Comparison of Strategic Conflict Prevention Methods for Departure Planning in Drone Delivery

Vlaskin, Sasha; Sunil, Emmanuel; Nieuwenhuisen, Dennis; Ellerbroek, Joost; Hoekstra, Jacco

DOI
10.1109/DASC62030.2024.10749661

Publication date
2024

Document Version
Final published version

Published in
DASC 2024 - Digital Avionics Systems Conference, Proceedings

Comparison of Strategic Conflict Prevention Methods for Departure Planning in Drone Delivery

Sasha Vlaskin, Emmanuel Sunil, Dennis Nieuwenhuisen
Aircraft Operations, Air Traffic Management and Airports
Netherlands Aerospace Centre
Amsterdam, the Netherlands
{sasha.vlaskin, emmanuel.sunil, dennis.nieuwenhuisen}@nlr.nl

Joost Ellerbroek, Jacco Hoekstra
CNS/ATM, Control and Simulation, Control and Operations
TU Delft
Delft, the Netherlands
{j.ellerbroek, j.m.hoekstra}@tudelft.nl

2024 AIAA DATC/IEEE 43rd Digital Avionics Systems Conference (DASC) | 979-8-3503-4961-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/DASC62030.2024.10749661

Abstract*—U-Space drone operations are expected to be a driver for further urban development, especially through use cases such as medical and commercial parcel delivery. In particular, package delivery using small drones shows great promise, with e-commerce giants such as Amazon deploying limited-scale drone delivery trials in rural areas. As the technology matures, large-scale operations will take place in constrained urban areas, leading to high airborne traffic densities. It is necessary to develop a robust automated separation management system that actively ensures safe separation of drones both in the air and on the ground. This paper focuses on the Strategic element, more specifically on pre-departure planning. The aim of this is to reduce the chance of conflicts around vertiports, where spatial and environmental constraints make tactical resolutions difficult. This work focuses on two scenarios: a single pad for both takeoffs and landings (in a spatially constrained urban area) and 4 takeoff-landing pad pairs (for a distribution center). Several methods are compared for this takeoff sequencing task, coupled with a conflict detection algorithm: A First-Come First-Served method that applied delay to conflicting flights, a Mixed-Integer Programming approach, a Genetic Algorithm, Particle Swarm Algorithm and Simulated Annealing were used. For a single-pad approach, first-come first-served works best in terms of computation time and total deployment time (or makespan). For the multi-pad approach however, changing the flight sequence through metaheurisitic methods and mixed-integer linear programming show a reduction in total deployment time.

***Keywords*—drones, delivery, planning, BlueSky, U-Space, UTM, Strategic Deconfliction**

I. Introduction

The consumer and commercial drone markets are expected to grow significantly in the coming decades, with some 7 million total drones expected to fly within EU airspace alone by the year 2050 [1]. Drone delivery is expected to be a large driver for this, with companies such as Amazon with its Prime Air [2] Concept and Google with their hybrid eVTOL Wing drones investing heavily in the sector. In order to have an economically viable solution, high throughput and therefore high traffic densities will be required, as forecasts indicate that thousands of simultaneous deliveries could occur. In order to support this growth in a structured fashion, all facets of U-Space [3] need to be thought out and implemented. One of these main concerns is the set of methods used to keep drones at a safe distance from each other, or what is referred to as Separation Management. Significant research contributions to the field have been made in aspects such as Airspace Design [4], Strategic Deconfliction, Tactical CD&R [5] and collision avoidance [6]. Nevertheless, a systems approach taking all layers into account can be beneficial.

This paper is a part of the effort to design such an integrated separation management system, and focuses on the pre-flight aspect of drone delivery. More specifically, this paper deals with the comparison of conflict-free takeoff scheduling methods for hybrid drone delivery in a constrained urban environment, but without a discretized route network. Two pad layouts are considered in this study. Firstly, a single pad layout for constrained environments where expansive takeoff/landing structures are not possible. Here, takeoffs and landings occur on the same pad, and therefore all potential conflict at the pad are considered. Secondly, a multi-pad structure is used, with 4 (takeoff and landing) pad pairs, representing a more complex structure optimised for higher throughput.

The main goal is therefore to provide a conflict-free takeoff schedule, a task for which meta-heuristic approaches such as Simulated Annealing, Particle Swarm Optimisation and a Genetic Algorithm approach are compared to a Mixed-Integer Programming approach and a First-Come First-Served delay approach. The resulting schedules are then compared in terms of resulting conflicts, losses of separation, and total deployment time, or makespan. For this, the BlueSky [7] Air Traffic Management simulator is used.

First, the optimisation problem and required methodology behind the sequencing methods will be discussed. This is followed by a detailed description of the simulated urban environment, mission profiles, the assumptions made, and the experiment setup. Subsequently, the outcomes of the fast-time simulations will are presented and analysed. The paper ends with the main conclusions of this study and recommendations for future work.

II. Methodology

This section will cover the Problem Definition, Sequencing Methods used, followed Urban Environment setup, Scenario Design and generation for verification of the solutions obtained in the BlueSky ATC simulator [7].

A. Optimization Problem Definition

The optimization's goal is to generate a conflict-free takeoff sequence from a single input: a vector of flight times of queued drones. The optimizer's *output* is limited to the takeoff times. This output is used along with known climb and loiter times (derived from drone performance) to determine the conflict count of a given solution. Since equal priority (or ready at t=0 flights are limited), a *Queue Length* is the maximum length of an input vector, or the length of the block of flights that can be freely sequenced. For instance, a queue length of 25 would mean 25 flights can be freely sequenced, and a scenario of 100 flights would be broken down in 4 sequenced blocks with this queue length. Conflicts here are defined as *scheduling* conflicts - that is, a loss of separation near the pad that is predicted given current scheduling.

B. Sequencing Methods

1) Evenly Spaced Takeoffs: Evenly spacing the takeoffs was attempted as a sanity check. Due to the takeoff-landing coupling for the single pad scenario, this is deemed infeasible since it yields excessive conflicts.

2) First-Come First-Served: This is the simplest method that takes conflicts into account. It does not change the sequence flights, so it uses a first-come-first-served approach. This is described by Algorithm 1.

Algorithm 1 First-Come First-Served Approach

```
$initialize$ takeoff time array $t_{takeoff}$
for all flights do
    $t_{next}$ = max($t_{takeoff}$) + 60
    perform initial $ConflictProbe$
    while conflicts are present do
        $t_{next}$ = $t_{next}$ + 15
        perform $ConflictProbe$
    end while
    append $t_{next}$ to $t_{takeoff}$
end for
```

Thus, a takeoff is initially scheduled 60 seconds after the previous flight. The resulting schedule is checked for conflicts, and if any are present, the flight is delayed by additional 15 seconds until it is conflict-free.

3) Mixed-Integer Linear Programming: A mixed integer linear programming approach was also used. The constraints formulated ensure that all landings and takeoffs are sufficiently spaced in order to avoid scheduling conflicts. Generating the constraints requires $\frac{n(n-1)}{2}$ unique pairs for potential conflicts. For every conflict pair, we have the takeoff times t_i, t_j such that

$$t_i, t_j \in R \cap (0, 80000) \tag{1}$$

and binary variables b_k such that

$$b \in R \cap (0, 1) \tag{2}$$

which are used to formulate the disjunctive constraints needed to have absolute value constraints (since the flight order does not matter). A big-M is also utilised here, meaning we use a value of M as 100,000 as a large penalty. The values of 80,000 and 100,000 were made on the basis that the typical makespan, or total duration, was found not exceed 80,000 seconds for 100-drone scenarios. Since the values are bounded to this as a maximum, this helps to narrow the search space. The first constraint, which ensures the difference between takeoff times t_i and t_j is of at least the buffer time B_t is defined as follows for every possible pair of drones:

$$t_i - t_j \geq B_t - b_k \cdot M$$
$$t_i - t_j \leq -B_t + (1 - b_k) \cdot M$$

This formulation is used here as the absolute value is needed, and disjunctive constraints such as these linearize the problem. Likewise for takeoff-landing conflicts, a similar constraint form is used, except order matters and 4 constraints per pair are needed:

$$t_i + t_i^{ret} - t_j \geq B_m - b_k \cdot M$$
$$t_i + t_i^{ret} - t_j \leq -B_m + (1 - b_k) \cdot M$$
$$t_i - t_j - t_j^{ret} \geq B_m - b_k \cdot M$$
$$t_i - t_j - t_j^{ret} \leq -B_m + (1 - b_k) \cdot M$$

where t_{ret} is the time to return to the parcel center after takeoff, incorporating climb, descent, loiter and flight times.

The final set of constraints is that for the landing-on-landing conflict prevention. This is again 2 constraints, formalised as:

$$t_i + 2 * t_i^f - t_j - 2 \cdot t_j^f \geq B_l - b_k \cdot M$$
$$t_i + 2 * t_i^f - t_j - 2 \cdot t_j^f \leq -B_l + (1 - b_k) \cdot M$$

where t^f is the flight time of the drone in cruise, excluding the takeoff, landing and loiter, since those are the same for both drones. Note that these are only used for the single-pad scenarios, where takeoff-on-landing conflicts can occur. In the multi-pad scenarios, only the first 4 constraints are needed.

The *objective* to minimize is simply the sum of takeoff times.

$$O = \sum_{0}^{n} t_i \tag{3}$$

In total, there are thus $8 \cdot \frac{n(n-2)}{2}$ constraints needed for a sequence of n drones, meaning that for example, for the longest sequencing queue length of 25 used we have 8 x 300 = 2400 constraints.

4) Genetic Algorithm: Genetic Algorithms are a family of bio-inspired optimisation algorithms which attempt to mimic nature by having solutions 'evolve' through natural selection after mutations [8]. The hyper-parameters for the algorithm use are seen in Table I. A cost function and hyper-parameters such as the number of generations, number of parents mating and others are needed. The fitness function used takes the form seen in Equation 4

$$\mathrm{F} = 1000 - 30 \cdot n_{conf} - 0.01 \cdot t_{tot} \tag{4}$$

where n_{conf} is the total number of *scheduling* conflicts and t_{tot} is the makespan achieved, or the total time from first takeoff to final landing at the origin pad. The hyper-parameters used are provided in Table I.

TABLE I: Hyper-parameters Used in Genetic Algorithm Implementation

Parameter	Value
Number of Generations	2000
Number of Genes	Equivalent to Queue Length
Solutions per Population	20
N. of Parents Mating	10
Mutation Percentage [%]	77
Keep Elitism	4
Mutation Type	random

5) Particle Swarm Optimisation: Particle Swarm Optimisation (PSO) [9] is another meta-heuristic algorithm, inspired by the social behaviour of birds flocking. It relies on particles, randomly initialised within the search space, exhibiting swarming behaviour to find the optimal solution. Equation 5 describes how the particle positions are updated.

$$\begin{aligned}\text{Velocity}_i(t+1) = w \cdot \text{Velocity}_i(t) + c_1 \cdot r_1 \cdot \\ (\text{Personal Best}_i - \text{Position}_i(t)) + c_2 \cdot r_2 \\ \cdot (\text{Global Best} - \text{Position}_i(t))\end{aligned} \tag{5}$$

$$\text{Position}_i(t+1) = \text{Position}_i(t) + \text{Velocity}_i(t+1) \tag{6}$$

The position update equations include the hyper-parameters c_1 and c_2, which are the $acceleration$ constants used in the velocity update phase of the algorithm. Finally, r_1 and r_2 are binary random variables. These hyper-parameters here dictate how the particles move through the search space.

The position, per particle, is then updated through Equation 6. This approach was expected to have similar solutions to the Genetic Algorithm and the Simulated Annealing implementation: however, the computation times were longer due the need to compute the cost per particle. The cost function used was as follows:

$$C = n_{conf} + 0.009 \cdot t_{tot} \tag{7}$$

where n_{conf} is the total number of conflicts and t_{tot} is the makespan. The hyper-parameters used for the implementation of this method are far fewer than those required for the Genetic Algorithm, therefore tuning them proved easier. These are seen in Table II.

TABLE II: PSO Hyper-parameters

Parameter	Value	Explanation
c_1	1.3	Kept high for faster runtime
c_2	1.3	Same as c_1
w	0.9	Inertia parameter

6) Simulated Annealing: Simulated Annealing [10] is a method inspired by metallurgy. A high 'temperature' relates to a large magnitude of the movement of the 'molecules' used at the start of the process to allow for the system to *explore* a wide range of solutions. As the temperature drops, the system becomes less likely to accept solutions with a higher cost, and therefore prioritises *exploitation*. The only hyper-parameter needed to initialize a method instance is the maximum number of iterations. This was set to 1000 for this work. More specifically, the Dual Annealing [11] implementation was used in this work. The cost function for this approach is given as follows:

$$C = n_{conf} + 0.001 \cdot t_{tot} \tag{8}$$

where n_{conf} is again the total number of conflicts and t_{tot} is the makespan.

III. Experiment Setup

Several sequencing methods have been compared in this paper. Sequences of 25, 50 and 100 flights were considered. In total, 5 different origin-destination pair datasets were generated for each sequence length (based on OSM [12] addresses). The planning algorithms at hand were run for planning queue lengths of 5, 10 and 25 flights whose sequence can be changed. This was done for a fully deterministic system (with simulation takeoff times unperturbed), a low uncertainty delay scenario, and a higher uncertainty delay scenario. Each combination was run for the 5 different scenarios for the given total number of drones.

A. Urban Environment Modelling and Mission Profile

In order to accurately represent the constraints that urban operations pose on drone operations, the urban environment needs to be defined. For this paper, the urban area of Amsterdam is simulated. To achieve this, the OpenStreetMap [12] Overpass [13] API is used is used to import all potential delivery point coordinates and relevant geo-data. Since the drones are assumed to operate between 50 and 100 meters above ground level (AGL) as per the U-Space concept [14], a safety margin of 20 meters is given. Thus, geofences are defined for all buildings taller than 30 meters, and all parcel center locations within the simulation environment are considered. The simulation area itself, including geofences, is visualised in Figure 1.

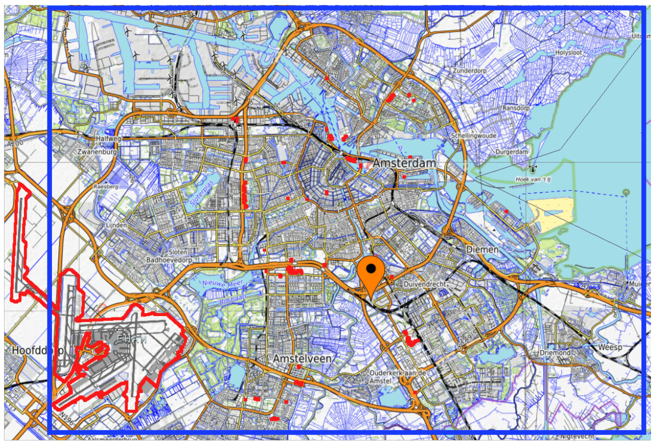

Figure 1. Amsterdam Simulation Area, with the Distribution Center indicated by an orange chevron, Simulation Bounds marked in blue and geofences in red. Data obtained from OSM [12]

For the route generation, unlike projects such as Metropolis II [15], this work assumes that street networks are not to be used as reference. Instead, the most direct route is planned, and all of the aforementioned geofences are considered and avoided using a method inspired by D. Rein-Weston's Branching Planner [16] such that minimal waypoints are added. The output is seen in Figure 2, where the geofence itself (in this case Schiphol Airport) is seen in cyan and the modified route in pink, with the destination being marked at the bottom of the figure with a label.

B. Airspace Design and Mission Profile

The airspace design featured two vertical layers: the *inbound* layer and the *outbound* layer, utilising two discrete altitude values for cruise. The outbound layer was set to 100 meters, and inbound to 50, in conformity with the expected Very-

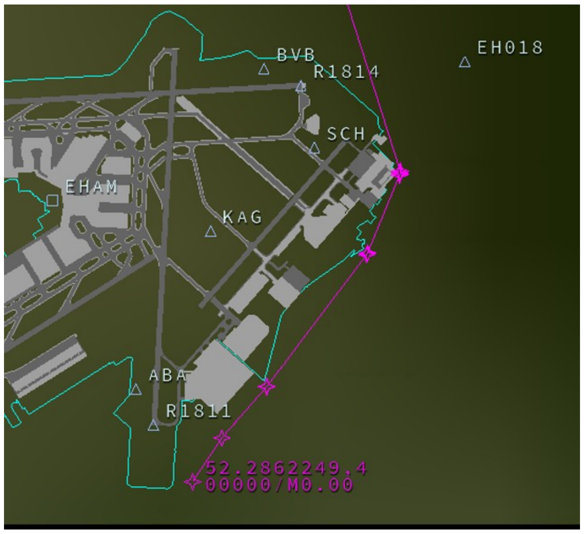

Figure 2. Geofence Route Solution around Amsterdam Schiphol Airport

Low-Level airspace bounds and in compliance with U-Space legislation [14]. A flight would take place as follows:

1) Drone is loaded with payload, checked, and begins vertical climb to the outbound layer at 100 meters height above ground level (AGL)
2) Drone transitions to fixed-wing and accelerates to cruise, flying all pre-programmed waypoints to its destination
3) Drone slows down just short of the destination, transitioning to hover.
4) Drone descends to delivery height.
5) Drone loiters at delivery location for 10 seconds until delivery is completed
6) Drone proceeds to climb again, this time to 50m AGL. Then it transitions to cruise and flies back to the distribution center.
7) The drone is cleared off the pad within 60 seconds of landing.

The mission profile is visualised in Figure 3.

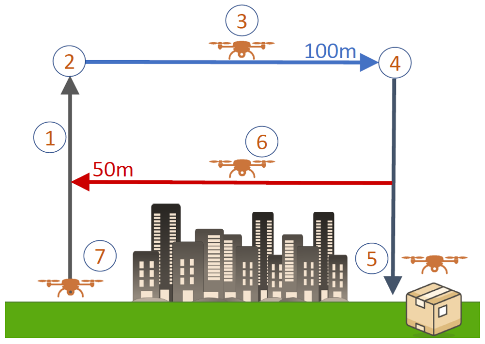

Figure 3. Visualisation of Mission Profile Used

The Origin-Destination pairs, and corresponding routes, are generated before the flight sequencing takes place by sampling from the delivery location database, obtained from OpenStreetMap [12]. For every number of flights, 5 random origin-destination pair sets are selected, with pre-computed geofence-free routes.

C. Pad Layout

There are two pad layout used in this study. There are:

1) *Single Multi-mode pad:* The first layout studied is a singular pad, used for both takeoffs and landings. This layout is used to simulate a heavily spatially constrained operation, such as a shop near the city center or a restaurant using drone delivery. For this, it is expected that the single pad resource will be the greatest constraint to operations.
2) *Multi-pad setup:* For less constrained zones, such as parcel fulfilment centers or drone vertiports, where more pads can be placed, a layout of 4 pad pairs (one takeoff and one landing) is proposed. This pad layout can be visualised in Figure 4.

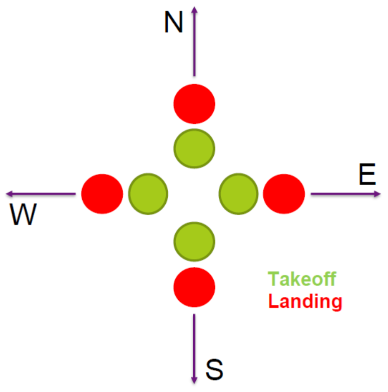

Figure 4. Multi-pad layout

Before the optimisation occurs, the flight queues are split by heading. Since there are 4 pads with a Northward orientation for the upper pad, the assignment is done to Pad 1 if the outbound flight heading is in the ranges (315,360) and (0,45), Pad 2 if the heading is (45,135), Pad 3 with (135,225) and finally Pad 4 with (225, 315) degrees.

D. Independent Variables

- *Sequencing Method:* the method used to strategically de-conflict the routes, this can be First-Come First-Served Delay (FC), Mixed Integer Programming (MIP), Genetic Algorithm (GA), Particle Swarm Optimisation (PSO), Simulated Annealing (SA)
- *Number of Drones to Plan:* 25, 50 and 100 were used.
- *Length of Planning Queue:* 5, 10 and 25 were used.

E. Dependent Variables

The dependent variables for the set of experiments are summarised here.

- *Makespan*: the *makespan* is the total time it takes for all drones in a given scenario to make their deliveries and return. This is also the total scenario duration, from first takeoff to final landing.
- *Number of Conflicts at Pad*: the number of conflicts at pad is simply the number of conflicts that occurs when a drone is taking off or landing. These are due to scheduling errors.

F. Control Variables

The control variables for the experiments were as follows:

- *Pad Blocking Time*: the time (assumed equal for takeoff and landing) that the pad is blocked for at each takeoff and landing. The blocking time here is 60 seconds.
- *Number of Pads and Positions*: A single mixed-use landing and takeoff pad is used for the main experimental runs
- *Routing method used:* this dictated how the routes were generated
- *Pad Layout used:* single or multi-pad

IV. Results and Discussion

The schedules resulting from each of the algorithms were tested with a Conflict Detection algorithm that checks for planning conflicts. This algorithm is therefore present in the cost/objective functions for the methods where that is required, namely the metaheuristic methods. Subsequently, these schedules were outputted to scenario files for the BlueSky Open ATM Simulator [7] and validated for conflicts at the pad, LoS and the makespans (total duration from first deployment to last landing of the drones). The sequence queue length is introduced as a means of limiting the size of the input vectors for the meta-heuristic algorithms. Thus, for 100 flights to plan, a queue length of 25 would mean that 4 blocks of sequenced flights occur after each other - the optimisation algorithm is run independently on the 4 resulting 25-flight blocks. The ranges of values used for this were 5, 10 and 25. In the plots, Simulated Annealing is referred to as “SA”, First-Come First-Served as “FC”, Particle Swarm Optimisation as “PSO”, Mixed-Integer Programming as “MIP” and finally the Genetic Algorithm as “GA”.

A. Varying Number of Drones to Plan: Single Pad

First, the total conflicts and makespans for a total demand of 100 fligths are plotted in Figures 5, 6. The queue lengths of 5 and 10 are omitted here as they exhibit worse results in both conflict counts and makespans.

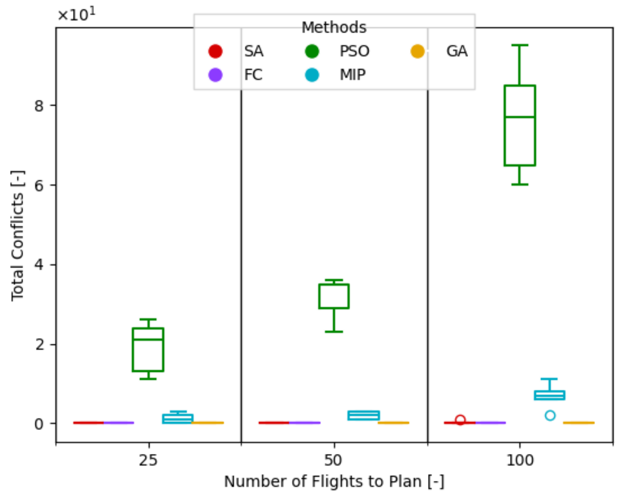

Figure 5. Conflicts for a QL of 25

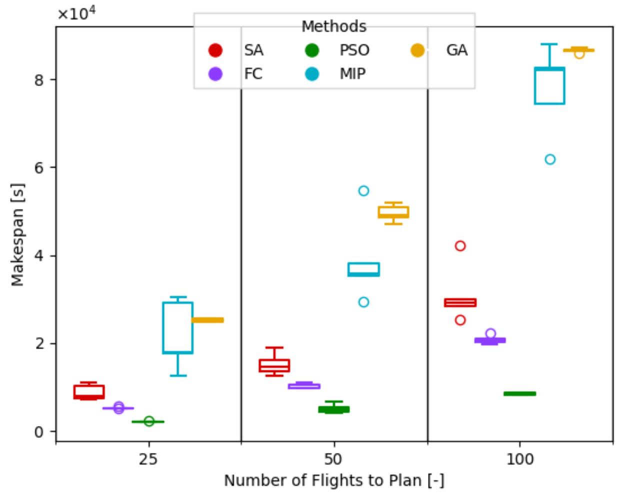

Figure 6. Makespans for a QL of 25

Out of the methods with few to no conflicts, Simulated Annealing and the First-Come First-Served algorithm seem to perform best for minimising the makespan (or the time from first takeoff to last landing). The influence of the Queue Length is unclear from these plots - some results see improvement and others show a decrease in performance, thus another visualisation method is necessary.

B. *Varying the Queue Length: Single Pad*

It is seen in the previous analysis that increasing the sequencing queue length does not always yield better results. Therefore, it is useful to visualise the Makespans and Total Conflicts as a function of this. Plotting the outputs with the Queue Length varied yields Figure 7, where the makespans outputs are plotted for Queue Lengths of 5, 10 and 25.

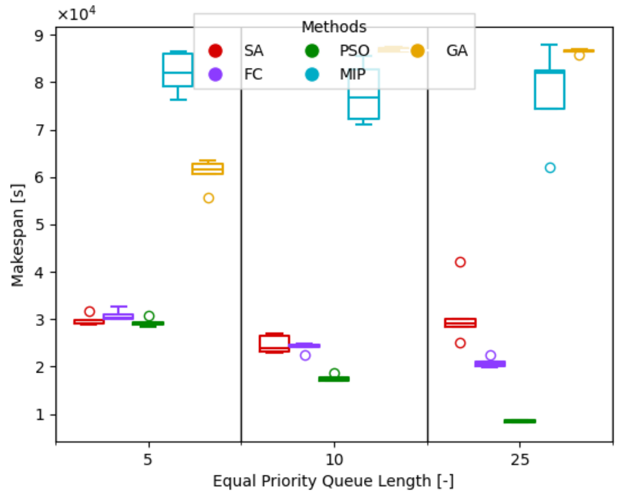

Figure 7. Makespans at varied Queue Length, for $n_{drones} = 100$. PSO performs best, followed by FC and SA

Here it is seen that while avoiding conflicts, the Genetic Algorithm is unable to ensure better packing (and therefore shorter makespans). Particle Swarm optimisation seems to outperform Simulated Annealing slightly in terms of makespan. However, the PSO implementation proves to be ineffective at scheduling conflict prevention for a queue length of 25, as seen in Figure 8

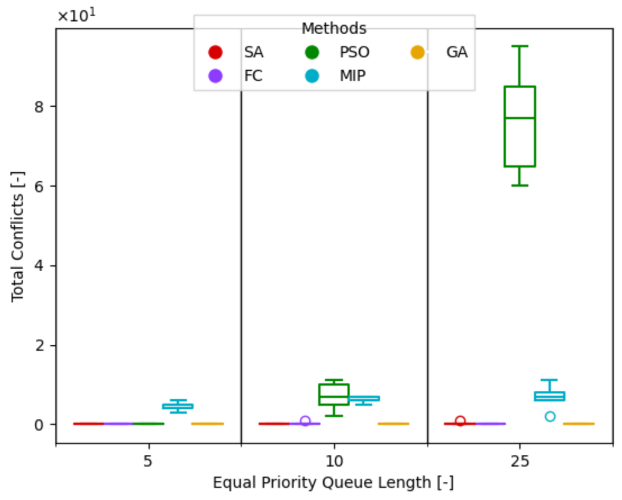

Figure 8. Total Conflicts for different Queue Lengths: SA, FC and GA perform well

C. *Varying Number of Drones: Multi-Pad*

For the multi-pad scenario, the number of drones is varied at QL=25. The results are seen in Figure 9.

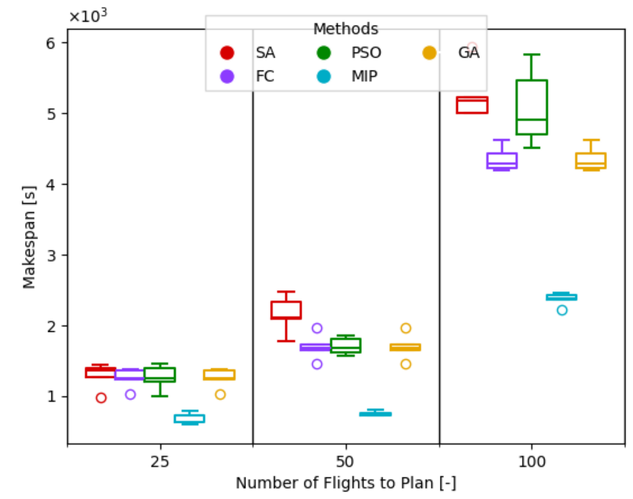

Figure 9. Makespans for a QL of 25 (multi)

Notably, for all optimizers, the multi-pad implementation yields zero scheduling conflicts.

D. *Varying the Queue Lengths: Multi-Pad*

The same analysis for varied queue length is performed for the multipad setup. This yields Figure 10.

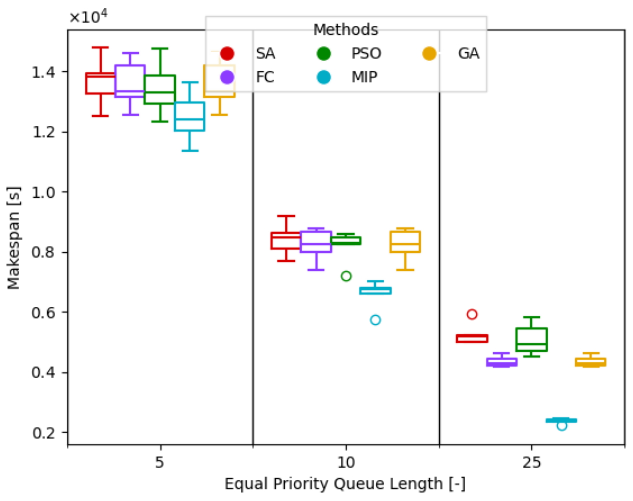

Figure 10. Makespans at varied Queue Length (multi), for $n_{drones} = 100$. MIP performs well

Notably MIP now performs best in terms of makespan, or total deployment time. GA and FC converge to identical solutions.

E. *Computation Times*

The time to compute the takeoff sequences are an important consideration when implementation in a real-world scenario is concerned. In terms of these, the single and multipad scenarios for 100-drone queues are given. These are seen in Figure 11 and Figure 12.

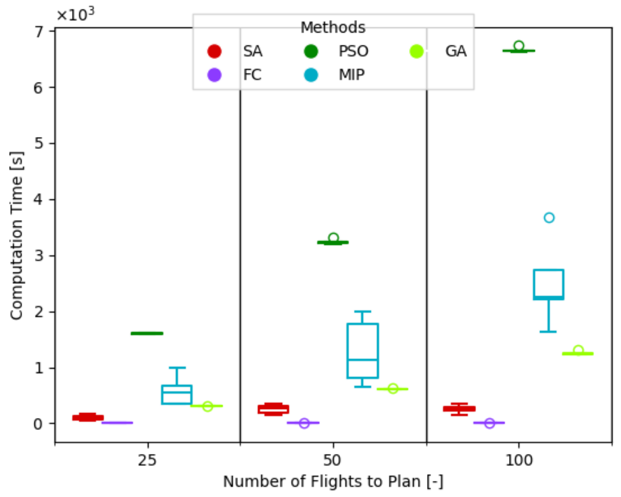

Figure 11. Computation Times (single), QL=25: ES, FC and SA perform well

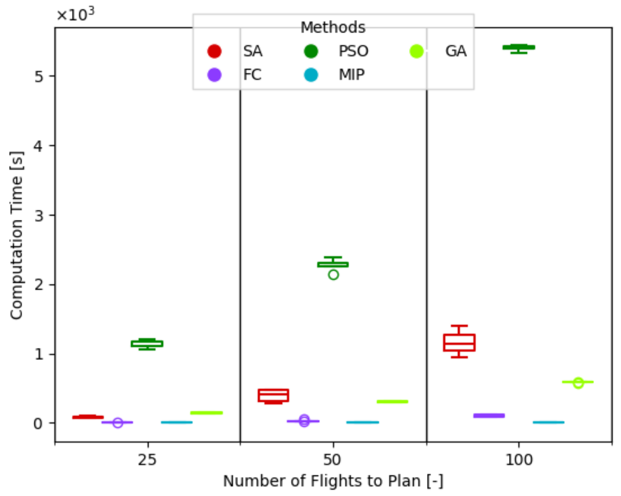

Figure 12. Computation Times (multi), QL=25: PSO performs worst with MIP being fastest

The optimisations were CPU-bound, and a Ryzen 3 3100 with 16 GB of RAM was used. For the single-pad scenario, the MIP and PSO perform the worst, with SA showing the lowest computation time of the metaheuristic methods. For the multi-pad layout, PSO takes the longest for the computation, with MIP and FC being the fastest.

F. Discussion

From these experiments, it has been shown that multiple methods can be used to plan delivery traffic for urban drone parcel delivery. Despite extensive attempts to improve the fitness function formulation or the hyper-parameters for the Genetic Algorithm, the result could not be improved further for the Single-Pad case, and while free of conflicts, the schedule is still far from optimal. Simulated Annealing and Particle Swarm Optimisation show similar results for the makespans (time from first takeoff to last landing), however only simulated annealing yields schedules which are free of conflicts.

Compared to the First-Come First-Served Delay method, Simulated Annealing performs adequately at a queue length of 10. However, for a single-pad problem, the results show a clear advantage in favour of the First-Come First-Served Delay method. This is because the dimensionality of the problem (and optimisation time) grow as we increase the sequencing queue length. The multi-pad layout however sees an improvement over First-Come First-Served when the MIP is used, especially for longer flight queues. This means that for a constrained single-pad case, sequencing shows no improvement, but that for a multi-pad

A point of importance is that this work investigates scheduling methods specifically without taking the impact on the other layers into account. Further work will be done to quantify the interactions between this strategic layer and the Tactical and Collision avoidance layers, in order to produce an uncertainty-robust system.

In the multi-pad scenario, MIP works fastest with less constraints and sufficient resources. Interestingly, Simulated Annealing is also bested by the Genetic Algorithm.

V. Conclusion and Recommendations

The objective of the study was to investigate which strategic planning methods could help drone delivery in an urban airspace through conflict-free departure planning. It is concluded that:

- For single-pad operations, sequencing provides no significant advantage over first-come first-served approaches. This is due to the takeoff-landing coupling, and a difficulty in removing mixed (takeoff-landing) conflicts.
- For multi-pad operations, the best sequences are provided by the MIP approach, which also shows a shorter computation time.
- Simulated annealing performs better than the rest of the meta-heuristic methods in terms of total delivery time/makespan, while maintaining little to no scheduling conflicts.
- Out of the meta-heuristic algorithms, Simulated Annealing is the simplest to implement, needing the least hyper-parameter choices. This means that there is less tuning required, with only the objective function remaining a factor. Flexibility can be achieved as well, by instance by taking the First-Come First-Served solutions as input, since they use the same conflict detection algorithm.

It is recommended that:

- Further research be performed in determining the optimal pad configurations for drone delivery
- The in-flight conflicts and traffic from other parcel centers need to be a focal point for further work
- When space permits, separate takeoff and landing pads should be used.
- A dynamic re-scheduling tool is needed in case uncertainties arise and the airspace complexity grows. Such a tool could also reconfigure airspace when needed.

This paper provides some insights into what scheduling methods can be used in UTM applications, more specifically drone delivery. Further research is needed to investigate how scheduling can be dynamically altered in order to deal with uncertainties arising in flight such that the schedule remains robust.

References

[1] SESAR Joint Undertaking, "European drones outlook study: unlocking the value for europe.," 2017.

[2] Amazon, Inc., "Amazon Prime Air Website,"

[3] Single European Sky ATM Research 3 Joint Undertaking, *U-space – Blueprint*. Publications Office, 2017.

[4] Sunil, Emmanuel; Hoekstra, Jacco; Ellerbroek, Joost; Bussink, F; Vidosavljevic, A; Nieuwenhuisen, D, "The influence of traffic structure on airspace capacity," *7th International Conference on Research in Air Transportation*.

[5] M. Ribeiro, J. Hoekstra, J. Ellerbroek, "Determining Optimal Conflict Avoidance Manoeuvres At High Densities With Reinforcement Learning," 2020.

[6] F. Schimpf, S. Notter, A. Ahmad, and W. Fichter, "Attention-based spatial encoding for multi agent coordination," 01 2023.

[7] J. M. Hoekstra and J. Ellerbroek, "Bluesky ATC simulator project: an open data and open source approach," in *Proceedings of the 7th international conference on research in air transportation*, vol. 131, p. 132, FAA/Eurocontrol USA/Europe, 2016.

[8] J. H. Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

[9] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks (ICNN)*, vol. 4, (Perth, Australia), pp. 1942–1948, IEEE, 1995.

[10] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[11] D. J. Wales and J. P. Doye, "Global optimization by basin-hopping and the lowest eigenmode of the hamiltonian," *Physical Review Letters*, vol. 78, no. 9, pp. 1791–1794, 1997.

[12] OpenStreetMap contributors, "Planet dump retrieved from https://planet.osm.org ." https://www.openstreetmap.org, 2017.

[13] OpenStreetMap Wiki, "Overpass API — OpenStreetMap Wiki," 2023. [Online; accessed 2-November-2023].

[14] L. Bajzikova, S. bernard, D. Bouvier, H. Drevillon, A. Hourclats, M. Carrazs, "Military and U-Space: Guidelines (D1 U-Space evaluation)," p. 174, 5 2023.

[15] Metropolis 2 Consortium, "Metropolis 2 final project results report," Multidisciplinary Digital Publishing Institute, 2022.

[16] D. Rein-Weston, "Four-Dimensional Trajectory Planning in Air Traffic Management: Feasibility of a Heuristic Branching Method,"