^{\mathsf{min}}$. For simplicity, we define the shape as a union of simple geometrical shapes such as cylinders and cones. The wind buffer is the union of the sequences of this shape along a straight-line path whose length is the minimum stopping distance and whose direction is $\theta$. We shall assume the wind field of a drone in non-stationary poses can be larger than the wind field in stationary poses. Thus, the geometrical shapes will scale with the speed of the drone. We ignore the effect of the environment on wind fields by assuming that drones will not fly too close to walls or ground. For more information about wind field modeling, please refer to [17], which analyzes the aerodynamic effects of UAVs in the proximity of walls and ground.

## V. The Reservation Grid

As in AIM [4], [8], we define a *reservation grid*, which subdivides the 3D workspace into a set of equally-sized *cells*, each of which is a small cubic region that drones can reserve. The timeline is also discretized into a sequence of equal-length *time steps*. A *tile* is a pair $(c, t)$, where $c$ is a cell and $t$ is a time step. Unlike AIM, our reservation grid has

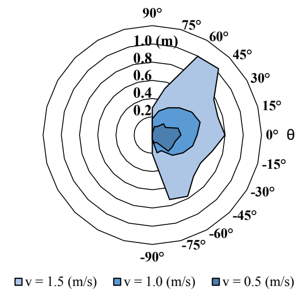

Fig. 3: The overshooting distance when our drone flies at speed $v$ and at angle $\theta$.

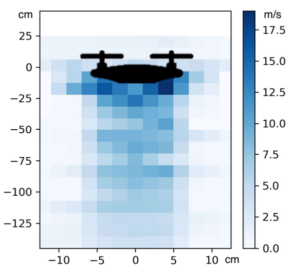

Fig. 4: A wind field of our drone at 1.5 m above the ground.

two kinds of tile reservations. In a *non-exclusive* reservation, a tile can be non-exclusively reserved by multiple drones: after a drone non-exclusively reserves a tile, other drones can still non-exclusively reserve the tile. In an *exclusive* reservation, a tile is solely reserved by one drone; after a drone exclusively reserves a tile, other drones cannot non-exclusively or exclusively reserve the tile.

We utilize the reservation grid to quickly check the compatibility of buffer profiles. Given a buffer $b_t$ at time $t$, let $T_{b_t}$ be the set of tiles whose cells overlap with $b_t$ at $t$. Given a buffer profile $\mathsf{Buf}_i = (\partial_i, \xi_i)$, the set of all tiles overlapping with $\partial_i$ is $\mathsf{Tile}_i^{\partial} = \bigcup_{b_t \in \partial_i, 0 \leq t \leq t_{\mathsf{end}}} \{T_{b_t}\}$, whereas the set of all tiles overlapping with $\xi_i$ is $\mathsf{Tile}_i^{\xi} = \bigcup_{b_t \in \xi_i, 0 \leq t \leq t_{\mathsf{end}}} \{T_{b_t}\}$. We call $\mathsf{Tile}_i^{\partial}$ and $\mathsf{Tile}_i^{\xi}$ the *safety buffer's footprints* and the *wind buffer's footprints* of $\nu_i$, respectively. We can check the compatibility of two buffer profiles $\mathsf{Buf}_i$ and $\mathsf{Buf}_j$ of two drones $\nu_i$ and $\nu_j$ by 1) exclusively reserving all tiles in $\mathsf{Tile}_i^{\partial}$ and $\mathsf{Tile}_j^{\partial}$ and 2) non-exclusively reserving all tiles in $\mathsf{Tile}_i^{\xi}$ and $\mathsf{Tile}_j^{\xi}$. If the reservations are successful, $\mathsf{Buf}_i$ and $\mathsf{Buf}_j$ are fully compatible.[4] Likewise, we can check partial compatibility by exclusively reserving all tiles in $\mathsf{Tile}_i^{\partial}$ and $\mathsf{Tile}_j^{\partial}$. Hence, we can check the validity of a timed formation plan by making reservations for the drones' footprints.

## VI. Modeling the Deviated Trajectories

Partial compatibility is required when a drone $\nu_i$ has to enter the wind field of another drone $\nu_j$ without collision. We would like to let one drone enter the wind field of another drone in a controlled way to reduce the makespan. To this end, we model how the drones' trajectories deviate from their intended trajectories under the influence of the wind field. Note that the influence is mutual—while $\nu_i$ is blown away by $\nu_j$, $\nu_j$ could be put upward slightly.

Due to the complexity of multi-drone interactions, we opt for a data-driven approach—we trained a deep learning model to predict the trajectories of multiple agents from their past trajectories. Social LSTM [1] was first proposed to predict the future trajectories of people based on their

[4]Strictly speaking, $\mathsf{Buf}_i$ and $\mathsf{Buf}_j$ are *approximately* compatible since two buffers do not necessarily overlap even if they overlap with the same tile—the buffers can occupy different parts of the cell. However, these false positives are acceptable in our applications. We can avoid some of these false positives by reducing the size of the cells in the reservation grid.

past positions. [23] added an attention mechanism to Social LSTM, which captures the relative importance of each person when navigating in the crowd. [10] replaced LSTM with transformer networks for trajectory forecasting. [13] is a new take on the multi-modal prediction of pedestrian trajectories using transformer networks and graph convolutional networks. In our experiments, we trained a Social LSTM model with an attention mechanism described in [23]. The training data were generated by a simulator we developed. The simulator can simulate drone trajectories when they fly toward each other on random trajectories under the influence of the wind fields. Although the model can only be used to predict the trajectories of a small number of drones, we use model averaging to combine the outputs of the model for many random subsets of drones. Finally, we trained a model that can take the prefixes of the trajectories of an arbitrary number of drones and generate the predicted trajectories of the drones by taking the wind field effects into account. The predicted trajectories will then be used to make exclusive reservations in the reservation grids for collision avoidance.

## VII. GOAL SWAPPING IN FORMATION PLANNING

Wang and Rubenstein recently introduced a distributed shape formation algorithm that allows a swarm of robots to form a user-specified shape quickly [24]. This algorithm uses a goal selector to distribute the goals among the swarm for scheduling the collision-free paths. Each agent uses local communication to refine the goal assignment and control its position in a distributed fashion. This algorithm is a provably correct, fully distributed shape formation algorithm that can provide collision-free and deadlock-free guarantees, requiring the use of local communication only. Inspired by this goal-swapping approach, we devise a version of the algorithm for centralized control settings. We improve the goal-swapping technique by incorporating wind field modeling and the reservation grid in the algorithm.

### A. Scheduling of Formation Plans

Given two motion plans $\pi_i$ and $\pi_j$, let $\delta_{i,j}$ be the *minimum* time such that the buffer profile $\mathsf{Buf}_i$ of $\nu_i$ given $\pi_i$ and the start time $t_i$ is partially compatiable with the buffer profile $\mathsf{Buf}_j$ of $\nu_j$ given $\pi_j$ and $t_j = t_i + \delta_{i,j}$, where $\delta_{i,j} \geq 0$. In other words, if we delay the execution of $\pi_j$ for at least time $\delta_{i,j}$ after the start time of $\nu_i$, the buffer profiles of both drones will be partially compatible; otherwise, they will not be partially compatible if $\pi_j$ starts after $\pi_i$ but before $t_i + \delta_{i,j}$. We can compute $\delta_{i,j}$ by setting $t_i = 0$ and increasing $\delta_{i,j}$ from 0 to $|\mathsf{Traj}[\pi_i]| - 1$ until $\mathsf{Buf}_i$ and $\mathsf{Buf}_j$ are partially compatible. Notice that if $\mathsf{Buf}_i$ and $\mathsf{Buf}_j$ are still incompatible when $\delta_{i,j} \geq |\mathsf{Traj}[\pi_i]|$, it is impossible that $\mathsf{Buf}_i$ and $\mathsf{Buf}_j$ are partially compatible and $\delta_{i,j}$ does not exist. It can happen when the last pose of $\nu_i$ after executing $\pi_i$ blocks $\nu_j$ from reaching its goal pose with $\pi_j$ indefinitely. If both $\delta_{j,i}$ and $\delta_{j,i}$ do not exist (e.g., when $\nu_i$ and $\nu_j$ fly toward each other in a straight line), the buffer profiles of $\nu_i$ and $\nu_j$ will never be partially compatible.

**Algorithm 1** Find a schedule $\Gamma$ that minimizes the makespan $\mathsf{makespan}(\Pi, \Gamma)$ for a given formation plan $\Pi$.

```
1:  procedure OptimizeSchedule($\Pi$)
2:    $\Gamma_{\mathsf{min}} := \emptyset$; $l_{\mathsf{min}} := \infty$
3:    for $Restart = 1$ to $Restart_{\mathsf{max}}$ do
4:      $K := 1$; $\rho := \langle k_i \rangle_{1 \leq i \leq n}$ is a random permutation of $1..n$
5:      while $K \leq K_{\mathsf{max}}$ do
6:        if $\delta_{k_j,k_i}$ exists for all $1 \leq i < j \leq n$ then
7:          $t_{k_1} = 0$; $t_{k_j} = \max_{1 \leq i < j}\{t_{k_i} + \delta_{k_i,k_j}\}$ for $1 < j \leq n$
8:          $\Gamma := \{t_{k_j}\}_{1 < j \leq n}$; $l := \mathsf{makespan}(\Pi, \Gamma)$
9:          if $l < l_{\mathsf{min}}$ then $\Gamma_{\mathsf{min}} := \Gamma$; $l_{\mathsf{min}} := l$; $K := 0$
10:       if $K > 0$ then undo the last swap in Line 11 if any
11:       $K := K + 1$; Randomly swap two elements in $\rho$
12:   return $\Gamma_{\mathsf{min}}$
```

**Algorithm 2** The use of goal swapping to find $\Pi_{\mathsf{min}}$ and $\Gamma_{\mathsf{min}}$ that minimize the makespan.

```
1:  procedure OptimizeMakespanByGoalSwapping
2:    $\Pi_{\mathsf{min}} := \emptyset$; $\Gamma_{\mathsf{min}} := \emptyset$; $l_{\mathsf{min}} := \infty$; $K := 1$
3:    $\Pi$ is the initial formation plan.
4:    while $K \leq K_{\mathsf{max}}$ do
5:      $\Gamma :=$ OptimizeSchedule($\Pi$); $l := \mathsf{makespan}(\Pi, \Gamma)$
6:      if $\Gamma \neq \emptyset$ and $l < l_{\mathsf{min}}$ then
7:        $\Pi_{\mathsf{min}} := \Pi$; $\Gamma_{\mathsf{min}} := \Gamma$; $l_{\mathsf{min}} := l$; $K := 0$
8:      if $K > 0$ then undo the last goal swap in Line 10 if any
9:      if some $\nu_1$ and $\nu_2$ satisfy the goal swapping rule then
10:       Swap the goals of $\nu_1$ and $\nu_2$ and update $\Pi$ accordingly
11:     $K := K + 1$;
12:   return $(\Pi_{\mathsf{min}}, \Gamma_{\mathsf{min}})$
```

The following theorem states a sufficient condition for the validity of a timed formation plan.

*Theorem 1:* A timed formation plan $(\Pi, \Gamma)$ is valid if there exists an ordering $\langle \nu_{k_1}, \nu_{k_2}, \ldots, \nu_{k_n} \rangle$ of drones such that for any $1 \leq i < j \leq n$, $\delta_{k_i,k_j}$ exists and $t_{k_j} \geq t_{k_i} + \delta_{k_i,k_j}$.

*Proof:* Since $\delta_{k_i,k_j}$ is non-negative, we have $t_{k_j} \geq t_{k_{j-1}}$ for $1 < j \leq n$. Thus, $t_{k_1} \leq t_{k_2} \leq \ldots \leq t_{k_n}$, which means the motion plans will be executed in the ordering $\langle \nu_{k_1}, \nu_{k_2}, \ldots, \nu_{k_n} \rangle$. Moreover, the buffer profile $\mathsf{Buf}_{k_i}$ of $\nu_{k_i}$ given $\pi_{k_i}$ and time 0 is partially compatiable with the buffer profile $\mathsf{Buf}_{k_j}$ of $\nu_{k_j}$ given $\pi_{k_j}$ and time $(t_{k_j} - t_{k_i})$ for $1 \leq i < j \leq n$, since $t_{k_j} - t_{k_i} \geq \delta_{k_i,k_j}$. Therefore, the buffer profiles of every pair of drones are partially compatible, and $(\Pi, \Gamma)$ is valid. ■

Theorem 1 suggests a method for finding a schedule $\Gamma$ for a given formation plan $\Pi$ such that $(\Pi, \Gamma)$ is valid: find an ordering $\langle \nu_{k_1}, \nu_{k_2}, \ldots, \nu_{k_n} \rangle$ such that $\delta_{k_j,k_i}$ exists for any $1 \leq i < j \leq n$. Then we can compute $\Gamma = \{t_{k_j}\}_{1 \leq j \leq n}$ where $t_{k_j} = \max_{1 \leq i < j}\{t_{k_i} + \delta_{k_i,k_j}\}$ and $t_{k_1} = 0$ such that the timed formation plan $(\Pi, \Gamma)$ is valid.

In practice, it is easy to find an ordering that makes $(\Pi, \Gamma)$ valid since many random permutations of drones would work. However, to find $\Gamma$ that yields the *minimum* makespan for a given $\Pi$, we have to enumerate *all* possible permutations of drones such that $\delta_{k_j,k_i}$ exists and $t_{\mathsf{end}} = \mathsf{makespan}(\Pi, \Gamma)$ is minimum. The time complexity of this exhaustive search is exponential to $n$, making it impractical even when $n$ is small. Therefore, we devise a local search algorithm to return a suboptimal solution in polynomial time. Algorithm 1 is the pseudocode of our local search algorithm. The algorithm is based on the famous min-conflict heuristic, which is highly effective in many combinatorial optimization problems [16], [19]. Initially, the algorithm randomly generates a permutation $\rho$ of the indexes of drones (Line 4). Then it iteratively modifies $\rho$ by swapping the indexes in $\rho$ (Line 11) to see whether the minimum makespan $l_{\mathsf{min}}$ can be reduced (Line 9). If $l_{\mathsf{min}}$ cannot be reduced after $K_{\mathsf{max}}$ swaps (Line 5), the local search reaches a local minimum, and then it uses the random restart strategy to get out of local minima (Line 3). Eventually, the algorithm returns $\Gamma_{\mathsf{min}}$, the schedule that gives a suboptimal makespan.

### B. Minimizing Makespans by Goal Swapping

We can further reduce the makespan by utilizing the goal swapping technique in [24]. We define a *goal swapping rule*, which takes $\nu_1$, $\nu_2$, $\Pi$, and $\Gamma$ as inputs, and check whether the makespan of the two drones (ignoring all other drones) can be reduced if they swap their goals. More precisely, let $\rho_1^{\mathsf{init}}$ and $\rho_2^{\mathsf{init}}$ be the initial poses of $\nu_1$ and $\nu_2$, and let $\rho_1^{\mathsf{goal}}$ and $\rho_2^{\mathsf{goal}}$ be the current goal poses of $\nu_1$ and $\nu_2$, respectively. Let $\pi_1$ and $\pi_2$ be the motion plans for $\nu_1$ and $\nu_2$ in $\Pi$ to move from $\rho_1^{\mathsf{init}}$ and $\rho_2^{\mathsf{init}}$ to $\rho_1^{\mathsf{goal}}$ and $\rho_2^{\mathsf{goal}}$, respectively. Let $t_1$ and $t_2$ be the start times for $\nu_1$ and $\nu_2$ in $\Gamma$, respectively. Suppose the goal of $\nu_1$ changes to $\rho_2^{\mathsf{goal}}$ and the goal of $\nu_2$ changes to $\rho_1^{\mathsf{goal}}$. The goal swapping rule computes the new motion plans $\pi_1'$ and $\pi_2'$ and the new start times $t_1'$ and $t_2'$. If $\max(t_1' + |\mathsf{Traj}[\pi_1']|, t_2' + |\mathsf{Traj}[\pi_2']|) < \max(t_1 + |\mathsf{Traj}[\pi_1]|, t_2 + |\mathsf{Traj}[\pi_2]|)$, we say $\nu_1$ and $\nu_2$ *satisfy* the goal swapping rule, and we can swap the goals of $\nu_1$ and $\nu_2$.

However, the goal-swapping rule can make wrong decisions if it ignores the wind field effect. For example, suppose two drones fly horizontally in parallel in the opposite direction at the same start time, as shown in Fig. 5. If $D_1$ is slightly smaller than $D_2$, the drones do not swap their goals according to the goal-swapping rule. However, if the wind field of Drone A pushes Drone B downward, it causes Drone B to fly in the blue trajectory, which is much longer than $D_1$, the length of the dotted red trajectory. Then, swapping the goals can reduce the makespan of these two drones since this eliminates the wind field effect. This simple example shows that the goal-swapping rule could make wrong decisions if it does not consider the deviated trajectories. In general, the same mistake occurs in more complicated scenarios since the trajectories' lengths are underestimated when the wind field effect is ignored. Therefore, it is necessary to predict the deviated trajectories when using the goal-swapping rule.

Algorithm 2 utilizes the goal-swapping rule to minimize the makespan by local search. Initially, all drones fly toward their goal poses according to the initial formation plan $\Pi$ (Line 3). The algorithm randomly chooses a pair of drones that satisfies the goal swapping rule (Line 9) and then swaps their goals and updates $\Pi$ accordingly (Line 10). After that, it calls OptimizeSchedule to find a schedule $\Gamma$ for $\Pi$ and compute the makespan (Line 5). If the makespan is reduced,

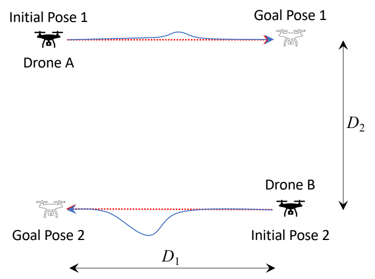

Fig. 5: A wrong decision made by the goal-swapping rule.

the algorithm keeps $\Pi$ and $\Gamma$ as the current best solution (Line 7); otherwise, the goal swap is reverted (Line 8). The above process repeats $K_{\mathsf{max}}$ times, and then the algorithm returns the best solution.

## VIII. EXPERIMENTAL EVALUATION

We conducted two experiments to evaluate our algorithms. In Experiment 1, we implemented a drone swarm simulator to compare our algorithm with three baseline algorithms: 1) the intersection management (IM) approach in which drones stop and wait if another drone blocks its way [24], 2) the buffered Voronoi Cell (BVC) method [29] and 3) the artificial potential field (APF) methods [20]. In Experiment 2, we repeated Experiment 1 with a team of small-sized quadcopters called Crazyflie [18] in the real world.

**Experimental Setup** Experiment 1 was based on a drone simulator we developed for aerial swarm research. The simulator was implemented in C++23 with the OpenGL and SDL2 libraries. We implemented a simple drone dynamics model in which the drone can move freely in any direction subject to some speed and acceleration constraints. We retained the simulation's realism by tuning the model's parameters based on the performance of a real drone we built in our lab. The wind model was based on the wind speed measurements around the real drone using an anemometer as described in Sec. IV. The simulator can instantiate $n$ drones and take an initial formation $F_{\mathsf{init}}$ and a goal formation $F_{\mathsf{goal}}$ as inputs. All poses in both $F_{\mathsf{init}}$ and $F_{\mathsf{goal}}$ cannot intersect with each other. The simulator controlled the simulated drones by a controller that acts according to a timed formation plan generated by our formation planning algorithm or a baseline controller such as those in the APF methods. The simulator can process the drag and check any collision in real-time. When all drones arrive at the goal poses, the simulator will report the total execution time of the simulation as the makespan.

We implemented Algorithm 1 and Algorithm 2 as well as the three baseline algorithms in the simulator. We implemented the intersection management approach in which a drone will move toward its goal pose in a straight line, but it will stop when a collision is imminent. A stopped drone will resume its flight when its path is clear. We implemented the buffered Voronoi Cell method based on the reference implementation in the Crazyflie system [18], which has implemented the method. We also implemented the APF method in which a drone will move according to a potential field function derived from the potential field function of its goal pose minus a weighted sum of the potential field functions of other drones. A drone will move towards its goal pose by following the gradient of the potential field.

To highlight the effect of allowing drones to enter wind fields (EWF) and goal-swapping (GW), we implemented four versions of our algorithm for the four combination of these features. To forbid drones from entering any wind fields, we replace the partial compatibility condition with the full compatibility condition when checking the validity of timed formation plans using the reservation grid described in Sec. III and Sec. V. To turn off goal swapping, we simply do not use Algorithm 2 to find the timed formation plans and instead use Algorithm 1 to find a schedule $\Gamma$ for $\Pi$.

While our implementation of the neural network closely mirrors the implementation in [23], we have to create a custom dataset of trajectories and modify a few parameters to make it suitable for inferring drone trajectories. We used a discretization of our arena of $5m \times 5m \times 3m$ motion capture space into a grid corresponding to $500 \times 300$ images. Position trajectories from the simulator were sampled at $0.01s$ intervals, which also served as the basis for defining a unit sequence length for the neural network. 500 trajectories of sequence length of $5s$ and 10 agents in constant velocity motion were generated with simulated wake interaction, which was preprocessed based on the discretization parameters and normalization. During training, a dataloader was used to sample a data sequence for training and validation, which used negative log-likelihood loss under the predicted gaussian distribution. During inference, the first $0.5s$ of the straight-line trajectory sequence was used as the observed sequence, assuming that all drones began their schedule and were in motion.

In Experiment 2, we used a drone swarm system called Crazyflie 2.1 nano quadcopters to evaluate our algorithms in the real world. We used six Crazyflie drones, each of which was equipped with infrared motion tracking markers such that its position can be tracked by an Optitrack motion capture system with ten infrared cameras that offer a sub-millimeter accuracy. We ran our algorithms on a desktop computer that remotely controls the drones via a 2.4 GHz communication channel. The 168MHz ARM microcontrollers on the drones will control the drones.

**Results** Fig. 6 shows the results of Experiment 1. As can be seen, our algorithms always have lower average makespans when compared with the IM, BVC, and APF approaches, regardless of the number of drones. These baseline algorithms are reactive planners and do not conduct long-term planning. Hence, our planning approach clearly outperforms them in terms of minimizing the makespans.

Among the four versions of our algorithm, the version that allows drones to enter wind fields with goal swapping is the best. In fact, it is better than the other versions in two different ways. First, when compared with the versions in which drones were forbidden to enter any wind fields, the best version outperformed them by a large margin. Clearly, it is beneficial to allow drones to enter the wind fields of other drones such that there are more ways to minimize the makespan. Second, when compared with the versions without goal swapping, the best version also outperformed them. In fact, goal swapping is better with and without the permission of entering wind fields. Therefore, goal swapping is indeed helpful in reducing the makespans. Moreover, goal swapping can yield a much larger performance improvement than allowing drones to enter wind fields.

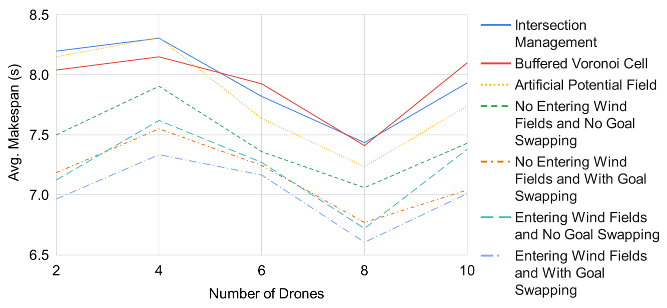

Fig. 6: Average makespan vs. the number of drones.

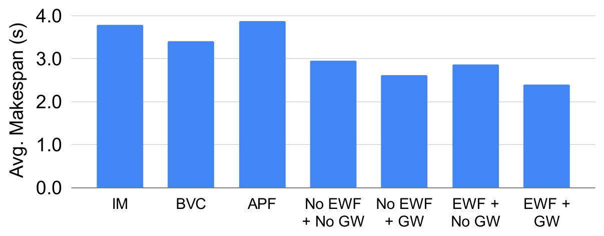

Fig. 7: The average makespans in the real drone systems.

We can draw the same conclusion from the results of Experiment 2 with the real drones. Fig. 7 shows that the average makespans in Experiment 2 with the real drones are smaller than that in Experiment 1 due to the limited size of the room in which we conducted the experiment. Nonetheless, our algorithm outperformed the baseline algorithms in Experiment 2. Among the four versions of our algorithm, the version that allows drones to enter wind fields with goal swapping was still the best.

## IX. Summary and Future Work

We have presented a new grid-based formation planning algorithm for moving a fleet of drones in formation from one location to another. Our approach utilizes a reservation grid for collision avoidance with respect to drones' safety buffers and wind fields. Wind buffers can simplify the trajectory optimization process by ignoring the irregularity of the shape of the wind fields. Our planning algorithm is a local search algorithm that aims to find a schedule of the start times of the motion plans in a formation plan with goal swapping. Our experiments with real and simulated drones show that if we allow drones to fly into the wind field of another drone, we could further reduce the makespan. However, this maneuver depends on the prediction of the altered trajectory under the influence of wind fields. Therefore, a possible future work is to use reinforcement learning to find an optimal policy for formation planning with respect to wind fields.

## References

[1] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social LSTM: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[2] Javier Alonso-Mora, Andreas Breitenmoser, Martin Rufli, Roland Siegwart, and Paul Beardsley. Image and animation display with multiple mobile robots. *The International Journal of Robotics Research*, 31(6):753–773, 2012.

[3] Olov Andersson, Mariusz Wzorek, Piotr Rudol, and Patrick Doherty. Model-predictive control with stochastic collision avoidance using bayesian policy optimization. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4597–4604, 2016.

[4] Tsz-Chiu Au, Chien-Liang Fok, Sriram Vishwanath, Christine Julien, and Peter Stone. Evasion planning for autonomous vehicles at intersections. In *IEEE/RSJ International conference on Intelligent Robots and Systems*, pages 1541–1546, 2012.

[5] Vaibhav Bajaj and Sachit Rao. Autonomous shape formation and morphing in a dynamic environment by a swarm of robots. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1765–1767, 2020.

[6] Marcello Cirillo. From videogames to autonomous trucks: A new algorithm for lattice-based motion planning. In *IEEE Intelligent Vehicles Symposium*, pages 148–153, 2017.

[7] Marcello Cirillo, Tansel Uras, and Sven Koenig. A lattice-based approach to multi-robot motion planning for non-holonomic vehicles. In *IEEE/RSJ International conference on Intelligent Robots and Systems*, pages 232–239, 2014.

[8] Kurt Dresner and Peter Stone. A multiagent approach to autonomous intersection management. *Journal of Artificial Intelligence Research (JAIR)*, March 2008.

[9] Haibin Duan, Qinan Luo, Yuhui Shi, and Guanjun Ma. Hybrid particle swarm optimization and genetic algorithm for multi-uav formation reconfiguration. *IEEE Computational Intelligence Magazine*, 8(3):16–27, 2013.

[10] Francesco Giuliari, Irtiza Hasan, Marco Cristani, and Fabio Galasso. Transformer networks for trajectory forecasting. In *International Conference on Pattern Recognition (ICPR)*, pages 10335–10342, 2020.

[11] Florence Ho, Ruben Geraldes, Artur Goncalves, Marc Cavazza, and Helmut Prendinger. Improved conflict detection and resolution for service UAVs in shared airspace. *IEEE Transactions on Vehicular Technology*, 68(2):1231–1242, 2018.

[12] Jie Huang, Guoqing Tian, Jiancheng Zhang, and Yutao Chen. On unmanned aerial vehicles light show systems: Algorithms, software and hardware. *Applied Sciences*, 11(16):7687, 2021.

[13] Yao Liu, Lina Yao, Binghao Li, Xianzhi Wang, and Claude Sammut. Social graph transformer networks for pedestrian trajectory prediction in complex social scenarios. In *ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1339–1349, 2022.

[14] Pengda Mao, Yan Gao, Bo Wang, An Yan, Xiaoyu Chi, and Quan Quan. Fast light show design platform for k-12 children. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9396–9402. IEEE, 2021.

[15] Yan Meng, Hongliang Guo, and Yaochu Jin. A morphogenetic approach to flexible and robust shape formation for swarm robotic systems. *Robotics and Autonomous Systems*, 61(1):25–38, 2013.

[16] Steven Minton, Mark D. Johnston, Andrew B. Philips, and Philip Laird. Minimizing conflicts: A heuristic repair method for constraint-satisfaction and scheduling problems. *Artificial Intelligence*, 58(1–3):161–205, 1992.

[17] C. Paz, E. Suárez, C. Gil, and C. Baker. CFD analysis of the aerodynamic effects on the stability of the flight of a quadcopter UAV in the proximity of walls and ground. *Journal of Wind Engineering and Industrial Aerodynamics*, 206:104–378, 2020.

[18] James A. Preiss, Wolfgang Hönig, Gaurav S. Sukhatme, and Nora Ayanian. Crazyswarm: A large nano-quadcopter swarm. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3299–3304, 2017.

[19] Rok Sosič and Jun Gu. Efficient local search with conflict minimization: A case study of the $n$-queens problem. *Knowledge and Data Engineering*, 6(5):661–668, 1994.

[20] Hang Sun, Juntong Qi, Chong Wu, and Mingming Wang. Path planning for dense drone formation based on modified artificial potential fields. In *2020 39th Chinese Control Conference (CCC)*, pages 4658–4664, 2020.

[21] Sebastian Thrun and Arno Bücken. Integrating grid-based and topological maps for mobile robot navigation. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 944–950, 1996.

[22] Jur Van Den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *Robotics Research: The 14th International Symposium ISRR*, pages 3–19. Springer, 2011.

[23] Anirudh Vemula, Katharina Muelling, and Jean Oh. Social attention: Modeling attention in human crowds. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4601–4607, 2018.

[24] Hanlin Wang and Michael Rubenstein. Shape formation in homogeneous swarms using local task swapping. *IEEE Transactions on Robotics*, 36(3):597–612, 2020.

[25] Fangyu Wu, Dequan Wang, Minjune Hwang, Chenhui Hao, Jiawei Lu, Trevor Darrell, and Alexandre Bayen. Motion planning in understructured road environments with stacked reservation grids. In *ICRA Workshop on Perception, Action, Learning (PAL)*, 2020.

[26] Peng Yang, Ke Tang, Jose A Lozano, and Xianbin Cao. Path planning for single unmanned aerial vehicle by separately evolving waypoints. *IEEE Transactions on Robotics*, 31(5):1130–1146, 2015.

[27] Jingjin Yu and Steven M LaValle. Shortest path set induced vertex ordering and its application to distributed distance optimal formation path planning and control on graphs. In *52nd IEEE Conference on Decision and Control*, pages 2775–2780, 2013.

[28] Jingjin Yu and Steven M LaValle. Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics. *IEEE Transactions on Robotics*, 32(5):1163–1177, 2016.

[29] Dingjiang Zhou, Zijian Wang, Saptarshi Bandyopadhyay, and Mac Schwager. Fast, on-line collision avoidance for dynamic vehicles using buffered voronoi cells. *IEEE Robotics and Automation Letters*, 2(2):1047–1054, 2017.

[30] Hai Zhu and Javier Alonso-Mora. Chance-constrained collision avoidance for MAVs in dynamic environments. *IEEE Robotics and Automation Letters*, 4(2):776–783, 2019.