Feasibility pump for aircraft deconfliction with speed regulation

Sonia Cafieri, Claudia D'Ambrosio

J Glob Optim
DOI 10.1007/s10898-017-0560-7

Feasibility pump for aircraft deconfliction with speed regulation

Sonia Cafieri[1] · **Claudia D'Ambrosio**[2]

Received: 24 January 2017 / Accepted: 20 August 2017

Abstract We propose Feasibility Pump heuristics for the crucial problem of aircraft conflict avoidance arising in air traffic management. This problem can be modeled as a mixed integer nonlinear optimization problem, whose solution can be very computationally demanding. Feasibility Pump is an iterative algorithm that, at each iteration, solves alternatively two easier subproblems represented by relaxations of the original problem, minimizing the distance between their solutions. We propose in this paper specific formulations for the subproblems to be handled, tailored to the problem at hand. Numerical results show that, on the considered test problems, good-quality, in some cases optimal, feasible solutions are always obtained.

Keywords Mixed integer nonlinear programming · Feasibility pump · Aircraft conflict avoidance · Mathematical programming · Reformulations · Heuristic algorithm

1 Introduction

Aircraft conflict avoidance for en-route flights constitutes a prominent example of problem that urgently needs to be addressed in Air Traffic Management (ATM) to ensure a higher level of automation, and consequently more efficiency and safety in the current context of growing air traffic on the world-scale.

Aircraft sharing the same airspace are said to be *in conflict* when they get too close to each other during their flight, according to their predicted trajectories. More specifically, a conflict is due to a loss of separation between aircraft, occurring when their relative horizontal and vertical distances do not satisfy anymore two given safety-distance constraints. The problem

✉ Claudia D'Ambrosio
dambrosio@lix.polytechnique.fr

Sonia Cafieri
sonia.cafieri@enac.fr

[1] Université de Toulouse, ENAC, 31055 Toulouse, France

[2] LIX CNRS (UMR7161), École Polytechnique, 91128 Palaiseau Cedex, France

Published online: 25 September 2017

is then to identify a potential loss of separation between pairs of aircraft in a monitored portion of the airspace, and to issue suitable separation maneuvers to provide a new, conflict-free, aircraft configuration.

Aircraft conflict avoidance is naturally modeled as an optimization problem, as one usually seeks to separate aircraft by deviating as little as possible from their original flight plans, i.e., minimizing the impact of the separation maneuvers on the flight efficiency. In this paper, we focus on approaches based on mixed integer nonlinear programming (MINLP), that is shown to be a powerful framework for aircraft conflict avoidance, and is attracting a growing attention in recent years. An overview of MINLP modeling for aircraft conflict avoidance is provided in [9]. Modeling is strictly dependent on the separation maneuvers chosen to solve conflicts. The most common way to achieve separation by air traffic controllers is based on aircraft heading angle deviations. Another, although less preferred, way, uses flight level changes to separate aircraft. Conflict avoidance can also be performed through aircraft velocity regulation. This kind of regulation can be carried out through a *subliminal control*, promoted in recent years by the European aeronautical project ERASMUS [7], according to which aircraft speeds should be modified only in a very small range (namely, from -6 to $+3\%$) around the original speeds. A subliminal speed control is considered promising in view of more automated ATM systems in a next future, thanks to its limited impact on the workload of air traffic controllers. Therefore, in the present paper we consider this kind of separation maneuver.

The first approaches based on mixed integer optimization date back to 2002. In [18,21], a geometrical construction on aircraft trajectories to express aircraft separation leads to the definition of Mixed Integer Linear Programming (MILP) models based on velocity changes or heading angle changes. The work of Pallottino et al. [18] is more recently extended in [1–3], where mixed integer linear and nonlinear models based on various aircraft separation techniques are presented. Speed regulation maneuvers are recently used in [19,20], where a space-discretization approach is used to represent aircraft trajectories and separation. MINLP based on speed regulation is proposed in [10], that in this paper we consider as a reference for modeling (see Sect. 2), [8], and, more recently, [11,12].

MINLP models for aircraft conflict avoidance are, for their nature, quite complex. The complexity is mainly due to the pairwise nonlinear nonconvex separation constraints. This makes exact solution algorithms very computationally demanding when the number of aircraft considered simultaneously is large. In this context, it is interesting to obtain good quality feasible solutions for the problem under consideration, or, on the contrary, to detect its infeasibility. This interest is twofold. First, as decision variables and constraints in MINLP models for conflict avoidance depend on the aircraft maneuvers chosen to perform separation, getting feasibility information with respect to the selected maneuvers may help to adjust the mathematical model. Second, it is known that a good feasible solution provides a good upper (in the case of a minimization problem) bound to be used as a cutoff value within a branch-and-bound algorithm, and may also allow tighter bounds to be propagated through bound tightening techniques. In this paper we focus on feasibility seeking in MINLP programs for aircraft conflict avoidance. More specifically, the main contribution of this paper is represented by Feasibility Pump heuristics tailored to the problem of aircraft conflict avoidance. Feasibility Pump [6,14] builds two sequences of points, one of NLP feasible solutions (i.e, feasible for a continuous relaxation of the problem), and the other of solutions with integral value for the integer variables (but possibly violating the constraints), and iterates until the two sequences converge to a feasible solution of the MINLP. The two sequences are generated by solving two subproblems obtained from the original MINLP. Starting from the general framework of Feasibility Pump for MINLP, we propose specific formulations for the sub-

problems to be handled, tailored to the problem at hand. Moreover, we improve Feasibility Pump so that, once it finds a first feasible solution, it tries to improve it thanks to a so-called "optimality cut".

The paper is organized as follows. In Sect. 2 we introduce the aircraft conflict avoidance (deconfliction) problem and recall the MINLP model based on speed regulation due to Cafieri and Durand [10]. Some improvements to this model based on preprocessing and reformulations are proposed in Sect. 3. We present Feasibility Pump heuristics tailored to the specific problem of interest in Sect. 4. In Sect. 5, we present and discuss the results of numerical tests, validating the proposed approach. Section 6 concludes the papers with future research directions.

2 The aircraft deconfliction problem with speed regulation

In this section we first describe in some more detail the problem under consideration, then recall the mathematical program proposed by Cafieri and Durand [10], here used as reference model.

Let us consider a portion of the airspace, monitored during a time window. We consider, in particular, the case of en-route cruise flights, at a *tactical level*, i.e., potential conflicts are resolved a few minutes before the loss of separation potentially occurs. We are given a set A of aircraft, that are all at the same flight level, implying that only the horizontal separation distance has to be considered for each pair of aircraft to ensure safety. Starting from a current configuration, characterized by initial position (in 2-dimensional space), heading angle, and speed for each aircraft, the problem consists in finding a new configuration that is conflict-free along all the observed time window.

In our model, aircraft are allowed to change once their speed, in particular following the subliminal control paradigm, while aircraft headings are kept fixed. Following [10], the main decision variables are, for all $i \in A$, continuous variables q_i, expressing the percentage of speed variation of aircraft i. These variables are bounded for operational reasons and, when subliminal speed control is applied, these bounds are very tight, as explained in Sect. 1. The main constraints are represented by the pairwise aircraft separation constraints:

$$\|\mathbf{x}_i(t) - \mathbf{x}_j(t)\| \geq d \quad \forall t \geq 0,\ (i, j) \in B \tag{1}$$

where $\mathbf{x}_i(t)$ denotes the position of aircraft i at time t, d is the horizontal separation standard, $\|\cdot\|$ is the Euclidean norm, and the set $B = \{(i, j) \mid i \in A, j \in A, i < j\}$ is introduced for ease of notation. The model in [10] does not rely on any time discretization for the above constraints and a reformulation is used instead. Assuming that uniform motion laws apply, the position $\mathbf{x}_i(t)$ is given by $\mathbf{x}_i(t) = \mathbf{x}_i(0) + t\ \mathbf{v}_i q_i$, where both the initial position $\mathbf{x}_i(0)$ and velocity $\mathbf{v}_i$ are known vectors, while variable q_i represents the speed variation. By substituting in (1) and squaring, the second-degree polynomial function $t^2\|\mathbf{v}_i q_i - \mathbf{v}_j q_j\|^2 + 2t\left(\mathbf{v}_i q_i - \mathbf{v}_j q_j\right) \cdot \mathbf{x}_{ij}^0 + \|\mathbf{x}_{ij}^0\|^2$ is obtained (where $\mathbf{x}_{ij}^0$ is the relative position of aircraft i with respect to aircraft j at time $t = 0$) which attains its minimum in $[0, \infty)$ at time instant

$$t_{ij}^m = \frac{-\left(\mathbf{v}_i q_i - \mathbf{v}_j q_j\right) \cdot \mathbf{x}_{ij}^0}{\|\mathbf{v}_i q_i - \mathbf{v}_j q_j\|^2}.$$

For all $(i, j) \in B$, variables t_{ij}^m are continuous variables of the problem, and the separation constraints are reformulated as

$$\|\mathbf{v}_i q_i - \mathbf{v}_j q_j\|^2(\|\mathbf{x}_{ij}^0\|^2 - d^2) - (\mathbf{x}_{ij}^0 \cdot (\mathbf{v}_i q_i - \mathbf{v}_j q_j))^2 \geq 0 \quad \text{if} \quad t_{ij}^m \geq 0. \tag{2}$$

These constraints being piecewise-defined, a binary variable y_{ij}, for each $(i, j) \in B$, is introduced as

$$y_{ij} = \begin{cases} 1 & \text{if } t_{ij}^m \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

and constraints (2) are re-written accordingly as explained in the following.

Introducing for all $(i, j) \in B$ (continuous) auxiliary variables p_{ij}, representing the inner product appearing in the separation constraint, and w_{ij}, representing the square of the relative velocity, the constraints of the problem can be summarized as follows:

– defining constraints for p_{ij} and w_{ij}, respectively linear and quadratic:

$$p_{ij} = \mathbf{x}_{ij}^0 \cdot (\mathbf{v}_i q_i - \mathbf{v}_j q_j) \quad \forall (i, j) \in B \tag{3}$$

$$w_{ij} = \|\mathbf{v}_i q_i - \mathbf{v}_j q_j\|^2 \quad \forall (i, j) \in B \tag{4}$$

– defining constraints for t_{ij}^m, involving a bilinear product:

$$t_{ij}^m w_{ij} + p_{ij} = 0 \quad \forall (i, j) \in B \tag{5}$$

– constraints checking the sign of t_{ij}^m, involving a bilinear product with a binary variable:

$$t_{ij}^m (2y_{ij} - 1) \geq 0 \quad \forall (i, j) \in B \tag{6}$$

– pairwise separation constraints, involving a quadratic term and a product with a binary variable:

$$y_{ij} \left(\|\mathbf{v}_i q_i - \mathbf{v}_j q_j\|^2 \left(\|\mathbf{x}_{ij}^0\|^2 - d^2 \right) - \left(\mathbf{x}_{ij}^0 \cdot \left(\mathbf{v}_i q_i - \mathbf{v}_j q_j \right) \right)^2 \right) \geq 0 \quad \forall (i, j) \in B. \tag{7}$$

As an objective function, we consider the sum of the deviations of the aircraft speeds, to be minimized:

$$\min \sum_{i=1}^{n} (q_i - 1)^2. \tag{8}$$

A list of sets, parameters, and variables of the model is displayed in Table 1.

The problem formulation is a mixed integer nonlinear program. In the next sections, we discuss some possible reformulations and we propose a Feasibility Pump approach tailored to the problem.

Table 1 Sets, parameters, and variables used in the MINLP

Sets and parameters	
A	Set of aircraft
d	Standard separation distance
$\mathbf{x}_i(0)$	Initial position vector of aircraft i
$\mathbf{v}_i$	Initial velocity vector of aircraft i
Variables	
q_i	Percentage of speed variation of aircraft i
t_{ij}^m	Time when the separation function between i and j attains its minimum in $[0, \infty)$
y_{ij}	(Binary) variables to model logical conditions
p_{ij}	Auxiliary variables for the inner product in the separation constraint
w_{ij}	Auxiliary variables for the square of the relative velocity

3 On mathematical model reformulations

In this section we present some exact reformulations and relaxations of MINLP problem (3)–(8). The reason is twofold: first, we aim at obtaining a MINLP problem that is easier to solve by standard MINLP solvers. Second, as explained in details in Sect. 4, we will need such reformulations for designing the Feasibility Pump algorithm.

3.1 Project out variable t_{ij}^m

Firstly, note that, $\forall (i, j) \in B$, t_{ij}^m is a dependent variable. However, it appears only in constraint (5), where it is defined, and in constraint (6) where it is linked to binary variable y_{ij}.

Let us replace $t_{ij}^m = -\frac{p_{ij}}{w_{ij}}$ in constraint (6) and simplify it by exploiting the fact that, by definition, w_{ij} is non-negative:

$$-p_{ij}(2y_{ij} - 1) \geq 0 \quad \forall (i, j) \in B.$$

To make the constraint above linear, we can observe that its meaning is:

$$y_{ij} = \begin{cases} 1 & \text{if} \quad p_{ij} < 0 \\ 0 & \text{otherwise} \end{cases} \qquad \forall (i, j) \in B. \tag{9}$$

Now, the same can be obtained by linking variables p and y with the two linear constraints below:

$$\underline{p}_{ij} y_{ij} \leq p_{ij} \leq \overline{p}_{ij}(1 - y_{ij}) \quad \forall (i, j) \in B \tag{10}$$

where $\forall (i, j) \in B$, $\underline{p}_{ij}$ and $\overline{p}_{ij}$ are a lower and an upper bound on variable p_{ij}, respectively. These bounds can be easily found as the only variables appearing in the definition of p_{ij} (3), i.e., the q variables, are bounded for operational reasons. Thus, we replace nonlinear constraints (5) and (6) by linear constraints (10).

3.2 Linearize the separation constraint

Constraints (7) are disjunctive constraints modeling the fact that, $\forall (i, j) \in B$, the separation constraints have to be active if and only if $y_{ij} = 1$. The same effect can be obtained by rewriting it as:

$$\|\mathbf{v}_i q_i - \mathbf{v}_j q_j\|^2 \left(\|\mathbf{x}_{ij}^0\|^2 - d^2\right) - \left(\mathbf{x}_{ij}^0 \cdot \left(\mathbf{v}_i q_i - \mathbf{v}_j q_j\right)\right)^2 \geq M_{ij}(1 - y_{ij}) \qquad \forall (i, j) \in B$$

where M_{ij} is the so-called bigM, i.e., a "large enough" constant. More details about how to compute it are provided in Section 4.2. Separation constraints above are still nonlinear. To make another step forward to linearization, $\forall (i, j) \in B$ we replace $\|\mathbf{v}_i q_i - \mathbf{v}_j q_j\|^2$ with w_{ij}, introduce an additional variable, $z_{ij} = p_{ij}^2 = \left(\mathbf{x}_{ij}^0 \cdot \left(\mathbf{v}_i q_i - \mathbf{v}_j q_j\right)\right)^2$, and replace the last nonlinear term with it. Now we have linear separation constraints:

$$\left(\|\mathbf{x}_{ij}^0\|^2 - d^2\right) w_{ij} - z_{ij} \geq M_{ij}(1 - y_{ij}) \quad \forall (i, j) \in B. \tag{11}$$

3.3 The MINLP reformulation

We report the complete reformulation in the following:

$$\min \sum_{i \in A} (q_i - 1)^2 \tag{12}$$

$$a_k = v_{ik} q_i - v_{jk} q_j \qquad \forall (i, j) \in B, k \in \{1, 2\} \tag{13}$$

$$b_k = a_k^2 \qquad \forall (i, j) \in B, k \in \{1, 2\} \tag{14}$$

$$p_{ij} = \sum_{k \in \{1,2\}} x_{ijk}^0 a_k \qquad \forall (i, j) \in B \tag{15}$$

$$z_{ij} = p_{ij}^2 \qquad \forall (i, j) \in B \tag{16}$$

$$w_{ij} = \sum_{k \in \{1,2\}} b_k \qquad \forall (i, j) \in B \tag{17}$$

$$\underline{p}_{ij} y_{ij} \leq p_{ij} \leq \overline{p}_{ij} (1 - y_{ij}) \qquad \forall (i, j) \in B \tag{18}$$

$$\left(\|\mathbf{x}_{ij}^0\|^2 - d^2 \right) w_{ij} - z_{ij} \geq M_{ij} (1 - y_{ij}) \qquad \forall (i, j) \in B \tag{19}$$

$$y_{ij} \in \{0, 1\} \qquad \forall (i, j) \in B \tag{20}$$

where $\forall i \in A, k \in \{1, 2\}$, v_{ik} represents the kth component of the velocity vector $\mathbf{v}_i$ in a 2-dimensional space and x_{ijk}^0 the kth component of the vector $\mathbf{x}_{ij}^0$ of relative initial positions of aircraft i and j.

Note that the nonlinear terms are isolated in the second and the forth sets of constraints. A MILP relaxation can be easily obtained by applying standard linear relaxations to these two sets of constraints. In particular, we do it by the classical method of generating the tangents to the curves representing the nonlinear terms at some points (see, for example, [17]).

4 Feasibility Pump Algorithms

In this section, we first introduce the general scheme of Feasibility Pump algorithms and then present a new version, tailored to solve the aircraft deconfliction with speed regulation problem.

4.1 The general scheme of Feasibility Pump

Feasibility Pump (FP) was introduced by Fischetti et al. [15] for mixed integer linear programming problems, then extended to convex MINLPs [6] and to nonconvex MINLPs [13,14]. The main idea is to decompose the problem into two subproblems that are easier to solve, and iteratively solve these two subproblems by trying to minimize, for each of them, the distance to the best solution of the other subproblem. The subproblems are modified at each iteration until the two solutions coincide, i.e., a feasible solution is found.

More formally, let us define a generic MINLP problem:

$$(P) \begin{cases} \min f(x, y) \\ g(x, y) \leq 0 \\ x \in X \\ y \in Y \cap \mathbb{Z}^p, \end{cases}$$

where X, Y are two polyhedra of appropriate dimension, $f : \mathbb{R}^{n+p} \to \mathbb{R}$ is convex without loss of generality, and functions $g : \mathbb{R}^{n+p} \to \mathbb{R}^m$ are nonlinear and, possibly, nonconvex. Problem (P) is well-known to be, in general, practically difficult to solve. However, we can identify two relaxations of problem (P) that are "easier" to solve, like, for example:

$$(P_1)\begin{cases}\min f(x, y)\\ g(x, y) \le 0\\ x \in X\\ y \in Y\end{cases} \qquad (P_2)\begin{cases}\min f(x, y)\\ \overline{g}(x, y) \le 0\\ x \in X\\ y \in Y \cap \mathbb{Z}^p\end{cases}$$

where $\overline{g}$ represents a convex (maybe linear) relaxation of g. From a theoretical viewpoint, (P_1) and (P_2) are as difficult to solve as (P). In particular, (P_1) is possibly a nonconvex NLP, while (P_2) is a convex MINLP. However, compared to (P), the two relaxations deal with one source of nonconvexity at time, thus making them easier to solve in practice. Note that $(P) = (P_1) \cap (P_2)$, thus a point (x^*, y^*) that is feasible for both (P_1) and (P_2) is also feasible for (P).

In the basic version of FP, we neglect the objective function $f(x, y)$ because the aim is finding any feasible solution. We will discuss this in detail in the next section. The general scheme of basic Feasibility Pump algorithm can be summarized in Algorithm 1. The ini-

Algorithm 1: FP general scheme

1: $\ell = 0$;
2: initialize $(\hat{x}^0, \hat{y}^0)$;
3: **repeat**
4: $\quad \ell + +$;
5: $\quad$ Find $(\tilde{x}^\ell, \tilde{y}^\ell)$, i.e., the solution of (P_1) that minimizes $\|(x, y) - (\hat{x}^{\ell-1}, \hat{y}^{\ell-1})\|$;
6: $\quad$ Find $(\hat{x}^\ell, \hat{y}^\ell)$, i.e., the solution of (P_2) that minimizes $\|(x, y) - (\tilde{x}^\ell, \tilde{x}^\ell)\|$;
7: **until** $(((\hat{x}^\ell, \hat{y}^\ell) == (\tilde{x}^\ell, \tilde{y}^\ell))$ **or** time/iteration limit reached)

tialization step (step 2: in Algorithm 1) typically consists on solving locally the continuous relaxation of (P) and, in case the integer variables take fractional values, round them. The corresponding solution might be infeasible but it can be used as starting point $(\hat{x}^0, \hat{y}^0)$.

In the next section, we present a new version of FP, tailored for the aircraft deconfliction with speed regulation problem. We refer the reader to [6,14,15] for details on FP algorithms for general MI(N)LP problems.

4.2 Tailored FP Algorithm

We now describe in details the tailored version of FP algorithm that we designed and developed. First of all, let us define subproblems (P_1) and (P_2).

4.2.1 Subproblem (P_1)

As explained in the previous section, subproblem (P_1) represents a continuous relaxation of (P). We obtain it by dropping only the integrality requirements, i.e., (20). At iteration ℓ of the FP algorithm, we have $(P_1)^\ell$. Its objective function is defined as follows:

$$\min \|y - \hat{y}^{\ell-1}\|_1 + \omega \sum_{i \in A} (q_i - 1)^2 \tag{21}$$

with $\omega \geq 0$. Note that, when ω is set to 0, i.e., when we are considering a pure feasibility problem, solving $(P_1)^\ell$ means: (i) testing the compatibility of values $\hat{y}^{\ell-1}$ with a feasible solution of problem (P); (ii) if no feasible solution corresponds to $\hat{y}^{\ell-1}$, a feasible solution is computed by minimizing the distance $\|y - \hat{y}^{\ell-1}\|$. Setting ω to a value greater than 0 makes also optimality come into the play. As $(P_1)^\ell$ is a nonconvex NLP, it has, in general, multiple local optima. In this context we cannot afford the effort finding the global optimum, thus we solve it to local optimality and might discard feasible solutions of (P). To limit the consequences of dealing with local optima, we can divide the solving phase in two parts:

- Solve $(P_1)^\ell$ to local optimality, but multiple times, i.e., using randomly generated starting points;
- If no solution is found, then solve $(P_{1fix})^\ell$:

$$\min \sum_{i \in A} (q_i - 1)^2 \tag{22}$$

$$(12) - (18) \tag{23}$$

$$y = \hat{y}^{\ell-1} \tag{24}$$

Unfortunately, preliminary computational results showed that considering $(P_{1fix})^\ell$ does not help in converging to a feasible solution of (P). Thus, we will report only results with part 1 above, while skiping part 2.

4.2.2 Subproblem (P_2)

Several approaches to define subproblem (P_2) are possible. The common point is the definition of a problem easier than (P) but that preserves its integrality requirements. In the MILP case [15] solving (P_2) is represented by a simple rounding phase. In this case, (P_2) can be interpreted as min $\|y - \tilde{y}^\ell\|$ subject to integrality requirements and bounds on y and no other constraint. The same approach has been generalized to the convex MINLP case in [5]. When a rounding phase is choosen, an issue that has to be taken into account is the possibility of FP to cycle, i.e., to get stuck on previously seen values of $(\hat{x}^\ell, \hat{y}^\ell)$ and/or $(\tilde{x}^\ell, \tilde{y}^\ell)$. To break cycles, a flipping procedure is typically considered.

Bonami et al. [6] proposed an alternative approach for the convex MINLP case: they define (P_2) as a MILP relaxation of (P) by keeping the linear constraints and linearizing the nonlinear ones throught outer-approximation (OA) cuts added at each iteration so as to cut previously considered solutions. This approach has been generalized to the nonconvex MINLP case by D'Ambrosio et al. [14]: linear constraints are considered, constraints known to define convex feasible subsets are linearized through OA cuts, and the remaining constraints are dropped. Unfortunately this approach cannot guarantee to prevent cycles as it might happen that no OA cut can be added to improve the MILP relaxation. In [14] the authors propose to use a taboo list to discard previously seen solutions of (P_2) by flipping.

Here we propose the following approach: we define (P_2) as (13), (15), (17)–(20) plus the standard secant linearization of constraints (14) and (16). To deal with potential cycles, we adopt the taboo list approach mentioned before.

As for the objective function, we define it as

$$\min \ \|(a, b, q) - (\tilde{a}^\ell, \tilde{b}^\ell, \tilde{q}^\ell)\|_2^2.$$

Note that we linearize the objective function above so as to obtain a MILP relaxation of (P). MILP problems are theoretically difficult, but standard solvers are effective to find the optimal solution. Thus, in Sect. 5, we solve (P_2) to optimality.

Finally, we comment on how we derive lower and upper bounds on b and z variables and a proper value of M. First, note that, as q variables are bounded and a (resp. p) variables depend on them, we can easily derive valid bounds for b variables (resp. z). Note that, for aircraft pairs $(i, j) \in B$ such that aircraft have parallel trajectories, i.e., $\exists k \in \{1, 2\} : v_{ik} = v_{jk}$, we do not impose the separation constraints because we check in a preprocessing step that aircraft flying on parallel trajectories are already separated. We then replace set B with set $\overline{B} = \{(i, j) \in B | v_{i1} \neq v_{j1} \text{ and } v_{i2} \neq v_{j2}\}$.

As w variables depend on b variables, they can be easily bounded too. Consequently, we can derive valid values of parameter M.

4.2.3 Objective function improvement procedure

In a pure feasibility setting, after a feasible solution is found, the algorithm stops. However, we are interested in continuing to improve such a solution. This is done by adding to (P_1) what we call "optimality cut", i.e.,

$$\sum_{i \in A} (q_i - 1)^2 \leq f^* - \varepsilon \tag{25}$$

where f^* is the value of the objective in the best feasible solution of (P) found so far. We stop the algorithm when subproblem (P_1) becomes infeasible or a time/iteration limit is reached. Note that, in the case of (P_1) infeasible, there is no guarantee that no better solution for (P_1) exists, as we solve it to local optimality. However, in practice, the solution found are satisfactory as shown in Sect. 5.

4.2.4 Main scheme

We report the main scheme of the tailored FP algorithm for aircraft deconfliction with speed regulation problem in Algorithm 2.

5 Computational Results

In this section, we present and discuss the results of numerical experiments carried out to test the proposed FP approach. Models were implemented using the AMPL modeling language for optimization problems [16]. We used CPLEX v. 12.6.3 to solve the linear relaxation of the initial problem and IPOPT v. 3.10.2 [22], that implements an Interior-Point method, for the NLP relaxation. These solvers were run with their default settings. The tests were performed on a 2.66 GHz Intel Xeon (octo core) processor with 32GB of RAM and Linux Operating System.

We tested our algorithms on the same sets of problem instances used in [10]. A first set of instances is generated in such a way to have n aircraft in 2-dimensional space, placed on a circle and all headed towards its center (or slightly deviated by $\pm 5^\circ$ with respect to such direction). These instances are denoted with nX where X is replaced by the number of aircraft. Trajectories are straight-lines. These problems are highly symmetric problems, and difficult to solve. The number of conflicts in the same conflict zone, around the center of the circle, amounts to $n(n-1)/2$. A second set of instances is considered, in such a way that aircraft are no more flying towards the center of a circle, but moving along straight intersecting trajectories. This simulates more realistic aircraft configurations. These instances

Algorithm 2: FP Algorithm for the aircraft deconfliction with speed regulation

Require: ω, niter_max, nsp_max
  compute initial solution $(\hat{x}^0, \hat{y}^0)$;
  TL = $\{(\hat{x}^0, \hat{y}^0)\}$, $\ell = 0$, $f^* = +\infty$;
  while ($\ell <$ niter_max) **do**
    $\ell++$;
    for $s = 0; s <$nsp_max; $s++$ **do**
      if ($s > 0$) **then**
        Select randomly the starting point for the NLP solver (within the variables bound ranges);
      end if
      Solve $(P_1)^\ell$ and get the new solution $(\tilde{x}^\ell, \tilde{y}^\ell)$;
      if ($\|\tilde{y}^\ell - \hat{y}^{\ell-1}\|_2 = 0$) **then**
        TL = TL $\cup\{(\tilde{x}^\ell, \tilde{y}^\ell)\}$
        if ($f^* > f(\tilde{x}^\ell, \tilde{y}^\ell)$) **then**
          $(x^*, y^*) = (\tilde{x}^\ell, \tilde{y}^\ell)$
        end if
      end if
      if (a feasible solution for $(P_1)^\ell$ was found) **then**
        break;
      end if
    end for
    if (a feasible solution for $(P_1)^\ell$ was not found) **then**
      break;
    end if
    Solve $(P_2)^\ell$ and get the new solution $(\hat{x}^\ell, \hat{y}^\ell)$;
    if ($\|(\tilde{x}^\ell, \tilde{y}^\ell) - (\hat{x}^\ell, \hat{y}^\ell)\| = 0$) **then**
      TL = TL $\cup\{(\hat{x}^\ell, \hat{y}^\ell)\}$
      if ($f^* > f(\hat{x}^\ell, \hat{y}^\ell)$) **then**
        $(x^*, y^*) = (\hat{x}^\ell, \hat{y}^\ell)$
      end if
    end if
  end while
  if ($f^* < +\infty$) **then**
    return (x^*, y^*);
  end if

are denoted with nX_cY where X is the number of aircrafts and Y the number of conflicts. In both sets of instances, aircraft move initially all at the same speed, namely 400 NM/h (where 1 NM (Nautical Mile) = 1852 m). These speeds are modified after executing the proposed algorithms, in such a way that aircraft are separated, and their new values, according to the subliminal speed control bounds, range from 376 to 412 NM/h. Variables q are indeed in the interval [0.94, 1.03]. The value of the standard aircraft separation d is set to 5 NM.

Our experiments were performed with the following FP setting:

- at each iteration, we generate nsp_max $= 20n(n-1)/2$ starting points to solve subproblem (P_1)
- iteration limit (niter_max): $10n$
- $\omega \in \{0, 0.5, 1\}$.

Each of the values we tried for ω defines a different version of the FP algorithm. We name them FP1, FP2, and FP3, respectively.

We compare the performance of the 3 versions of FP with respect to the one of COUENNE solver v. 0.4, see [4], run on the model proposed by Cafieri and Durand [10] (C&D) with default options, but a time limit of 7200 seconds.

Table 2 reports, for each instance, the objective function value of: (i) the first solution found by each method (columns 2–5); (ii) the best solution found by each method (columns 6–9); (iii) the lower bound computed by COUENNE when solving the mathematical model by Cafieri and Durand [10]. The best value(s) are reported in bold. COUENNE can find the global optimum only for small instances, namely those with $n \leq 5$ and, for these instances, also FP2 and FP3 can find it. For instance n5 and all the instances of the second set, COUENNE reaches the time limit providing a feasible solution (column 9) and a lower bound (last column). However, FP2 and FP3 can always find a better solution in these cases. Finally, for the remaining instances, i.e., those of the first set with $n \geq 6$, COUENNE cannot find a feasible solution within the time limit, while the 3 versions of FP always do. The feasible solutions with the best value are always found by FP3, the only exception being instance n7.

In Table 3 we report, for each instance and for each of the 3 versions of FP and COUENNE: (i) time to find the first solution; (ii) time to find the best solution; (iii) time to stop. We note that the 3 versions of FP show CPU times of comparable order of magnitude. As expected, as the dimension and the complexity of the instances grow, the CPU time grows significantly. However, excluding the smallest instances, FPs still show reasonable performances for small to medium size instances compared to C&D, that, on the contrary, reaches the time limit for almost all the instances. Moreover, we remark that the CPU times needed by FPs to find the first feasible solution are between roughly 0 and 1.2 CPU seconds. This makes the approach viable from the application perspective.

Table 4 reports the number of iterations of FP needed to find the best solution. The number of iterations of the best FP version, namely FP3, is always very low (it never exceeds 5 iterations). In the table, we do not report the number of iterations needed to find the first feasible solution because it is always ≤ 2 (the only exception being n7_c6 for FP1). Note also that the total number of iterations is always the one needed to find the best solution plus 1, where the extra iteration is needed to check that the new optimality cut makes subproblem P_1 infeasible, the only exception being instances n7_c6 and n10_c10 for FP1 for which the iteration limit (70 and 100, respectively) is reached.

Finally, we comment on the comparison of the best version of FP, namely FP3, and the heuristic algorithm presented in [10]. The heuristic is based on a decomposition into subproblems (clusters) and on local exact solutions of these subproblems. Note that the algorithm includes a modification of the current solution based on a random decision. We run 10 times the algorithm on the same machine used to test FP to fairly compare the two heuristics. The objective function corresponding to the best solution found by FP3 is always better than the best solution value found among the 10 runs of the heuristic presented in [10]. As for the CPU time, the heuristic reaches the time limit for 5 instances (nX with X $\geq$ 7 and n10_c10), thus FP3 is more robust as it can always find a solution. However, for the other instances, the CPU time needed for FP3 to converge is always worse than the best of the 10 runs of the heuristic and almost always worse than the average over the 10 runs. In particular, for instances nX, FP3 shows a CPU time of roughly the same order of magnitude of the heuristic. For instances nX_cY the CPU time of FP3 is roughly two orders of magnitude larger than the CPU time of the heuristic. However, we remark that the solution provided by the heuristic is the first feasible solution found. Comparing it to the first solution provided by FP3, we note that the CPU time of FP3 is always smaller than the one of the best run of the heuristic. FP3 remains better on the value of the objective function, the only exception being instance n6. We conclude that FP3 is a good compromise between efficiency and reliability.

Table 2 Objective function value of the first and the best solution found by each method.

instance	First solution				Best solution				LB
	FP1	FP2	FP3	C&D	FP1	FP2	FP3	C&D	C&D
n2	0.003143	**0.002531**	**0.002531**	**0.002531**	0.002583	**0.002531**	**0.002531**	**0.002531**	0.002531
n3	0.004410	0.003218	**0.001667**	0.003218	0.001744	**0.001667**	**0.001667**	**0.001667**	0.001667
n4	0.004915	**0.004029**	**0.004029**	**0.004029**	0.004044	**0.004029**	**0.004029**	**0.004029**	0.004029
n5	0.005072	**0.003056**	**0.003056**	0.003504	0.003089	**0.003056**	**0.003056**	0.003504	0.003046
n6	0.006848	**0.006682**	**0.006682**	–	0.006185	**0.006088**	**0.006088**	–	0.002614
n7	0.009224	**0.008010**	0.008270	–	0.009224	**0.008010**	0.008270	–	0.001302
n8	0.008853	0.008328	**0.008321**	–	0.008517	0.008328	**0.008321**	–	0.000265
n9	0.012223	0.010353	**0.009487**	–	0.012223	0.010353	**0.009487**	–	0.000223
n10	0.011801	0.013383	**0.011016**	–	0.011801	0.013383	**0.011016**	–	0.000091
n6_c5	0.007075	0.004383	**0.001295**	0.002549	0.001323	**0.001295**	**0.001295**	0.001327	0.000093
n7_c4	0.006777	**0.001617**	**0.001617**	0.007254	**0.001627**	**0.001617**	**0.001617**	0.001695	0.000015
n7_c6	–	0.001898	**0.001679**	0.002229	–	**0.001579**	**0.001579**	0.002153	0.000007
n8_c4	0.005160	**0.002384**	**0.002384**	0.002515	0.002410	**0.002384**	**0.002384**	0.002387	0.000528
n10_c10	0.006588	**0.001470**	**0.001470**	0.001543	0.005024	**0.001470**	**0.001470**	0.001543	0.000000

Table 3 CPU time to find the first and the best solution and to terminate.

instance	First solution				Best solution				Total			
	FP1	FP2	FP3	C&D	FP1	FP2	FP3	C&D	FP1	FP2	FP3	C&D
n2	**0.00**	**0.00**	**0.00**	**0.00**	0.02	0.01	0.01	**0.00**	0.66	**0.49**	0.50	0.08
n3	**0.00**	**0.00**	**0.00**	**0.00**	0.04	**0.01**	**0.01**	**0.01**	5.69	3.96	5.33	**0.99**
n4	**0.00**	**0.00**	**0.00**	**0.00**	0.03	0.01	0.01	**0.00**	5.39	5.29	**4.91**	8.28
n5	0.01	0.01	0.01	**0.00**	0.05	0.01	0.01	**0.00**	11.19	10.47	**10.10**	t.l.
n6	**0.01**	**0.01**	**0.01**	–	**0.04**	0.09	0.10	–	**25.59**	32.77	34.40	t.l.
n7	**0.37**	0.38	0.38	–	**0.37**	0.38	0.38	–	99.01	102.34	**98.84**	t.l.
n8	0.05	0.03	**0.01**	–	0.20	0.03	**0.01**	–	149.52	122.55	**118.82**	t.l.
n9	**0.84**	0.86	**0.84**	–	**0.84**	0.86	**0.84**	–	**368.49**	377.11	383.80	t.l.
n10	**1.17**	1.20	1.18	–	**1.17**	1.20	1.18	–	642.60	663.26	**639.99**	t.l.
n6_c5	**0.01**	**0.01**	**0.01**	8.57	0.16	0.02	**0.01**	75.75	**148.50**	166.13	177.85	t.l.
n7_c4	0.01	0.01	0.01	**0.00**	0.16	**0.01**	**0.01**	12.20	313.10	**200.13**	204.82	t.l.
n7_c6	–	**0.01**	**0.01**	507.03	–	0.03	**0.02**	3421.49	**3.69**	224.43	191.28	t.l.
n8_c4	0.01	0.01	0.01	**0.00**	0.08	**0.01**	**0.01**	52.73	210.82	**202.30**	203.75	t.l.
n10_c10	**0.01**	**0.01**	**0.01**	0.04	0.13	**0.02**	**0.02**	0.04	**7.72**	950.04	975.36	t.l.

Table 4 Number of iterations needed by each FP version to find the best solution.

Instance	FP1	FP2	FP3
n2	5	**2**	**2**
n3	11	3	**2**
n4	7	**2**	**2**
n5	11	**2**	**2**
n6	6	**4**	**4**
n7	**2**	**2**	**2**
n8	5	**2**	**2**
n9	**2**	**2**	**2**
n10	**2**	**2**	**2**
n6_c5	32	3	**2**
n7_c4	27	**2**	**2**
n7_c6	–	4	**3**
n8_c4	14	**2**	**2**
n10_c10	17	**2**	**2**

6 Conclusions

We have considered a very difficult problem arising in the aircraft traffic management context, namely the problem of aircraft deconfliction with speed regulation. We propose a tailored Feasibility Pump algorithm based on reformulations and relaxation of the original problem, that can be modeled as a mixed integer nonlinear programming problem. Its aim is to alternatively solve two easier subproblems represented by relaxations of the original problem by minimizing the distance of the solution with respect to the last solution found by solving the other subproblem. The two subproblems are designed so that a solution that is feasible for both of them is also feasible for the original problem. Once a first feasible solution is found, Feasibility Pump tries to improve it thanks to an optimality cut, added to the first subproblem. We compare computationally three different versions of Feasibility Pump with a global optimization solver, namely COUENNE, on the mathematical formulation proposed by Cafieri and Durand [10] on two sets of instances. The results show that Feasibility Pump algorithms clearly outperform COUENNE on almost all the instances, with the exception of the smallest instances. Moreover, Feasibility Pump can always find a feasible solution within 1.20 CPU second, making the approach very interesting to practically solve the real-world application.

In the future we would like to explore alternative exact methods with the aim of first finding quickly a feasible solution, then to provide a good lower bound in order to measure the quality of the solutions.

Acknowledgements The authors gratefully acknowledge the financial support by French National Research Agency (ANR) through Grant ANR 12-JS02-009-01 “ATOMIC”.

References

1. Alonso-Ayuso, A., Escudero, L.F., Martín-Campo, F.J.: Collision avoidance in air traffic management: a mixed-integer linear optimization approach. IEEE Trans. Intell. Transp. Syst. **12**(1), 47–57 (2011)

2. Alonso-Ayuso, A., Escudero, L.F., Martín-Campo, F.J.: A mixed 0–1 nonlinear optimization model and algorithmic approach for the collision avoidance in ATM: velocity changes through a time horizon. Comput. Opera. Res. **39**(12), 3136–3146 (2012)
3. Alonso-Ayuso, A., Escudero, L.F., Martín-Campo, F.J.: Exact and approximate solving of the aircraft collision resolution problem via turn changes. Transp. Sci. **50**(1), 263–274 (2015)
4. Belotti, P., Lee, J., Liberti, L., Margot, F., Wachter, A.: Branching and bounds tightening techniques for non-convex MINLP. Optim. Methods Softw. **24**(4–5), 597–634 (2009)
5. Bonami, P., Gonalves, J.P.M.: Heuristics for convex mixed integer nonlinear programs. Comput. Optim. Appl. **51**(2), 729–747 (2012)
6. Bonami, P., Cornuéjols, G., Lodi, A., Margot, F.: A feasibility pump for mixed integer nonlinear programs. Math. Program. **119**(2), 331–352 (2009)
7. Brochard, M.: Erasmus—en route air traffic soft management ultimate system. Technical report, Eurocontrol Experimental Centre
8. Cafieri, S.: Maximizing the number of solved aircraft conflicts through velocity regulation. In: MAGO 2014, 12th Global Optimization Workshop, pp. 1–4. Málaga, Spain (2014)
9. Cafieri, S.: MINLP in Air Traffic Management: Aircraft conflict avoidance. In: Terlaky, T., Anjos, M., Ahmed, S. (eds.) Advances and Trends in Optimization with Engineering Applications, pp. 293–301. MOS-SIAM Series on Optimization, SIAM, Philadelphia (2017)
10. Cafieri, S., Durand, N.: Aircraft deconfliction with speed regulation: new models from mixed-integer optimization. J. Global Optim. **58**(4), 613–629 (2014)
11. Cafieri, S., Omheni, R.: Mixed-integer nonlinear programming for aircraft conflict avoidance by sequentially applying velocity and heading angle changes. Eur. J. Oper. Res. **260**(1), 283–290 (2017)
12. Cafieri, S., Rey, D.: Maximizing the number of conflict-free aircraft using mixed-integer nonlinear programming. Comput. Oper. Res. **80**, 147–158 (2017)
13. D'Ambrosio, C., Frangioni, A., Liberti, L., Lodi, A.: Experiments with a Feasibility Pump Approach for Nonconvex MINLPs, pp. 350–360. Springer, Berlin (2010)
14. D'Ambrosio, C., Frangioni, A., Liberti, L., Lodi, A.: A storm of feasibility pumps for nonconvex MINLP. Math. Program. **136**(2), 375–402 (2012)
15. Fischetti, M., Glover, F., Lodi, A.: The feasibility pump. Math. Program. **104**(1), 91–104 (2005)
16. Fourer, R., Gay, D.M., Kernighan, B.W.: AMPL: A Modeling Language for Mathematical Programming, 2nd edn. Brooks/Cole, Boston (2002)
17. Liberti, L., Cafieri, S., Tarissan, F.: Reformulations in mathematical programming: a computational approach. Foundations of Computational Intelligence, vol. 3, pp. 153–234. Springer, Berlin (2009)
18. Pallottino, L., Feron, E.M., Bicchi, A.: Conflict resolution problems for air traffic management systems solved with mixed integer programming. IEEE Trans. Intell. Transp. Syst. **3**(1), 3–11 (2002)
19. Rey, D., Rapine, C., Fondacci, R., Faouzi, N.E.E.: Minimization of potential air conflicts through speed regulation. Transp. Res. Rec. J. Transp. Res. Board **2300**, 59–67 (2012)
20. Rey, D., Rapine, C., Fondacci, R., Faouzi, N.E.E.: Subliminal speed control in air traffic management: optimization and simulation. Transp. Sci **50**(1), 240–262 (2016)
21. Richards A, How J (2002) Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In: 2002 Proceedings of the American Control Conference, vol. 3, pp. 1936–1941
22. Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program. **106**(1), 25–57 (2006)