Scheduling Conflict-free Urban Air Mobility Operations with Answer Set Programming

Gioacchino Sterlicchio[a,*], Angelo Oddi[b], Riccardo Rasconi[b] and Francesca Alessandra Lisi[c]

[a]DMMM, Polytechnic University of Bari, Italy
[b]Institute of Cognitive Sciences and Technologies, National Research Council, Italy
[c]DIB and CILA, University of Bari Aldo Moro, Italy
ORCID (Gioacchino Sterlicchio): https://orcid.org/0000-0002-2936-0777, ORCID (Angelo Oddi): https://orcid.org/0000-0003-4370-7156, ORCID (Riccardo Rasconi): https://orcid.org/0000-0003-2420-4713, ORCID (Francesca Alessandra Lisi): https://orcid.org/0000-0001-5414-5844

Abstract. Urban Air Mobility (UAM) presents emerging challenges in airspace coordination, particularly in densely populated metropolitan areas where the proliferation of drones, air taxis, and other aerial vehicles increases the risk of mid-air conflicts. This work introduces a declarative approach to strategic deconfliction using Answer Set Programming (ASP), enabling time-synchronized and optimized flight routing to ensure conflict-free operations. We benchmark the ASP solution against a Constraint Programming (CP) model, evaluating both approaches through scalability tests and resource (time and memory) usage analysis. Results indicate that ASP consistently outperforms CP in execution time and scalability for small to medium-sized problem instances, while CP demonstrates more stable memory consumption but suffers from significant performance degradation as problem complexity increases. These findings highlight the potential of ASP for efficient and scalable deconfliction in future UAM systems.

1 Introduction

Urban Air Mobility (UAM) is transforming the way people and goods move within cities, offering a promising solution to alleviate ground traffic congestion, reduce emissions, and increase transportation efficiency [17]. The UAM ecosystem encompasses a wide range of air vehicles, including electric vertical takeoff and landing (eVTOL) aircraft, drones, air taxis, and helicopters, which are expected to operate in densely populated metropolitan areas. However, the integration of these new air vehicles into the existing airspace poses significant challenges, particularly in terms of safety, efficiency, and scalability [30]. To meet this challenge, industry actors and regulators are adjusting UAM Concept of Operations (ConOps) by expanding helicopter corridors into specific air lanes for eVTOLs [3] bypassing traditional Air Traffic Controller (ATC). For UAM, the US Federal Aviation Administration (FAA) and National Aeronautics and Space Administration (NASA) have proposed concepts for Unmanned Aircraft System Traffic Management (UTM) in recent years [13, 3]. One of the most critical challenges facing UAM is the need for effective conflict management, also known as deconfliction. Deconfliction refers to the process of predicting, detecting, and resolving potential conflicts between air vehicles, as well as between air vehicles and other air traffic, obstacles, or restricted airspace [30].

In this work, we build upon established frameworks to model and solve the *Strategic Deconfliction* (SD) problem in UAM. SD, the first of three layers in Air Traffic Management (ATM) [16], addresses conflict resolution prior to departure through airspace organization, demand-capacity balancing, and traffic synchronization. The second layer, *Tactical Deconfliction*, manages in-flight separation using real-time data such as position, heading, and speed – sourced from radar or telemetry. The third, *Collision Avoidance*, relies on onboard systems to prevent imminent collisions. Our focus is on the traffic synchronization aspect of SD, which involves proactive trajectory planning to reduce conflict risk and optimize airspace usage. We address this challenge using Answer Set Programming (ASP) to develop scalable, conflict-free routing strategies. ASP is well-suited for air traffic management due to its strength in solving complex combinatorial problems [15]. By encoding the problem as logical rules, it efficiently generates optimal, conflict-free flight plans that comply with safety and operational requirements. To the best of our knowledge, no existing literature applies ASP or similar logic-based declarative methods to SD in UAM. Most prior work relies on numerical techniques, as detailed in a later section. The key contributions of this paper are: (i) formalizing the SD problem, (ii) modelling the airspace topology, drone fleet, and the SD problem, and (iii) empirically evaluating the approach through scalability tests, comparing its performance with Constraint Programming (CP).

The paper is structured as follows: Section 2 reviews related work on startegic deconfliction in UAM. Section 3 introduces ASP fundamentals. Section 4 defines the SD problem with a motivating example and formalization. Section 5 details the ASP-based modelling approach. Section 6 presents the evaluation methodology and results. Finally, Section 7 concludes with key insights and future directions.

2 Related works

This section reviews related work in two areas: (i) state-of-the-art approaches to SD, and (ii) ASP-based scheduling in comparable domains, given its absence in current SD research.

Strategic deconfliction involves pre-departure decisions – such as ground delays and waypoint merging – to balance traffic de-

* Corresponding Author. Email: g.sterlicchio@phd.poliba.it

mand with airspace capacity. Sacharny and Henderson [20] propose a greedy scheduling algorithm that minimizes deviation from desired release times. Torabbeigi et al. [29] model parcel delivery using a set covering approach for strategic planning and mixed integer linear programming (MILP) for operations. The works [21, 22] introduce a lane-based navigation framework with one-way lanes and polygonal roundabouts, enabling computation of conflict-free launch intervals. Their Lane Strategic Deconfliction (LSD) method supports one-dimensional strategic deconfliction using this lane-based model. Tang et al. [26] formulate conflict-free trajectory planning as a mixed-integer Second-Order Cone program (SOCP). Bertram et al. [6] introduces FastMDP-GPU, a first-come-first-served scheduler generating conflict-free plans on demand. Sacharny et al. [25] contribute a comprehensive lane-based framework, including scheduling algorithms, performance analysis tools, and a tactical deconfliction protocol. Huang et al. [12] applie multi-agent reinforcement learning (MARL) to low-altitude conflict resolution. Thompson et al. [28] propose a rapidly-exploring random tree (RRT) and Thompson et al. [27] an A^* based algorithms for deconflicted routing. Chen et al. [7] combine demand-capacity balancing (DCB) with reinforcement learning, introducing both MILP-based and heuristic DCB methods. Liu et al. [16] model airspace using stacked hexagonal tessellation and solve SD via integer programming. Pradeep et al. [18] address SD in urban package delivery using a a mixed-integer non-linear programming (MINLP) model for scheduling under temporal constraints. Finally, Xue [30] compare centralized and decentralized architectures across all deconfliction layers, analysing their robustness to communication and state estimation errors.

ASP has been effectively applied to various scheduling problems. Ricca et al. [19] use ASP to automate team formation at the Gioia Tauro seaport. Dodaro and Maratea [8] and Alviano et al. [4] apply ASP to nurse scheduling and rescheduling, respectively, optimizing shift assignments, and minimizing disruptions. Dodaro et al. [9] addresses chemotherapy scheduling in oncology clinics. ASP has also been used for train scheduling [1, 2], combining routing and optimization via hybrid approaches with difference constraints using Clingo[DL].

3 Answer Set programming

Answer Set Programming (ASP) is a declarative programming paradigm [15] that is based on logic programming and non-monotonic reasoning, which allows for the representation of incomplete or uncertain information. A logic program consists of rules of the form: $a_1 \leftarrow a_2, \ldots, a_m, not\ a_{m+1}, \ldots, not\ a_n$. Where each a_i is an atom of form $p(t_1, \ldots, t_k)$ and all t_i are terms, composed function symbols and variables. Atom a_1 is often called head atom, while a_2 to a_m and a_{m+1} to a_n are also referred to as positive and negative body literals, respectively. As usual, not denotes negation as failure. Rules without body are called facts. The head is unconditionally true and the arrow is usually omitted. Conversely, rules without head are called integrity constraints (or denials). To ease the use of ASP in practice, several extensions have been developed. First of all, language constructs include conditional literals and cardinality constraints. The former are of the form $a : b_1, ..., b_m$, the latter can be written as $s\{d_1; ...; d_n\}t$, where a and b_i are possibly negated (regular) literals and each d_j is a conditional literal; s and t provide optional lower and upper bounds on the number of satisfied literals in the cardinality constraint. We refer to $b_1, \ldots, b_m$ as a condition. Note, more elaborate forms of aggregates are obtained by explicitly using functions (e.g., #sum) and relation symbols (e.g., <=).

Semantically, a logic program induces a set of stable models or answer sets, being distinguished models of the program determined by the stable models semantics [11]. ASP solvers (e.g. Clingo [10] and DLV [14]) use efficient algorithms based on non-monotonic reasoning and logic programming techniques to compute answer sets for given programs. These solvers typically employ grounding techniques to convert first-order logic into propositional form so that ASP solvers can be used for solving them as efficiently as possible with the current knowledge.

4 Problem statement

4.1 Lane-based airspace structure

The addressed problem follows the principle of the *lane-based* airspace structure highlighted in [25] as regard the creation and layout of the track network. In [23], the authors discuss the results of an in-depth comparison of FAA-NASA strategic deconfliction (FNSD) and Lane-based strategic deconfliction demonstrating that FNSD suffers from several types of complexity with an high computational burden in resolving flight paths which are generally absent from the lane-based method. The two major disadvantages of the lane-based system are: (i) aerial vehicle are restricted to a fixed set of lanes, and this may result in greater distance travelled, and (ii) the vehicle may require to turn more to follow lanes rather than a smooth trajectory. On the other hand, lanes allow for efficient and effective real-time deconfliction to mitigate contingencies [24]. The lane-based approach defines a set of one-way lanes where each lane is defined by an entry point, an exit point, and a one-dimensional curve between the two. The lane structure is modelled like a direct graph $G = (V, E)$, where V is the set of vertexes or lane entry-exit point and E the set of lanes. Two-way traffic between vertexes can be achieved by having air lanes next to each other at the same altitude or at different altitudes. Since aerial vehicles move in three dimensions, the lanes must form 3D corridors. Additional design constraints can be specified in terms of headway — that is, the safe separation distance between vehicles. The combination of headway requirements and corridor design supports a variety of vehicle trajectory constraints, while the directed graph structure imposed on the airspace provides agents with an organized environment for computation. Lanes may have other associated properties e.g., speed restrictions specified by the UTM. Lanes are connected so that every vertex has either in-degree or out-degree equal to one. This contrasts with both zone-based deconfliction (that presumes that vehicles can enter and exit in any direction and thus the entire zone must be reserved, which is inefficient for large areas), and with cell-based deconfliction (that combines zone reservation with general motion planning within each cell). Figure 1 shows an example of lane structure as a graph $G = (V, E)$, where $V = \{1, 2, 3, 4\}$ and $E = \{(1, 2), (2, 3), (3, 4)\}$ for the one-way structure and $E = \{(1, 2), (2, 1), (2, 3), (3, 2), (3, 4), (4, 3)\}$ for the two-way alternative. In particular, vertex 1 and 4 are ground nodes while vertex 2 and 3 are waypoints of the airspace structure at some altitude. The lanes allow only one direction of travel without overtaking.

4.2 Problem introduction

The SD problem is to produce a set of scheduled flight paths such that no two aircraft ever get closer than a specified safety distance or *headway* (h) either in time or space. Consider Figure 1. A flight must schedule its entry-exit times through a sequence of lanes, where

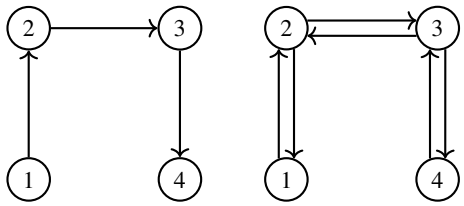


Figure 1. A simple one-way (a) and two-way (b) three-lane layout

the exit time from the previous lane equals the entry time of the following lane. In order to determine whether flights have a conflict, we use the Space-Time Lane Diagram (STLD) proposed in [25] to represent the situation graphically, as shown in Figure 2. The horizontal axis represents time, while the vertical axis represents the distance along the lane. A STDL is created for each lane. The two blue lines represent two scheduled flights named f_1 and f_2 with start times of 1 and 4 in lane 1–2 with speeds 2 and 1, respectively. The STDL shows their progress through the three lanes; it can be seen that there is always a time headway of at least 1 unit. Suppose a new flight f_3 must be scheduled, with speed 2, and the requested launch interval is $[0, 21]$. This means that the earliest launch time is 0 while the latest one is 21. The goal is to establish departure times that do not conflict with those already present and find the trajectory for all lanes that does not conflict with all flights travelling in their respective lanes. For example, consider f_3 starting at time 10 (red line); then it exits Lane 1-2 and enters Lane 2-3 at time 15; the figure shows that f_3 crosses the path of f_2 and therefore is disallowed. On the other hand, if f_3 starts at time 0, then its headway is always equal to 1 time unit from f_1, and since both flights are characterized by the same speed, they never get any closer. Moreover, for Lane 2–3 and Lane 3–4 f_3's headway is still 1 unit from f_1, so it is allowed.

4.3 Problem formalization

We formalize the SD problem as a couple (N, F) where N is the airspace network and F is the set of flights to be scheduled in N.

The network N is in its turn a tuple (V, E, Iv, Ev, l, h), where: (V, E) is a directed graph, $Iv \subset V$ is the set of ground nodes where a flight starts its trip, $Ev \subset V$ is the set of ground nodes where a flight ends its trip, $l : E \to \mathbb{N}$ assigns the lane length and $h \in \mathbb{N}$ is the headway distance (time or space) between flights.

The set F is represented by tuples $(S, L, start, end, e, l, s)$, where: (S, L) is an acyclic sub-graph of (V, E) and represents the flight route, $start : F \to Iv$ gives the ground vertex where a flight starts its trip, $end : F \to Ev$ gives the ground vertex where a flight ends its trip, $e : F \to \mathbb{N}$ gives the earliest time a flight can start its trip, $l : F \to \mathbb{N}$ gives the latest time a flight can start its trip and $s : F \to \mathbb{N}$ is the speed associated to a flight.

For our example, in Figure 1 (a) and Figure 2, the SD problem is defined for N as $V = \{1, 2, 3, 4\}$, $E = \{(1, 2), (2, 3), (3, 4)\}$, $Iv = \{1\}$, $Ev = \{4\}$, $l(1, 2) = 10$, $l(2, 3) = 10$ and $l(3, 4) = 10$ and $h = 1$. Suppose that f_1 and f_2 have been scheduled, the new flight $F = \{f_3\}$, to be scheduled, is defined as $S = \{1, 2, 3, 4\}$, $L = \{(1, 2), (2, 3), (3, 4)\}$, $start(f_3) = 1$, $end(f_3) = 4$, $e(f_3) = 0$, $l(f_3) = 21$ and $s(f_3) = 2$.

A solution to the SD problem (N, F) is represented by the pair (R, A), where: (i) R is a function that assigns to each flight a specific route in the network, and (ii) A is an assignment of arrival times to each flight at each node along their path, such that flights are pairwise deconflicted. A route is a sequence of nodes, pairwise connected by lanes. We write $v \in r$ and $(v, v') \in r$ to denote that node v or lane (v, v') are contained in the route $r = (v_1, \ldots, v_n)$ that is, whenever $v = v_i$ for some $1 \leq i \leq n$ or this additionally $v' = v_{i+1}$, respectively. A route $R(f) = (v_1, \ldots, v_n)$ for $f = (S, L, start, end, e, l, s) \in F$ has to satisfy:

$$v_i \in S \; \forall \, i, 1 \leq i \leq n \tag{1}$$

$$(v_j, v_{j+1}) \in L \; \forall \, j, 1 \leq j \leq n-1 \tag{2}$$

$$start(f) = v_1 \wedge end(f) = v_n \tag{3}$$

Conditions 1 and 2 enforce routes to be connected and feasible for the flight in question and Condition 3 ensures that each route is between a possible start and end node.

An assignment A is a function $F \times V \to \mathbb{N}$, where $A(f, v)$ is undefined whenever $v \notin R(f)$. The function A assigns the arrival time of a flight f to a node v. Given a route function R and $h \in \mathbb{N}$, an assignment A has to satisfy the following conditions:

$$A(f, start(f)) \geq e(f) \tag{4}$$

$$A(f, start(f)) \leq l(f) \tag{5}$$

Conditions 4 and 5 ensure that a flight starts its trip at the required departure time interval.

Next, for all $f_i, f_j \in F$ such that $start(f_i) = start(f_j)$:

$$A(f_i, start(f_i)) \neq A(f_j, start(f_j)) \tag{6}$$

$$|A(f_i, start(f_i)) - A(f_j, start(f_j))| \geq h \tag{7}$$

Condition 6 ensures that flights departure times are pairwise different when flights share the same starting node. Finally, Condition 7 ensures a safety distance between two flights that begin their trip from the same node.

For all $f = (S, L, start, end, e, l, s) \in F$, $R(f) = (v_1, \ldots, v_n)$ such that $1 \leq k \leq n$ and $h \in \mathbb{N}$

$$|A(f_i, v) - A(f_j, v)| \geq h \tag{8}$$

$$(A(f_i, v_k) \leq A(f_j, v_k)) \wedge (A(f_i, v_{k+1}) \leq A(f_j, v_{k+1})) \tag{9}$$

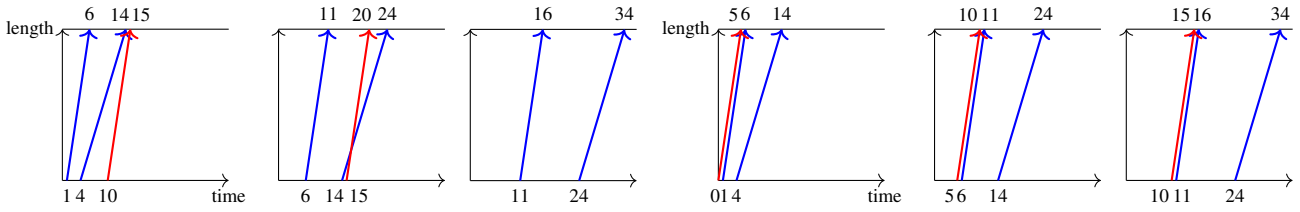
Condition 8 guarantees safe distance between flights at the same node and Condition 9 resolves the conflict between two flights that share the same lane. The solution of the previous example, shown in Figure 2, is $P(f_3) = (1, 2, 3, 4)$ and $A(f_3, 1) = 0, A(f_3, 2) = 5, A(f_3, 3) = 10, A(f_3, 4) = 15$.

Strategic deconfliction involves pre-flight planning to prevent aircraft conflicts while optimizing efficiency, safety, and operational performance. To determine the quality of a solution, we have focused our attention on minimizing total delay with respect to the required earliest launch time. We chose this metric because it is the most widely used in the literature. Nothing prevents using a different metric or more than one to determine the best plan by assigning an evaluation priority. The quality of a solution (P, A) is determined via the following condition:

$$minimize \sum_{f_i \in F} A(f_i, start(f_i)) - e(f_i) \tag{10}$$

5 Solving real-world SD problems with ASP

In this section, we present our ASP-based approach to solving a couple of variants of the SD problem (the full encoding is also included in the supplementary material). We first show how to represent flight data, followed by the actual encoding of the problem.



(a) lane 1–2 (b) lane 2–3 (c) lane 3–4 (d) lane 1–2 (e) lane 2–3 (f) lane 3–4

Figure 2. STDL representation of the flights. Conflict case (a–c), no conflict case (d–f)

5.1 *Data encoding*

For a given SD problem (N, F), the airspace network $N = (V, E, Iv, Ev, l, h)$ is represented by the facts

```
node(v). edge(v,v'). startn(iv). endn(ev).
length((v,v'),l). headway(h).
```

for each `v` $\in V$, `v,v'` $\in E$, `iv` $\in Iv$, `ev` $\in Ev$, `l` $\in \mathbb{N}$ and `h` $\in \mathbb{N}$.

The set of flights F where each $f = (S, L, start, end, e, l, s) \in F$ is defined by

```
flight(f). lane(f,v,v'). start(f,iv). end(f,ev).
speed(f,s). requested(f,e,l).
```

with `flight(f)` the flight identification and for each `v` $\in S$, `v,v'` $\in L$, $start(f) =$ `iv`, $end(f) =$ `ev`, $e(f) =$ `e`, $l(f) =$ `l` and $s(f) =$ `s`.

For example, the following facts encode the network in Figure 1 plus the headway

```
node(1..4).
edge(1,2). edge(2,3). edge(3,4)
startn(1). endn(4).
length((1,2),10). length((2,3),10). length((3,4),10).
headway(1).
```

while the set $F = \{f_1, f_2, f_3\}$ of flights to be scheduled, with the respective requested launch time interval, is given by

```
flight(f1). speed(f1,2).
lane(f1,1,2). lane(f1,2,3). lane(f1,3,4).
start(f1,1). end(f1,4).
requested(f1,1,5).

flight(f2). speed(f2,3).
lane(f2,1,2). lane(f2,2,3). lane(f2,3,4).
start(f2,1). end(f2,4).
requested(f2,2,4).

flight(f3). speed(f3,2).
lane(f3,1,2). lane(f3,2,3). lane(f3,3,4).
start(f3,1). end(f3,4).
requested(f3,3,4).
```

5.2 *Problem encoding*

In the following we describe the general problem encoding of Listing 1. We encode one feasible plan as an answer set. Line 1 defines the headway with the predicate `headway(h)` where `h` is a constant taken as input. In Line 3, for each flight `F` is assigned a starting time point `stpoint(F,X,T)` from its requested launch time interval `requested(F,E,L)` at the starting node `X` with `start(F,X)`. This rule ensures that Conditions 4 and 5 are met. Line 6 is a denial and eliminates the answer sets where 1) at least two flights share the same starting time point for the same starting node, and 2) there is no safe distance between flights at the same starting point. This denial implements Conditions 6 and 7 in the opposite way, or we look for plans that do not meet the denial. Rule at Lines 9–10 compute the estimated time of arrival. The predicate `eta(F,Y,T)` (Line 7) is true if there is a flight `F` that starts its trip at time `Ti` at node `X`, `F` travel through the lane `(X,Y)` with speed `S`. The arrival time is calculated taking into account the lane length `length((X,Y),D)` applying the formula $T = Ti + (D/S)$. Lines 11–12 is applied to all other nodes knowing the arrival time at the previous one. Lines 14–15 guarantees a safe distance between flights at the same node and Lines 17–22 resolve the conflict between two flights that share the same lane, thus implementing Condition 8 and Condition 9 respectively.

Listing 1. ASP-based encoding of the SD problem with constant speed.

```
1  headway(h).
2
3  1{stpoint(F,X,T) : T=E..L}1 :- flight(F),
4    requested(F,E,L), start(F,X).
5
6  :- headway(H), stpoint(F1,X,T1), stpoint(F2,X,T2),
7     F1!=F2, |T1-T2|<H.
8
9  eta(F,Y,T) :- stpoint(F,X,Ti), speed(F,S),
10   lane(F,X,Y), length((X,Y),D), T=(Ti+(D/S)).
11 eta(F,Y,T) :- eta(F,X,Ti), speed(F,S), lane(F,X,Y),
12   length((X,Y),D), T=(Ti+(D/S)).
13
14 :- headway(H), eta(F1,X,T1), eta(F2,X,T2),
15    F1!=F2, |T1-T2|<H.
16
17 :- eta(F1,X,Tx1), eta(F1,Y,Ty1), eta(F2,X,Tx2),
18    eta(F2,Y,Ty2), lane(F1,X,Y),
19    lane(F2,X,Y), F1!=F2, Tx1<Tx2, Ty2<Ty1.
20 :- stpoint(F1,X,Tx1), stpoint(F2,X,Tx2),
21    eta(F1,Y,Ty1), eta(F2,Y,Ty2), lane(F1,X,Y),
22    lane(F2,X,Y), F1!=F2, Tx1<Tx2, Ty2<Ty1.
```

Encoding of Listing 1 assumes constant speed of all flights through all lanes. For example, to avoid potential collisions with other aircraft or obstacles, speed adjustments are made to maintain safe distances or to avoid severe turbulence, speed is adjusted to reduce the impact of weather disturbances. To manage this kind of situation it is necessary to make a simple change for the predicate `speed(f,s)` into `speed(f,s,(x,y))`, where `f` flies at speed `s` through the lane `(x,y)`. At this point, Listing 1 subsumes a small variation for the rules that are necessary for estimated time of arrival. Listing 2 shows what is necessary to manage variable speed. In particular, the change is made only for for the predicate `speed(F,S,(X,Y))`.

Listing 2. ASP-based encoding of the SD problem with variable speed.

```
9  eta(F,Y,T) :- stpoint(F,X,Ti), speed(F,S,(X,Y)),
10   lane(F,X,Y), length((X,Y),D), T=(Ti+(D/S)).
11 eta(F,Y,T) :- eta(F,X,Ti), speed(F,S,(X,Y)),
12   lane(F,X,Y), length((X,Y),D), T=(Ti+(D/S)).
```

A simple ASP solution, with minimal headway $h = 1$, is given in the following Listing 3. There are a total of five scheduling plans, one for each answer set. We only show the first and the last because of space limitations. There is information on the route of the flights f_1, f_2, f_3 with the predicate `lane` and on the safe arrival time at each node with the `stpoint` and `eta` predicates.

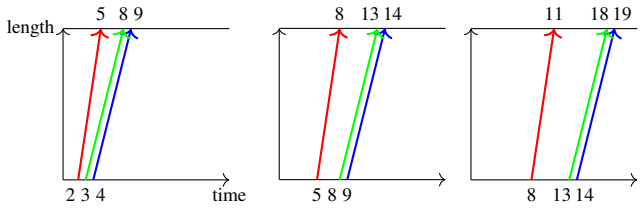
Listing 3. Possible solutions of the SD problem

```
Answer: 1
lane(f1,1,2) lane(f1,2,3) lane(f1,3,4)
stpoint(f1,1,5) eta(f1,2,10) eta(f1,3,15) eta(f1,4,20)

lane(f2,1,2) lane(f2,2,3) lane(f2,3,4)
stpoint(f2,1,2) eta(f2,2,5) eta(f2,3,8) eta(f2,4,11)

lane(f3,1,2) lane(f3,2,3) lane(f3,3,4)
stpoint(f3,1,4) eta(f3,2,9) eta(f3,3,14) eta(f3,4,19)
...

Answer: 5
lane(f1,1,2) lane(f1,2,3) lane(f1,3,4)
stpoint(f1,1,4) eta(f1,2,9) eta(f1,3,14) eta(f1,4,19)

lane(f2,1,2) lane(f2,2,3) lane(f2,3,4)
stpoint(f2,1,2) eta(f2,2,5) eta(f2,3,8) eta(f2,4,11)

lane(f3,1,2) lane(f3,2,3) lane(f3,3,4)
stpoint(f3,1,3) eta(f3,2,8) eta(f3,3,13) eta(f3,4,18)
```

Listing 4 explains the delay optimization process. The rule in Line 1 computes the delay `delay(F,D)` of a flight `F` as the absolute difference between the actual launch time `T` and the earliest requested launch time `E`. Line 2 is a sum of all the delays in each flight. Finally, Line 4 gives the best scheduled plan based on the minimal delay of the entire set of flights. Thus, the best scheduled plan of Listing 3 is `Answer 5` because `sum_delay(3)` is the smallest value among all the answer sets. The graphical representation of the solution is shown in Figure 3.

Listing 4. Delay optimization

```
1 delay(F,D) :- flight(F), stpoint(F,X,T),
2   requested(F,E,L), D=T-E.
3 sum_delay(Sd) :- Sd = #sum{D : flight(F),delay(F,D)}.
4
5 #minimize {Sd : sum_delay(Sd)}.
```



(a) lane 1–2 (b) lane 2–3 (c) lane 3–4

Figure 3. STDL representation of answer 5. `f1` is highlighted in blue, `f2` in red and `f3` in green

6 Evaluation

This section presents the experimental results obtained for the three presented use cases under three UAM network topologies. The SD problem instances are solved with the ASP encoding presented in Section 5 and also with the Constraint Programming (CP) approach (available only in the Supplementary Material[1] for reasons of space) for comparison purposes. We first present the experimental design, that is, the test instances and the experimental setup, including the software used and the parameters chosen.

6.1 Test instances

We consider three UAM network topologies, illustrated in Figure 4. These represent different types of layout that might be encountered in future UAM cases [5]:

- *Intracity/sub-urban*, focuses on air-based transportation within a city's core, addressing urban congestion and enabling rapid point-to-point mobility. Connecting suburbs to urban cores or other suburbs, extending transit reach and reducing reliance on highways.
- *Intercity*, focuses on air-based transportation between cities or major regional hubs, filling gaps between traditional ground transit and commercial aviation.
- *Airport shuttle*, transporting passengers between airports and urban centers, nearby cities, or transit hubs. This use case targets time-sensitive travellers seeking to bypass ground congestion and streamline airport access.

These topologies are modelled as directed graphs, where ground nodes (vertiport or vertistop) are coloured in red, while blue nodes represent waypoints up in the air. Figure 4 (a) represents a single city with multiple stops that connects the city center with suburban areas like a "subway in the sky". Figure 4 (b) represents different cities that are connected, and Figure 4 (c) represents an airport (node 1) that serves multiple cities.

6.2 Experimental setup

All experiments were run on a laptop computer with Ubuntu 20.04.4, AMD Ryzen 5 3500U @ 2.10 GHz and 8GB RAM. The Clingo ASP solver and Minizinc for CP (Gecode solver) were used with default solving parameters only the timeout has been set to 900 seconds (15 minutes). The evaluation focuses on efficiency (time and memory requirements), as well as effectiveness (ability to find a suitable plan to schedule flights). Specifically, we define the following research question (RQs). **RQ1)** How scalable is ASP by varying the number of flights? **RQ2)** How efficient is the ASP approach compared to the CP approach?

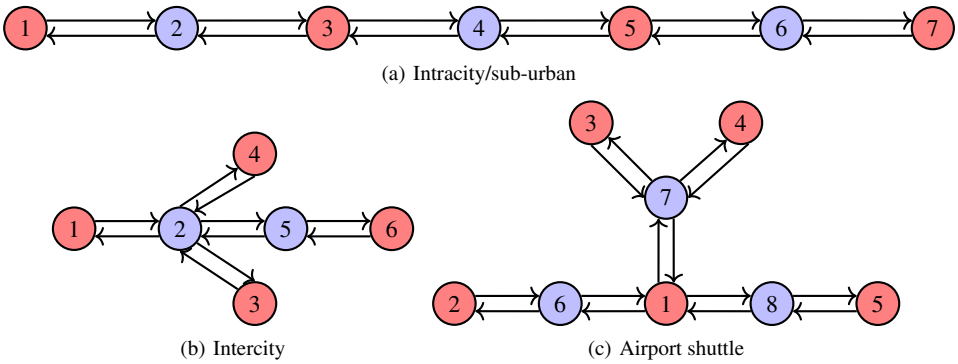
To answer the RQs, we evaluated the encodings for the three air network topologies. In the absence of public datasets used as benchmarks, we were forced to generate data for the three layouts (available in the Supplementary Material). We implemented an ASP-based data generator for this purpose. The launch interval requested for each flight has been chosen to follow a uniform distribution given a set of parameters such as the time horizon and min-max launch interval size. The Supplementary Material explains how the data were generated. We assume that the flight departure occurs at the beginning of the minute. We consider vehicle speed expressed in m/s, time and headway in minutes and lane length in m.

6.3 Results and analysis

In the following section, we discuss the experimental results of each RQ. We evaluate and discuss the proposed approach, varying the different parameters of the problem to understand the time and space requirements. The evaluation of our approach is based on analysing its efficiency through scalability tests by increasing the number of flights to be scheduled. These two dimensions are also addressed by comparing ASP with CP in order to understand the advantages and disadvantages of the two approaches.

Figure 5 shows the results obtained by varying the number of flights (tabular results can be found in the Supplementary Material). For this evaluation, the number of flights was increased from 15 to 400 over a 6-hour time horizon. The minimum headway is 1 minute with a required launch interval size ranging between 5 and 15 minutes. The Figure shows the time and memory requirements for the two approaches, ASP and CP, varying the number of flights

[1] https://figshare.com/s/8bc4a9a1eb0f2513ed56



(a) Intracity/sub-urban

(b) Intercity

(c) Airport shuttle

Figure 4. Air network topologies

in three scenarios: airport shuttle, intercity and intracity/sub-urban. When the line in the graph drops dramatically, it means that an out-of-memory occurred during the computation. ASP generally demonstrates greater efficiency in terms of execution time and memory consumption up to a certain number of flights. However, it struggles with very high numbers of flights (from 300 on), often exceeding the time limit or running out of memory. Conversely, CP tends to require more time and less memory compared to ASP and faces significant challenges with small numbers of flights (from 75 in the intercity scenario), frequently exceeding the time limit or running out of memory earlier than ASP. Overall, ASP appears to be more scalable and efficient in time while CP exhibits a more consistent memory usage but struggles significantly with the execution time as the problem complexity increases.

ASP is generally more efficient in terms of execution time for smaller problem sizes. However, as the problem size increases, ASP struggles to find the optimal solution within the memory limit, especially for higher numbers of flights. CP tends to require significantly more time to find the optimal solution compared to ASP, even for smaller problem sizes. CP consistently hits the time limit for a larger number of flights. Memory consumption remains relatively stable across different problem sizes. CP manages memory usage better than ASP, even though it struggles with execution time.

7 Conclusions

To the best of our knowledge, this is the first ASP-based formulation for Strategic Deconfliction (SD) in UAM. We define a set of conditions to ensure separation, and use ASP to model the airspace topology, drone fleet, and the SD problem itself. ASP offers key advantages: ease of modeling, support for complex combinatorial optimization, and adaptability to changing requirements. Our model supports realistic UAM scenarios and provides insight into the strengths and limitations of ASP and CP. ASP shows superior execution time for small to medium problem sizes but struggles with memory usage as complexity increases. In contrast, CP maintains stable memory consumption but often fails to meet time constraints, even on smaller instances. This comparison highlights the trade-off between time efficiency and memory stability in logic and constraint based approaches to SD.

Further evaluation of the proposed approach is still required. In particular, scalability with respect to increasingly complex scenarios – characterized by larger and denser graph topologies in terms of nodes and edges – remains an open dimension yet to be addressed. Future works, to improve the limitations encountered, can go in several directions. First, we intend to investigate the development of hybrid models that combine the strengths of both ASP and CP. Second, it could be interesting to explore advanced optimization techniques to improve the scalability, particularly for larger problem sizes. Techniques such as heuristic methods, parallel processing, and memory optimization could be considered. Finally, we plan to enhance the underlying encodings of both ASP and CP to reduce execution time and memory consumption. This could involve refining the search strategies.

References

[1] D. Abels, J. Jordi, M. Ostrowski, T. Schaub, A. Toletti, and P. Wanko. Train scheduling with hybrid ASP. In M. Balduccini, Y. Lierler, and S. Woltran, editors, *Logic Programming and Nonmonotonic Reasoning - 15th International Conference, LPNMR 2019, Philadelphia, PA, USA, June 3-7, 2019, Proceedings*, volume 11481 of *Lecture Notes in Computer Science*, pages 3–17. Springer, 2019. doi: 10.1007/978-3-030-20528-7_1. URL https://doi.org/10.1007/978-3-030-20528-7_1.

[2] D. Abels, J. Jordi, M. Ostrowski, T. Schaub, A. Toletti, and P. Wanko. Train scheduling with hybrid answer set programming. *Theory Pract. Log. Program.*, 21(3):317–347, 2021. doi: 10.1017/S1471068420000046. URL https://doi.org/10.1017/S1471068420000046.

[3] F. A. Administration. Urban air mobility concepts of operations (v2.0). Technical report, April 2023.

[4] M. Alviano, C. Dodaro, and M. Maratea. Nurse (re)scheduling via answer set programming. *Intelligenza Artificiale*, 12(2):109–124, 2018. doi: 10.3233/IA-170030. URL https://doi.org/10.3233/IA-170030.

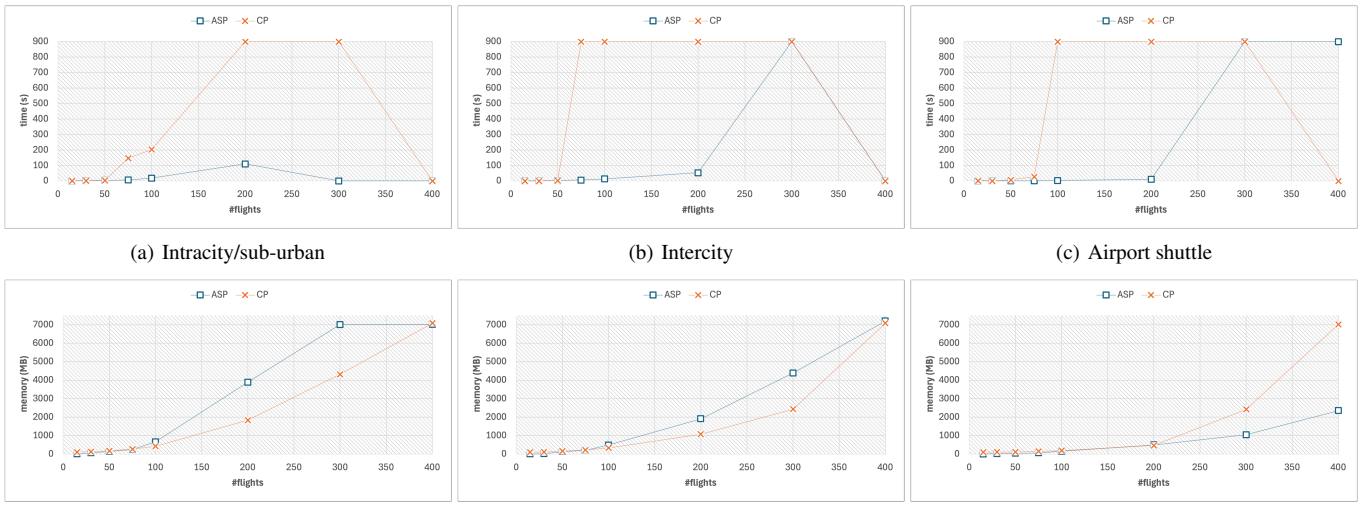
[5] L. Asmer, H. Pak, P. S. Prakasha, B. I. Schuchardt, P. Weiand, F. Meller, C. Torens, D. Becker, C. Zhu, K. Schweiger, et al. Urban air mobility use cases, missions and technology scenarios for the horizonuam project. In *AIAA Aviation 2021 Forum*, page 3198, 2021.

[6] J. R. Bertram, P. Wei, and J. Zambreno. Scalable FastMDP for pre-departure airspace reservation and strategic de-conflict. *CoRR*, abs/2008.03518, 2020. URL https://arxiv.org/abs/2008.03518.

[7] S. Chen, A. Evans, M. Brittain, and P. Wei. Integrated conflict management for UAM with strategic demand capacity balancing and learning-based tactical deconfliction. *IEEE Trans. Intell. Transp. Syst.*, 25(8): 10049–10061, 2024. doi: 10.1109/TITS.2024.3351049. URL https://doi.org/10.1109/TITS.2024.3351049.

[8] C. Dodaro and M. Maratea. Nurse scheduling via answer set programming. In M. Balduccini and T. Janhunen, editors, *Logic Programming and Nonmonotonic Reasoning - 14th International Conference, LPNMR 2017, Espoo, Finland, July 3-6, 2017, Proceedings*, volume 10377 of *Lecture Notes in Computer Science*, pages 301–307. Springer, 2017. doi: 10.1007/978-3-319-61660-5_27. URL https://doi.org/10.1007/978-3-319-61660-5_27.

[9] C. Dodaro, G. Galatà, A. Grioni, M. Maratea, M. Mochi, and I. Porro. An asp-based solution to the chemotherapy treatment scheduling problem. *Theory Pract. Log. Program.*, 21(6):835–851, 2021. doi: 10.1017/S1471068421000363. URL https://doi.org/10.1017/S1471068421000363.



(a) Intracity/sub-urban (b) Intercity (c) Airport shuttle

(d) Intracity/sub-urban (e) Intercity (f) Airport shuttle

Figure 5. Time (a–c) and memory (d–f) requirements varying the number of flights of the ASP approach compared with the CP approach

[10] M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub. Clingo= ASP+ control: Preliminary report. *arXiv preprint arXiv:1405.3694*, 2014.

[11] M. Gelfond and V. Lifschitz. Logic programs with classical negation. In D. H. D. Warren and P. Szeredi, editors, *Logic Programming, Proceedings of the Seventh International Conference, Jerusalem, Israel, June 18-20, 1990*, pages 579–597. MIT Press, 1990.

[12] C. Huang, I. Petrunin, and A. Tsourdos. Strategic conflict management for performance-based urban air mobility operations with multi-agent reinforcement learning. In *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 442–451. IEEE, 2022.

[13] P. Kopardekar, J. Rios, T. Prevot, M. Johnson, J. Jung, and J. E. Robinson. Unmanned aircraft system traffic management (utm) concept of operations. In *AIAA AVIATION Forum and Exposition*, number ARC-E-DAA-TN32838, 2016.

[14] N. Leone, C. Allocca, M. Alviano, F. Calimeri, C. Civili, R. Costabile, A. Fiorentino, D. Fuscà, S. Germano, G. Laboccetta, B. Cuteri, M. Manna, S. Perri, K. Reale, F. Ricca, P. Veltri, and J. Zangari. Enhancing DLV for large-scale reasoning. In M. Balduccini, Y. Lierler, and S. Woltran, editors, *Logic Programming and Nonmonotonic Reasoning - 15th International Conference, LPNMR 2019, Philadelphia, PA, USA, June 3-7, 2019, Proceedings*, volume 11481 of *Lecture Notes in Computer Science*, pages 312–325. Springer, 2019. doi: 10.1007/978-3-030-20528-7_23. URL https://doi.org/10.1007/978-3-030-20528-7_23.

[15] V. Lifschitz. Answer sets and the language of answer set programming. *AI Magazine*, 37(3):7–12, 2016.

[16] Y. Liu, Z. Zhou, W. Naqvi, and J. Chen. Strategic deconfliction of unmanned aircraft based on hexagonal tessellation and integer programming. *Journal of Guidance, Control, and Dynamics*, 46(12):2362–2372, 2023.

[17] H. Pak, L. Asmer, P. Kokus, B. I. Schuchardt, A. End, F. Meller, K. Schweiger, C. Torens, C. Barzantny, D. Becker, J. M. Ernst, F. Jäger, T. Laudien, N. Naeem, A. Papenfuß, J. Pertz, P. S. Prakasha, P. Ratei, F. Reimer, P. Sieb, C. Zhu, R. Abdellaoui, R.-G. Becker, O. Bertram, A. Devta, T. Gerz, R. Jaksche, A. König, H. Lenz, I. C. Metz, F. Naser, L. Schalk, S. Schier-Morgenthal, M. Stolz, M. Swaid, A. Volkert, and K. Wendt. Can urban air mobility become reality? opportunities and challenges of uam as innovative mode of transport and dlr contribution to ongoing research. *CEAS Aeronautical Journal*, May 2024. ISSN 1869-5590. doi: 10.1007/s13272-024-00733-x. URL https://doi.org/10.1007/s13272-024-00733-x.

[18] P. Pradeep, G. S. Yarramreddy, N. Amirsoleimani, A. Munishkin, R. A. Morris, M. Xue, K. Chour, and K. Kalyanam. Rolling horizon with k-position search method for strategic deconfliction of package delivery uas. In *AIAA AVIATION FORUM AND ASCEND 2024*, page 4456, 2024.

[19] F. Ricca, G. Grasso, M. Alviano, M. Manna, V. Lio, S. Iiritano, and N. Leone. Team-building with answer set programming in the gioia-tauro seaport. *Theory Pract. Log. Program.*, 12(3):361–381, 2012. doi: 10.1017/S147106841100007X. URL https://doi.org/10.1017/S147106841100007X.

[20] D. Sacharny and T. C. Henderson. A lane-based approach for large-scale strategic conflict management for uas service suppliers. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 937–945. IEEE, 2019.

[21] D. Sacharny, T. C. Henderson, and M. Cline. An efficient strategic deconfliction algorithm for large-scale uas traffic management. *School Comput., Univ. Utah, Salt Lake City, UT, USA, Rep. UUCS-20-010*, 2020.

[22] D. Sacharny, T. C. Henderson, and M. Cline. Large-scale uas traffic management (utm) structure. In *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 7–12. IEEE, 2020.

[23] D. Sacharny, T. C. Henderson, M. Cline, B. Russon, and E. Guo. Faanasa vs. lane-based strategic deconfliction. In *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 13–18. IEEE, 2020.

[24] D. Sacharny, T. C. Henderson, and E. Guo. A DDDAS protocol for real-time large-scale uas flight coordination. In *Dynamic Data Driven Applications Systems: Third International Conference, DDDAS 2020, Boston, MA, USA, October 2-4, 2020, Proceedings 3*, pages 49–56. Springer, 2020.

[25] D. Sacharny, T. C. Henderson, and V. Marston. Lane-based large-scale UAS traffic management. *IEEE Trans. Intell. Transp. Syst.*, 23 (10):18835–18844, 2022. doi: 10.1109/TITS.2022.3160378. URL https://doi.org/10.1109/TITS.2022.3160378.

[26] H. Tang, Y. Zhang, V. Mohmoodian, and H. Charkhgard. Automated flight planning of high-density urban air mobility. *Transportation Research Part C: Emerging Technologies*, 131:103324, 2021.

[27] E. L. Thompson, Y. Xu, and P. Wei. One-shot strategically deconflicted route and operational volume generation for urban air mobility operations. In *25th IEEE International Conference on Intelligent Transportation Systems, ITSC 2022, Macau, China, October 8-12, 2022*, pages 5174–5181. IEEE, 2023. doi: 10.1109/ITSC57777.2023.10421827. URL https://doi.org/10.1109/ITSC57777.2023.10421827.

[28] E. L. Thompson, Y. Xu, and P. Wei. A framework for operational volume generation for urban air mobility strategic deconfliction. In *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 71–78. IEEE, 2023.

[29] M. Torabbeigi, G. J. Lim, and S. J. Kim. Drone delivery scheduling optimization considering payload-induced battery consumption rates. *J. Intell. Robotic Syst.*, 97(3):471–487, 2020. doi: 10.1007/S10846-019-01034-W. URL https://doi.org/10.1007/s10846-019-01034-w.

[30] M. Xue. Urban air mobility conflict resolution: Centralized or decentralized? In *AIAA aviation 2020 forum*, page 3192, 2020.