Communication Papers of the 20th Conference on Computer Science and Intelligence Systems (FedCSIS) pp. 25–32
DOI: 10.15439/2025F9355
ISSN 2300-5963 ACSIS, Vol. 45

A Constraint Programming Approach for Urban Drone Trajectory Optimization

Zahraa Asfour, Sonia Cafieri, Andrija Vidosavljevic
ENAC, Université de Toulouse
Email: {zahraa.asfour, sonia.cafieri, andrija.vidosavljevic}@enac.fr

***Abstract*—We address the optimization of drone trajectories in urban environments. We introduce a Constraint Programming formulation for a version of the problem where the vehicles are at the same flight level and their path to fly by is known. We show that, compared to an approach based on a Mixed-integer Linear Optimization model introduced previously, the Constraint Programming approach allows better performances to be achieved in the solution of the problem for a specific instance structure.**

I. Introduction

THE growing use of drone delivery in urban environments requires safe and efficient low-altitude trajectory planning and optimization [4]. We address the problem of optimizing drone trajectories by adjusting ground delays and cruise speeds for flights operating along fixed horizontal paths at a common flight level. The main constraints come from a key operational requirement, and consists in maintaining a sufficient separation between pairs of drones at each time along their trajectories. Although the general problem of optimizing drone trajectories involves deciding horizontal paths and flight-levels too for the considered vehicles, the version addressed in this paper is worth exploring as it may be exploited in decomposition approaches for solving large-scale drone trajectory optimization problems.

In [2], a mixed-integer linear optimization model is introduced for drone trajectory optimization in urban environments, where the selection of horizontal paths, flight-levels, and ground delays are used as optimization levers. The Urban Drone Trajectory Model (UDTM) introduced in [1] extends this by also incorporating cruise speed decisions. In [8], optimization is based on adjusting cruise speeds and ground delays, with additional considerations for flight priorities. This is extended in [9], where scheduled take-off uncertainty is addressed too. In that work, the problem is solved using the adversarial Benders decomposition. The subproblem, formulated as a constraint programming model, which retains the same optimization levers as in [8], is solved by three different heuristics.

In this paper, we introduce a Contraint Programming (CP) formulation for the considered problem, as an alternative to the mixed-integer linear optimization model (MILP) from our earlier work [1]. Two optimization criteria are considered. The first minimizes the total deviation, defined as the sum of differences between actual and scheduled landing times across all flights. It reflects a system-level efficiency objective. The second minimizes the maximum individual deviation, aiming to limit the worst-case deviation. This criterion is relevant when coordination between multiple agents is limited [3]. We show that the CP formulation performs better than the MILP formulation on a specific instance structure, particularly when minimizing the maximum deviation.

The remainder of the paper is organized as follows. Section II presents the problem, recalls the MILP model from earlier work, and introduces the CP model. Section III presents and discusses numerical results. Section IV concludes the paper and discusses possible directions for future research.

II. CP and MILP Formulations

This section defines the problem under consideration, recalls the MILP model previously introduced in [1], and presents the CP model proposed in this study.

A. Problem statement

We consider a set of flight intentions, each defined by a departure point, an arrival point, a scheduled take-off time, a horizontal path, and an authorized cruise speed range, bounded between a minimum and a maximum value for each segment of the path. All flights operate at a single cruising flight-level and follow fixed horizontal paths in an urban environment modeled as a directed graph, as proposed in [2], where arcs represent street segments and nodes represent their intersections. Drones are assumed to be of the same type.

This study focuses on optimizing drone trajectories while avoiding Potential Loss of Separation (PLoS) during the cruise phase. A PLoS is defined as a situation in which the time difference between the arrivals of two drones at the same node is less than the minimum separation time. In this work, the minimum separation time is considered node-specific. It is derived from the minimum required separation distance and the minimal authorized cruise speed on the arcs leading to that node.

The goal is to determine, for each drone, a ground delay before takeoff and a cruising speed for each arc of its path, such that all separation between pairs of drones during the cruise phase are satisfied.

B. Model parameters

The main parameters used in both the MILP and CP formulations are summarized below.

Flight parameters

$\mathcal{F}$	set of all flights.
T_f	scheduled take-off time of flight $f \in \mathcal{F}$.
$\delta^{\max} \in \mathbb{R}^+$	maximum ground delay before take-off.

Horizontal path parameters

k_f	horizontal path assigned to flight $f \in \mathcal{F}$.
$K = \bigcup_{f\in\mathcal{F}}\{k_f\}$	set of all horizontal paths.
f_k	flight of path $k \in K$.
$\mathcal{N}_k$	set of nodes visited along path $k \in K$.
n_k^d	departure node of path $k \in K$.
n_k^a	arrival node of path $k \in K$.
$p_{k,n}$	predecessor of node $n \in \mathcal{N}_k \setminus \{n_k^d\}$ on path $k \in K$.
$d_{k,n}^{\min}, d_{k,n}^{\max}$	minimum and maximum travel time along arc $(p_{k,n}, n)$, based on speed bounds.
Γ_k	nominal landing time for path $k \in K$.
$N = \bigcup_{k\in K}\mathcal{N}_k$	set of all nodes visited by at least one flight.

For each pair of departure and arrival nodes associated with flight $f \in \mathcal{F}$, the horizontal path k_f is either selected from a predefined set of available paths (see Section III-B1), or computed using the A* algorithm (see Section III-B2). Note that, for each flight, the nominal landing time corresponds to the landing time obtained without ground delay and assuming the maximum allowed speed on all arcs.

Potential loss of separation parameters

$\mathcal{P}$	set of PLoS.
$k_i^1, k_i^2 \in K$	the two paths involved into the PLoS $i \in \mathcal{P}$.
$f_i^1, f_i^2 \in \mathcal{F}$	the two flights involved into the PLoS $i \in \mathcal{P}$.
$n_i \in \mathcal{N}_{f_i^1} \cap \mathcal{N}_{f_i^2}$	PLoS point where the paths k_i^1 and k_i^2 intersect within the PLoS $i \in \mathcal{P}$.
$M_i^{12}, M_i^{21} \geq 0$	big-M constants used to linearize the separation constraints for PLoS $i \in \mathcal{P}$.
$s_n \geq 0$	minimum separation time at node $n \in N$.
$\mathcal{V}_n$	set of all paths involved in a PLoS at node $n \in N$.

The set $\mathcal{P}$ includes all PLoS, identified by detecting intersecting nodes between the horizontal paths of different flights. Further details on how PLoS are identified, and on how the Big-M constants are computed, can be found in [1]. These constants are computable since all flight times are bounded by the earliest and latest possible arrival times at each node.

Note that in some cases, a pair of flight path share multiple consecutive nodes, giving rise to several detected PLoS. These PLoS are referred to as *correlated PLoS*. In such situation, the order of passage must be maintained across all involved nodes to ensure separation.

Correlated PLoS parameters

$\mathcal{O}$	set of all correlated PLoS.
$i_o^* \in \mathcal{P}$	first (reference) PLoS in correlated PLoS $o \in \mathcal{O}$.
$\mathcal{L}_o \subset \mathcal{P}$	set of remaining PLoS for $o \in \mathcal{O}$.
$k_o^1, k_o^2 \in K$	the two paths involved into the correlated PLoS $o \in \mathcal{O}$.

C. MILP model

The MILP formulation introduced in our previous work [1], corresponding to the UDTM-FPFL variant with a single cruising flight level, is recalled below. Two optimization criteria are considered: (i) minimizing the sum of deviations (referred to as the *SumDev* criterion), and (ii) minimizing the maximum individual deviation (denoted as *MaxDev*). Both formulations share variables and constraints and differ in the objective function.

The decision variable of the SumDev-MILP models are the following.

$t_{k,n}, \forall k \in K, \forall n \in \mathcal{N}_K$	continuous variable representing the arrival time of flight f_k at node n.
$u_i \in \{0,1\},\ \forall i \in \mathcal{P}$	binary variable equal to 1 if f_i^1 passes before f_i^2 at node n_i, 0 otherwise.

The SumDev-MILP formulation aims at minimizing the total deviation. For each path $k \in K$, the deviation is defined as the difference between the arrival time at the final node, t_{k,n_k^a}, and its nominal landing time Γ_k. Since Γ_k is a fixed value for each flight, this objective corresponds to minimizing the sum of individual delays. The formulation is defined as follows:

$$\min \sum_{k\in K}\left(t_{k,n_k^a} - \Gamma_k\right) \qquad \text{[SumDev-MILP]}$$

$$\text{s.t.} \quad t_{k,n_k^d} \geq T_{f_k},\ \forall k \in K \tag{1}$$

$$t_{k,n_k^d} \leq T_{f_k} + \delta^{\max},\ \forall k \in K \tag{2}$$

$$t_{k,n} - t_{k,p_{k,n}} \geq d_{k,n}^{\min},\ \forall k \in K, \forall n \in \mathcal{N}_k \setminus \{n_k^d\} \tag{3}$$

$$t_{k,n} - t_{k,p_{k,n}} \leq d_{k,n}^{\max},\ \forall k \in K, \forall n \in \mathcal{N}_k \setminus \{n_k^d\} \tag{4}$$

$$t_{k_i^2,n_i} - t_{k_i^1,n_i} \geq s_{n_i} - M_i^{12}(1-u_i),\ \forall i \in \mathcal{P} \tag{5}$$

$$t_{k_i^1,n_i} - t_{k_i^2,n_i} \geq s_{n_i} - M_i^{21}u_i,\ \forall i \in \mathcal{P} \tag{6}$$

$$u_{i_o^*} = u_i,\ \forall o \in \mathcal{O}, \forall i \in \mathcal{L}_o \tag{7}$$

$$t_{k,n} \in \mathbb{R}^+,\ \forall k \in K, \forall n \in \mathcal{N}_k$$

$$u_i \in \{0,1\},\ \forall i \in \mathcal{P}.$$

Constraints (1)–(2) define the feasible time window for the start of the cruise phase, taking into account a ground delay before take-off bounded between 0 and $\delta^{\max}$. Constraints (3)-(4) make sure the travel time along each arc to remain within admissible bounds. For any node n (excluding the departure node) on a path k, the arrival time $t_{k,n}$ depends on the arrival time at the preceding node $p_{k,n}$ and the time required to traverse the arc $(p_{k,n}, n)$. The minimum and maximum travel times, denoted respectively by $d_{k,n}^{\min}$ and $d_{k,n}^{\max}$, are derived from the maximum and minimum authorized cruise speeds on that arc and arc length. Constraints (5)-(6) ensure that a minimum time separation s_{n_i} is maintained between the two flights involved in each PLoS $i \in \mathcal{P}$. These constraints model a disjunctive condition: either flight f_i^1 passes before flight f_i^2 at node n_i, or the inverse. This disjunction is handled by introducing a binary variable u_i, which determines the order of passage, and big-M contraints. Finally, Constraint (7) ensures that for each correlated PLoS, the order of passage is same across all associated nodes.

The MaxDev-MILP formulation aims at minimizing the maximum individual deviation across all flights. To this end, an additional continuous variable $\Delta = \max_{k \in K} \left(t_{k,n_k^a} - \Gamma_k\right)$ is introduced representing the maximum deviation, and it is modeled with Constraint (8). The formulation is defined as follows:

$$\begin{aligned}
\min \quad & \Delta \qquad \text{[MaxDev-MILP]} \\
\text{s.t.} \quad & \Delta \geq t_{k,n_k^a} - \Gamma_k, \quad \forall k \in K && (8) \\
& \text{Constraints (1) to (7)} \\
& t_{k,n} \in \mathbb{R}^+, \ \forall k \in K, \forall n \in \mathcal{N}_k \\
& u_i \in \{0,1\}, \ \forall i \in \mathcal{P} \\
& \Delta \in \mathbb{R}^+.
\end{aligned}$$

D. CP model

An alternative formulation of the SumDev-MILP and MaxDev-MILP is proposed using CP formalism. Unlike the MILP formulation, which combines continuous and binary variables with linear constraints, the CP model is based on interval and integer variables and supports global constraints and logical operators.

a) Decision variables:

Interval variables. One fixed-duration interval variable, denoted $task_{k,n}$, defined for each path $k \in K$ and each visited node $n \in \mathcal{N}_k$, models the occupancy of node n by flight f_k for a duration s_n, corresponding to the minimum separation time required at that location. The start of this interval, StartOf($task_{k,n}$), represents the arrival time of flight f_k at node n, and is equivalent to the continuous variable $t_{k,n}$ in the MILP formulation.

Integer variable. A single integer variable Δ is introduced to represent the maximum deviation. It is used only in the *MaxDev* version to express the upper bound over all deviations.

b) Constraints:

Each CP constraint corresponds to a MILP counterpart and captures the same temporal or logical relationship.

Precedence and speed limit constraints:

$$\text{StartOf}(task_{k,n_k^d}) \geq T_{f_k}, \ \forall k \in K \tag{9}$$

$$\text{StartOf}(task_{k,n_k^d}) \leq T_{f_k} + \delta^{\max}, \ \forall k \in K \tag{10}$$

$$\text{EndBeforeStart}(task_{k,p_{k,n}}, \ task_{k,n}, \ d_{k,n}^{\min} - s_{p_{k,n}}), \quad \forall k \in K, n \in \mathcal{N}_k \setminus \{n_k^d\} \tag{11}$$

$$\text{StartBeforeEnd}(task_{k,n}, \ task_{k,p_{k,n}}, \ s_{p_{k,n}} - d_{k,n}^{\max}), \quad \forall k \in K, n \in \mathcal{N}_k \setminus \{n_k^d\}. \tag{12}$$

Constraints (9) and (10) ensure that the cruise phase starts within an admissible time window. These constraints are equivalent to the MILP Constraints (1) and (2). Constraints (11) and (12) impose time bounds on the travel between two consecutive nodes. They rely on the following global constraints, defined for two interval variables $task1$ and $task2$:

- Constraint (11) ensures that
$$\text{StartOf}(task_{k,n}) \geq \text{EndOf}(task_{k,p_{k,n}}) - s_{p_{k,n}} + d_{k,n}^{\min}$$
- Constraint (12) ensures that
$$\text{StartOf}(task_{k,n}) \leq \text{EndOf}(task_{k,p_{k,n}}) - s_{p_{k,n}} + d_{k,n}^{\max}$$

Since EndOf($task_{k,p_{k,n}}$) = StartOf($task_{k,p_{k,n}}$) + $s_{p_{k,n}}$, Constraints (11) and (12) together imply that the time between the start of two consecutive tasks lies within $[d_{k,n}^{\min}, \ d_{k,n}^{\max}]$. They are thus equivalent to constraints (3) and (4) in the MILP formulation.

Separation constraints:

$$\text{NoOverlap}\left(\text{SequenceVar}([task_{k,n}], \ \forall k \in \mathcal{V}_n)\right), \quad \forall n \in N. \tag{13}$$

Constraint (13) uses the global constraint NoOverlap to enforce time-based separation at each PLoS point. It is applied to the sequence variable SequenceVar($[task_{k,n}] k \in \mathcal{V}_n$), constructed from the set of interval variables $\{task_{k,n}\}_{k \in \mathcal{V}_n}$, where each interval models the presence of a flight at node n for a fixed duration s_n. The SequenceVar structure allows the solver to reason globally about the ordering of intervals on a shared unary resource. Combined with NoOverlap, it ensures that no two intervals in the set overlap in time, thereby preventing simultaneous occupancy of the node. This constraint is equivalent to the disjunctive separation Constraints (5)–(6) in the MILP formulation, but provides a more compact representation.

Order preservation constraint in correlated PLoS:

$$\text{SameSequence}\Big(\text{SequenceVar}([task_{k_o^1,n_{i_o^*}}, \ task_{k_o^2,n_{i_o^*}}]), \ \text{SequenceVar}([task_{k_o^1,n_i}, \ task_{k_o^2,n_i}])\Big), \quad \forall o \in \mathcal{O}, \ \forall i \in \mathcal{L}_o. \tag{14}$$

Constraint (14) ensures that the relative order in which two flights cross a sequence of PLoS points remains consistent when these nodes are part of the same correlated PLoS. This consistency is enforced by the global constraint SameSequence, which compares two ordered lists of interval variables, each representing the presence of the same pair of flights (paths) at different nodes. The first list defines the reference order at node $n_{i_o^*}$, while the second list corresponds to any other node $n_i, \forall i \in \mathcal{L}_o$ involved in the same correlated PLoS set. The constraint guarantees that if one flight precedes the other at the reference node, the same order is maintained at all other nodes. This constraint plays the same role as MILP Constraint (7).

Maximal deviation constraints:

$$\Delta \geq \text{StartOf}(task_{k,n_k^a}) - \Gamma_k, \ \forall k \in K. \tag{15}$$

Constraint (15) defines an upper bound Δ on the individual deviations from nominal values. It plays the same role as MILP constraint (8) in the *MaxDev* objective.

c) Model formulations:

The SumDev-CP formulation which minimizes the total deviation, is given by:

$$\begin{aligned} \min \quad & \sum_{k \in K} \left(\text{StartOf}(task_{k,n_k^a}) - \Gamma_k \right) \qquad \text{[SumDev-CP]} \\ \text{s.t.} \quad & \text{Constraints (9) to (14).} \end{aligned}$$

The MaxDev-CP formulation which minimizes the maximum individual deviation, is given by:

$$\begin{aligned} \min \quad & \Delta \qquad \text{[MaxDev-CP]} \\ \text{s.t.} \quad & \text{Constraints (9) to (15).} \end{aligned}$$

III. Experimental Results

This section presents the numerical results obtained for the MILP and CP formulations introduced in Sections II-C and II-D.

A. Experimental setup

All experiments were conducted using the Gurobi [5] solver for the MILP model and the IBM CP Optimizer [6] (via Docplex) for the CP model. For MILP formulations, Gurobi applies a Branch-and-Bound algorithm combined with cutting planes and presolve routines to explore the search space. For the CP model, the IBM CP Optimizer employs a depth-first search strategy enhanced by constraint propagation, domain reduction, and dynamic variable ordering. Default solver parameters were used in all cases. A single thread was enforced with a time limit of one hour. All runs were executed on a high-performance computing system equipped with 80 cores running at 2.10 GHz and 1 TB of memory.

Each experiment uses a set of generated flight intentions, as detailed in Section III-B. Each flight intention includes a departure and arrival node, a fixed horizontal path, a scheduled take-off time, and a cruise speed range between $4\ m/s$ and $10\ m/s$.

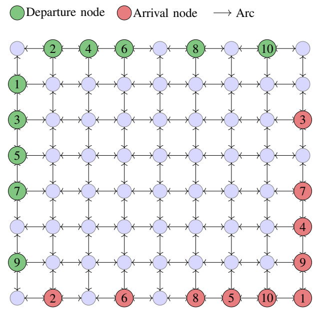

Fig. 1. Grid graph showing fixed departure (green) and arrival (red) node pairs.

B. Test instances

Two types of test instances are considered to evaluate the models. The first type is based on synthetic grids specifically designed to induce a high density of interconnected PLoS. By interconnected PLoS, we refer to situations where resolving one PLoS directly activate another. The second type consists of Vienna instances built on the urban network of the city of Vienna.

1) Instances with high density of interconnected PLoS: These instances are built on a synthetic grid graph, designed to define the structure of flight interactions.

The graph consists of 72 nodes and 254 arcs, with a fixed arc length per instance. Five arc lengths are considered: 60 meters, 250 meters, 500 meters, 1000 meters, and 2000 meters. To control traffic density, 10 representative flights have been designed. Their departure and arrival nodes are illustrated in Fig.1, and their scheduled take-off times (in seconds) are provided in TABLE I. These 10 representative flights are replicated to generate instances with 20, 30, 40, 50, and 60 flight intentions. Each additional group of 10 flights differs by a 4-second shift in the scheduled take-off time. A maximum ground delay $\delta^{\max}$ of 60 seconds (1 minute) is allowed for all flights.

TABLE I
Scheduled take-off times (in seconds) for the first 10 flights, relative to that of the first flight.

Pair	1	2	3	4	5	6	7	8	9	10
Scheduled take-off time (s)	0	6	13	7	8	12	12	16	31	32

For each number of flights and arc length, five instances are generated by randomly selecting, for each flight, a horizontal path from a predefined set of available paths. This results in a total of 25 Grid-based instances for each number of flights. The instance generation process is computationally inexpensive, with an average time of less than one second per instance.

2) Vienna instances: These instances are based on the urban network of the city of Vienna. The graph contains 4,441 nodes and 7,287 arcs, and spans a metropolitan area approximately 12 kilometers in diameter (see Fig. 2). Departure and arrival nodes are selected randomly, with the aim of producing flight paths that are spatially dispersed across the network.

Fig. 2. Vienna city graph.

We consider test sets with 50, 100, 150, 200, 250, and 300 flights. For each flight, the shortest horizontal path between a selected departure/arrival nodes is computed using the A* algorithm. Fifty representative instances are generated per number of flights. Each instance is produced in less than two seconds on average.

C. Results on instances with high density of interconnected PLoS

The results obtained on the considered Grid-based instances highlight clear differences in performance between models as the number of flights increases.

For the MaxDev objective, the CP formulation maintains consistent performance across all traffic levels and arc lengths, as shown in TABLE II, which presents the average solving time (in seconds) for each combination of arc length and number of flights. All instances are solved except for a single case with 60 flights. The only unresolved instance corresponds to the most constrained configuration, involving 60 flights and an arc length of 60 meters. In contrast, the number of instances solved to the optimality by the MaxDev-MILP decreases with increasing number of flights. As shown in TABLE III, which reports the number of optimally solved instances for each combination of number of flights and arc length, the MILP model solves only 21 out of 25 instances at 40 flights, 14 at 50 flights, and 4 at 60 flights. The most significant computational difficulties are observed for short arc lengths, particularly 60 meters. In such cases, the limited traversal time induces narrow feasible domains for the temporal variables. This considerably restricts the range of possible time shifts available to satisfy separation constraints, especially since a minimum separation distance of 32 meters—adopted from the work of [7]—is imposed between drones. In contrast, the CP model handles these tightly-constrained instances more efficiently. This behavior is clearly reflected in the solving times reported in TABLE II. For instance, at 50 flights with 60-meter arcs, MaxDev-CP solves all five instances in under 360 seconds on average, while MaxDev-MILP exceeds 3400 seconds.

TABLE II
AVERAGE SOLVING TIME (IN SECONDS) FOR CP AND MILP MODELS ON GRID-BASED INSTANCES, FOR DIFFERENT NUMBERS OF FLIGHTS AND ARC LENGTHS, WITH A ONE-HOUR TIME LIMIT.

		20 flights			
Arc length (meters)	60	250	500	1000	2000
MaxDev-MILP	0.74	0.06	0.04	0.04	0.05
MaxDev-CP	2.81	0.09	0.06	0.06	0.07
SumDev-MILP	3600.0	0.34	0.35	0.35	0.38
SumDev-CP	3600.0	13.52	13.33	13.26	13.67
		30 flights			
Arc length (meters)	60	250	500	1000	2000
MaxDev-MILP	203.74	0.88	0.84	0.83	0.93
MaxDev-CP	7.18	1.52	1.50	1.50	1.60
SumDev-MILP	3600.0	2966.43	3150.54	3020.26	2928.27
SumDev-CP	3600.0	3600.0	3600.0	3600.0	3600.0
		40 flights			
Arc length (meters)	60	250	500	1000	2000
MaxDev-MILP	3596.55	5.03	4.94	4.95	5.14
MaxDev-CP	20.52	2.94	2.96	2.94	2.94
SumDev-MILP	3600.0	3600.0	3600.0	3600.0	3600.0
SumDev-CP	3600.0	3600.0	3600.0	3600.0	3600.0
		50 flights			
Arc length (meters)	60	250	500	1000	2000
MaxDev-MILP	3439.21	1484.84	1509.12	1527.03	1533.80
MaxDev-CP	356.58	18.11	17.60	17.32	26.75
SumDev-MILP	3600.0	3600.0	3600.0	3600.0	3600.0
SumDev-CP	3600.0	3600.0	3600.0	3600.0	3600.0
		60 flights			
Arc length (meters)	60	250	500	1000	2000
MaxDev-MILP	3600.0	3002.21	3055.06	3011.09	2996.38
MaxDev-CP	1031.30	212.06	236.13	287.37	299.82
SumDev-MILP	3600.0	3600.0	3600.0	3600.0	3600.0
SumDev-CP	3600.0	3600.0	3600.0	3600.0	3600.0

Fig. 3 provides a detailed comparative analysis of the objective values and lower bounds obtained by the MaxDev-MILP and MaxDev-CP across Grid-based instances. Fig. 3a reports the differences in objective values (MaxDev-MILP minus MaxDev-CP) for each arc length and number of flights, while Fig. 3b displays the corresponding differences in lower bounds. Each color encodes a specific arc length category, and for each number of flights (40, 50, and 60), five instances are considered per arc length.

As shown in Fig. 3a, for 40 and 50 flights, the objective values returned by MaxDev-MILP and MaxDev-CP are identical across all instances when the arc length is 250 meters or greater. However, at 60 flights, deviations appear even

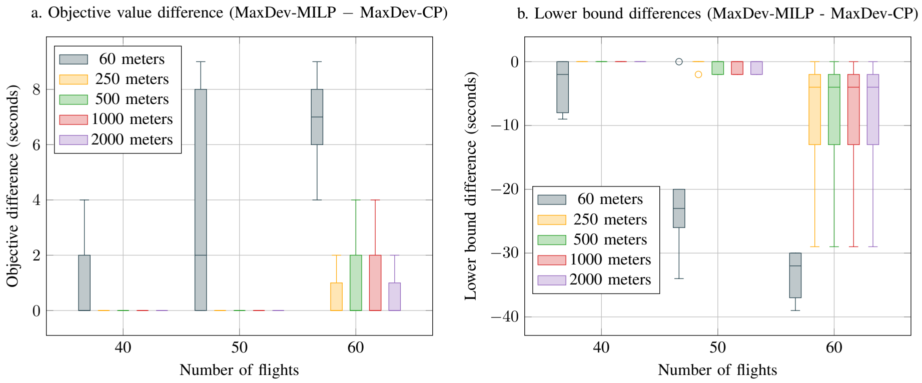

Fig. 3. Analysis of the MaxDev objective and lower bound values obtained by MaxDev-MILP and MaxDev-CP across Grid-based instances with a one-hour time limit.

TABLE III
NUMBER OF GRID-BASED INSTANCES SOLVED TO OPTIMALITY WITHIN THE ONE-HOUR TIME LIMIT, FOR VARYING NUMBERS OF FLIGHTS AND ARC LENGTHS.

Number of flights	20					30					40					50					60				
Arc length (meters)	60	250	500	1000	2000	60	250	500	1000	2000	60	250	500	1000	2000	60	250	500	1000	2000	60	250	500	1000	2000
MaxDev-MILP	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	1/5	5/5	5/5	5/5	5/5	1/5	4/5	3/5	3/5	3/5	0/5	1/5	1/5	1/5	1/5
MaxDev-CP	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	4/5	5/5	5/5	5/5	5/5
SumDev-MILP	0/5	5/5	5/5	5/5	5/5	0/5	2/5	2/5	2/5	2/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5
SumDev-CP	0/5	5/5	5/5	5/5	5/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5

for longer arcs, with several non-zero differences up to 4-9 seconds observed across all arc categories. The largest discrepancies occur for 60-meter arcs, where all instances display objective gaps, indicating that the MILP formulation fails to match the optimal solution achieved by CP. Fig. 3b further highlights the limitations of MILP in these configurations. For 60-meter arcs under high traffic levels (50 and 60 flights), the differences between the MILP lower bounds and the lower bounds returned by CP reach up to 30 seconds.

For the SumDev objective, both SumDev-MILP and SumDev-CP reach the one-hour time limit in instances with 30 flights, and make solving time a less informative indicator for their comparaison (see TABLE II). Fig. 4d shows the mean time at which the last bound update occurs, illustrating a clear distinction in the resolution strategies applied by the two approaches. In SumDev-CP, bounds are typically updated within the first few seconds and remain unchanged thereafter. At 20 flights, for instance, the last update occurs on average after 25 seconds. In contrast, MILP continues to refine bounds until the time limit.

Fig. 4a illustrates the number of instances where SumDev-CP achieves a lower SumDev objective than SumDev-MILP, across all considered number of flights. As shown in Fig. 4a, the number of instances where CP outperforms MILP varies with the number of flights. At 40 flights, CP outperforms MILP in 13 instances versus 12. At 50 flights, the advantage shifts to MILP, with 14 instances where it provides better solutions compared to 6 for CP. However, at 60 flights, CP outperforms MILP in 19 out of 25 instances. Fig. 4b shows the distribution of objective value differences (SumDev-MILP minus SumDev-CP). For traffic levels ranging from 20 to 50 flights, MILP tends to yield better solutions than CP for arc lengths of 250 meters and above. Most of the differences are small or negative, indicating slightly better objective values obtained by MILP. In contrast, for 60-meter arcs, CP outperforms MILP. This is especially clear at 40 and 50 flights, where the differences in favor of CP reach up to +80 and +56 seconds respectively. The MILP model appears less effective in handling short arcs. At 60 flights, CP yields better results than MILP across all arc lengths in the tested instances. In particular, for 60-meter arcs, the objective difference reaches values as high as +183 seconds in favor of CP, and the median difference remains positive for nearly all arc configurations. Fig. 4c displays the difference in lower bound values obtained by SumDev-MILP and SumDev-CP across all arc lengths and numbers of flights. As observed, SumDev-MILP tends to provide tighter bounds than CP throughout all configurations. This confirms the benefit of the continuous relaxation exploited by MILP.

Overall, the CP model demonstrates a clear advantage under the MaxDev objective in high-density of interconnected PLoS settings, both in terms of solving time and number of instances solved. It systematically outperforms the MILP formulation across all traffic levels tested. For SumDev, CP is more

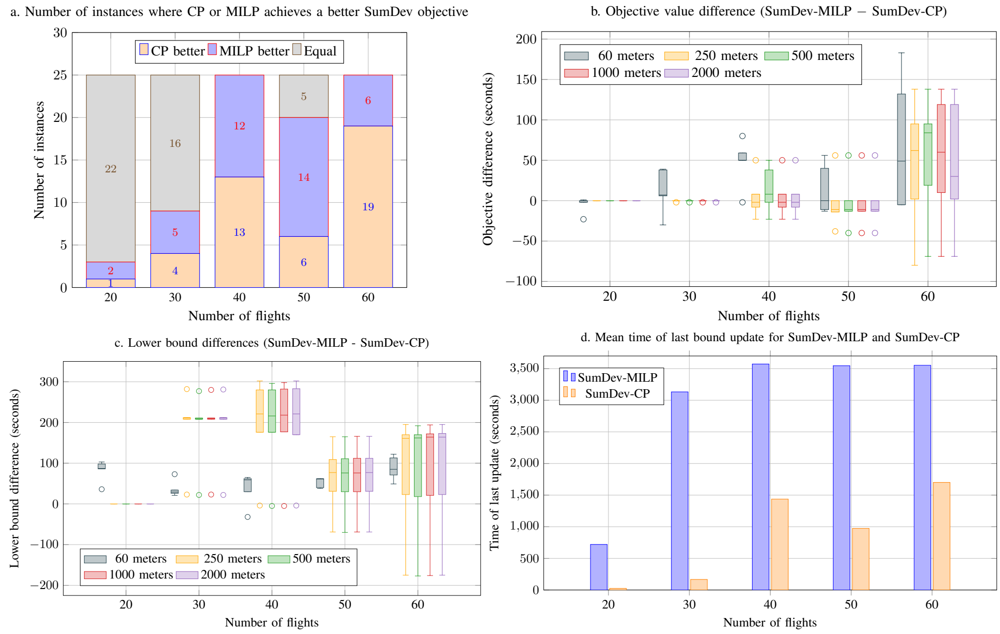

Fig. 4. Analysis of the SumDev objective and lower bound values obtained by SumDev-MILP and SumDev-CP across Grid-based instances with a one-hour time limit.

effective in generating high-quality feasible solutions early in the search, while MILP provides tighter lower bounds and proves optimality more quickly.

D. Results on Vienna instances

The experiments conducted on the Vienna instances exhibit a distinct computational behavior from the Grid-based instances described in Section III-C. These instances are characterized by a lower density of PLoS and a more dispersed network structure (see Fig. 2 and Fig. 5).

TABLE IV
AVERAGE SOLVING TIME (IN SECONDS) FOR CP AND MILP MODELS ON VIENNA INSTANCES, WITH A ONE-HOUR TIME LIMIT.

Number of flights	50	100	150	200	250	300
MaxDev-MILP	≤0.01	0.03	0.07	0.13	0.21	0.39
MaxDev-CP	0.02	0.07	0.17	0.34	0.62	1.01
SumDev-MILP	0.01	0.04	0.08	0.16	0.29	0.49
SumDev-CP	72.02	1080.05	2592.05	3600.00	3600.00	3600.00

TABLE IV shows the average solving times for CP and MILP models on Vienna instances, under a one-hour time limit. Under the MaxDev objective, both MaxDev-MILP and MaxDev-CP solve all instances up to 300 flights to the optimality. Solving times, however, differ. MaxDev-MILP remains consistently faster, with average times below 0.4 seconds compared to 1.01 seconds for CP at 300 flights. This behavior contrasts with the Grid-based instances, where CP was faster under the same objective. The observed shift can be attributed to the reduced number of interconnected PLoS in the Vienna network.

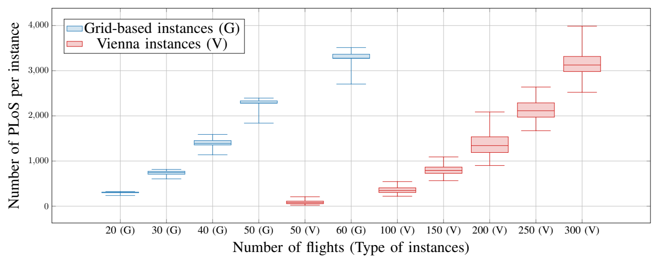

Fig. 5. Boxplot of the number of PLoS per instance as a function of the number of flights

For the SumDev objective, the MILP formulation successfully solves all instances within short computation times, independently of the traffic level. In contrast, the CP model loses its ability to prove optimality beyond 150 flights. This behavior is illustrated in Fig.6a, which displays the number of instances solved to optimality by SumDev-CP as a function of the number of flights. The corresponding differences in lower bounds between the two models are shown in Fig.6b, where the gap increases with problem size, reaching a median of 9.5 seconds at 300 flights.

Despite the inability to certify optimality with a one-hour time limit, both approaches return identical objective values across all traffic levels, as presented in Fig.6c. This observation suggests that the solutions produced by the CP model are optimal. Moreover, the CP formulation updates its lower bound very early during the search, as illustrated in Fig.6d: the average time of the last update remains below 0.02 seconds

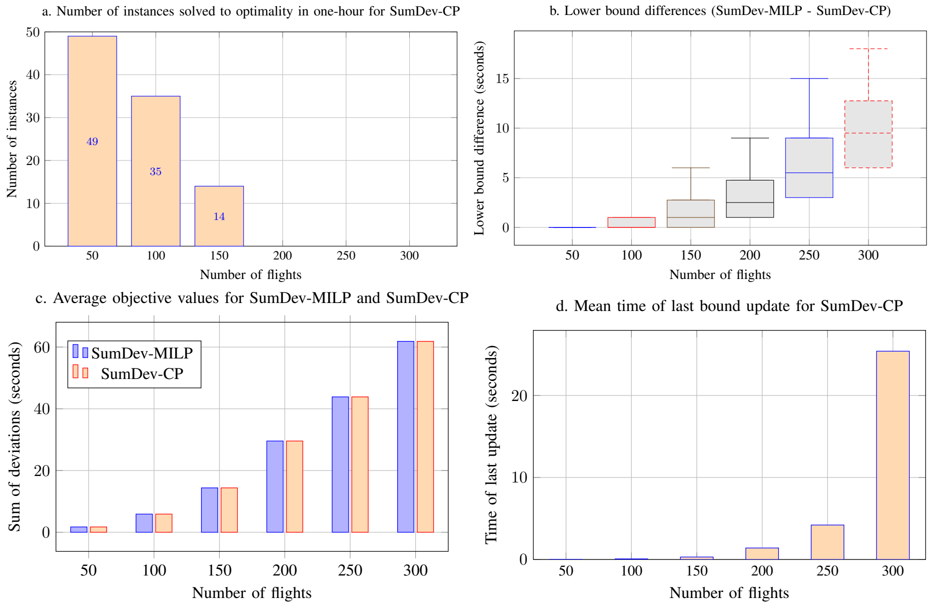

Fig. 6. Analysis of the SumDev objective and lower bound values obtained by SumDev-MILP and SumDev-CP across Vienna instances with a one-hour time limit.

for 50 flights and under 30 seconds for 300 flights.

In summary, the Vienna instances are solved efficiently across all numbers of flights by the MILP formulation. For the MaxDev objective, MILP and CP formulations exhibit comparable performance, with all instances solved to optimality and low solving times. For SumDev, MILP achieves optimality in all cases, while CP finds solutions with the same objective value but exhibits increasing optimality gaps as traffic increases. The early stagnation of CP bound updates limits its ability to certify optimality in large-scale configurations.

IV. Conclusion and Future Research

This paper compared MILP and CP formulations for optimization of urban drone trajectories under two deviation-based objectives. Numerical experiments on Grid-based and Vienna instances revealed that CP outperforms MILP for MaxDev in instances with high density of interconnected PLoS, while MILP provides tighter lower bounds for SumDev and scales better on Vienna instances. CP proves effective in quickly identifying good solutions, though it struggles to improve bounds. These results confirm the complementary strengths of both approaches, depending on the objective and instance structure. CP can serve as an effective tool for generating initial feasible solutions to warm-start MILP models. Future research directions include the integration of these models into decomposition approaches.

Acknowledgment

The authors thank Carla Juvin for insightful discussions on Constraint Programming formulations.

References

[1] Zahraa Asfour, Sonia Cafieri, and Andrija Vidosavljevic. Optimization of unmanned air vehicles trajectories in urban air mobility. Preprint, May 2025.

[2] Denis Bereziat, Sonia Cafieri, and Andrija Vidosavljevic. Metropolis II: Centralised and strategical separation management of UAS in urban environment. In *Proceedings of the 12th SESAR Innovation Days, Budapest, Hungary, 5-8 December 2022*, 2022.

[3] Geoffrey Ding and Hamsa Balakrishnan. Lexicographic min-max fairness in task assignments. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 3408–3414. IEEE, 2023. doi: 10.1109/CDC49753.2023.10384128.

[4] Malik Doole, Joost Ellerbroek, and Jacco Hoekstra. Estimation of traffic density from drone-based delivery in very low level urban airspace. *Journal of Air Transport Management*, 88:101862, 2020. doi: 10.1016/j.jairtraman.2020.101862.

[5] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, , 2025. https://www.gurobi.com/ (accessed: February 20, 2025).

[6] IBM ILOG CPLEX Optimization Studio CPLEX User's Manual, , 2025. https://www.ibm.com/fr-fr/products/ilog-cplex-optimization-studio (accessed: May 14, 2025).

[7] Andres Morfin Veytia, Calin Andrei Badea, Niki Patrinopoulou, Ioannis Daramouskas, Joost Ellerbroek, Vaios Lappas, Vassilios Kostopoulos, and Jacco Hoekstra. U-Space Utilisation of Airspace under Various Layer Function Assignments and Allocations. *Drones*, 7(7):444, 2023. doi: 10.3390/drones7070444.

[8] Mercedes Pelegrin, Claudia d'Ambrosio, Rémi Delmas, and Youssef Hamadi. Urban Air Mobility: From complex tactical conflict resolution to network design and fairness insights. *Optimization Methods and Software*, 38(6):1311–1343, 2023. doi: 10.1080/10556788.2023.2241148.

[9] Tom Portoleau and Claudia D'Ambrosio. A robust two-stage model for the urban air mobility flight scheduling problem. In *Proceedings of the 8th International Symposium on Combinatorial Optimization, ISCO 2024, La Laguna, Tenerife, Spain, May 22–24, 2024*. Springer, 2024. doi: 10.1007/978-3-031-60924-4_26.