Polynomial programming prevents aircraft (and other) crashes

Martina Cerulli, Leo Liberti

Polynomial programming prevents aircraft (and other) crashes

Martina Cerulli[1], Leo Liberti[1]

[1] *LIX, École Polytechnique, Institut Polytechnique de Paris, F-91128 Palaiseau, France*
Email:{mcerulli,liberti}@lix.polytechnique.fr

August 24, 2020

Abstract

Using a known algebraic result, we obtain a finite (if nonconvex) polynomial programming reformulation of a semi-infinite program modelling the aircraft deconfliction problem via subliminal speed regulation. Solving the reformulation often yields better results than the state of the art.
Keywords: semi-infinite programming, quadratic programming, distance constraint.

1 Introduction

In air traffic management, the aircraft deconfliction problem (ADP) aims at ensuring the respect of a required distance among flying aircraft, while optimizing a certain objective function. Several strategies are used to pursue this goal: changing aircraft altitudes, trajectories, heading angle, or speed. All of them are mainly implemented by human air traffic controllers (ATC) who are in charge of detecting and solving potential conflicts among aircraft flying in a restricted airspace.

Nowadays it is increasingly interesting to introduce automation in aircraft deconfliction, as well as in urban air mobility. The speed regulation strategy, in particular, has been studied in the context of the European project ERASMUS [9], which introduced the concept of subliminal speed control. This consists in slightly modifying aircraft speed in an imperceptible way for ATC, but in such a way that the number of conflicts is reduced upstream of the control.

In this paper, we focus on ADP via subliminal speed regulation (SRADP) and formulate it using Semi-Infinite Programming (SIP). In order to address the issue of uncountably many constraints, we reformulate it using Polynomial Programming (PP).

The application scope of our new PP reformulation for SIPs extends to all application settings where distance constraints must be imposed at each of uncountably many time instants or space points on a curve. We offer two examples. Trajectories in a fleet of underwater autonomous vehicles cannot come exceedingly close [1] during the time horizon of the operation. In reservoir engineering the (ramified) geometry of the underground pipes must be decided in such a way that branches from different wells are positioned at any point of a given trajectory depending on the ramification structure, but not too close to each other [6].

The fact that such SIPs could be reformulated to PPs (either via Lasserre-type semidefinite relaxations [11] or kinetic distance matrices [8]) was previously known. Previous results, however, only offered relaxations, because of the large size and polynomial degree of the corresponding (nonconvex) formulations. In this paper, on the other hand, we provide a reasonably small PP having the same polynomial degree as the original SIP, which can be solved in practice to derive feasible solutions.

The rest of this paper is organized as follows. In Sect. 2 we introduce the SIP formulation of the SRADP. In Sect. 3 we propose our new PP reformulation of the SIP. In Sect 4 we present some computational results showing the practical applicability of our PP reformulation.

2 Semi-infinite formulation

In this section we formulate the SRADP using Mathematical Programming (MP). The decision variables quantify the speed changes. The objective function consists in minimizing the total speed changes. An uncountable set of constraints guarantee the safety distance on each pair of aircraft flying in the airspace considered during the given time horizon. The following natural formulation of the SRADP was already presented in [4].

The index sets, parameters and decision variables are involved in all of the formulations presented in this paper (not only the SIP one). We remark that they are the same as in [3].

- **Sets:**
  - $A = \{1, \ldots, i, \ldots, n\}$ is the set of aircraft flying in a shared airspace;
  - $K = \{1, \ldots, k_{\max}\}$ is the set of dimension indices.

- **Parameters:**
  - $[0, T]$ is the time horizon taken into account, with T expressed in hours;
  - d is the safety distance between aircraft [Nautical Miles NM, 1 NM = 1852 m];
  - x^0_{ik} is the k-th component of the initial position of aircraft i;
  - v_i is the initially planned speed of aircraft i [NM/h];
  - u_{ik} is the k-th component of the direction of aircraft i;
  - q_i^{min} and q_i^{max} define the feasible range of the speed modification ratios of aircraft i s.t. $q_i^{\mathsf{min}} < 1 < q_i^{\mathsf{max}}$.

- **Variables:** q_i is the ratio of the implemented speed to the initially planned speed of aircraft i: $q_i = 1$ if the speed is equal to the initially planned one, $q_i > 1$ if it is increased, $q_i < 1$ if it is decreased. We assume that q_i is constant in the time horizon considered, namely that each aircraft starts flying with the new implemented speed.

We now present objective function and constraints.

$$\min_{q^{\mathsf{min}} \le q \le q^{\mathsf{max}}} \quad \sum_{i \in A} (q_i - 1)^2 \tag{1a}$$

$$\forall i < j \in A,\ \forall t \in [0, T] \quad \sum_{k \in K} \left[(x^0_{ik} - x^0_{jk}) + t(q_i v_i u_{ik} - q_j v_j u_{jk}) \right]^2 \ge d^2. \tag{1b}$$

We remark that Eq. (1b) contains uncountably many constraints quantified over a continuous time symbol t. Eq. (1b) ensures aircraft separation requiring that the squared Euclidean distance between each pair of aircraft (i, j) to be greater than or equal to d^2 at each instant t in $[0, T]$.

2.1 Polishing the polynomial

For each $i < j \in A$, we define the polynomial:

$$p_{ij}(t) := \sum_{k \in K} \left[(x^0_{ik} - x^0_{jk}) + t(q_i v_i u_{ik} - q_j v_j u_{jk}) \right]^2 - d^2$$

in function of t. We have:

$$\begin{aligned}
p_{ij}(t) &= \sum_{k\in K}\left[(x^0_{ik}-x^0_{jk})+t(q_iv_iu_{ik}-q_jv_ju_{jk})\right]^2-d^2\\
&= \sum_{k\in K}[(x^0_{ik}-x^0_{jk})^2+t^2q_i^2(v_iu_{ik})^2+t^2q_j^2(v_ju_{jk})^2-2t^2(v_iu_{ik})(v_ju_{jk})q_iq_j\\
&\quad +2t(x^0_{ik}-x^0_{jk})(v_iu_{ik})q_i-2t(x^0_{ik}-x^0_{jk})(v_ju_{jk})q_j]-d^2\\
&= \sum_{k\in K}(x^0_{ik}-x^0_{jk})^2+t^2q_i^2\sum_{k\in K}(v_iu_{ik})^2+t^2q_j^2\sum_{k\in K}(v_ju_{jk})^2-2t^2q_iq_j\sum_{k\in K}(v_iu_{ik})(v_ju_{jk})\\
&\quad +2tq_i\sum_{k\in K}(x^0_{ik}-x^0_{jk})(v_iu_{ik})-2tq_j\sum_{k\in K}(x^0_{ik}-x^0_{jk})(v_ju_{jk})-d^2\\
&= (B_iq_i^2+B_jq_j^2-2C_{ij}q_iq_j)t^2+2(D^i_{ij}q_i-D^j_{ij}q_j)t+A_{ij}-d^2,
\end{aligned}$$

where $A_{ij}, B_i, B_j, C_{ij}, D^i_{ij}, D^j_{ij}$ are constant (w.r.t. t) defined as follows:

$$\begin{aligned}
A_{ij} &:= \sum_{k\in K}(x^0_{ik}-x^0_{jk})\\
B_i &:= \sum_{k\in K}(v_iu_{ik})^2\\
B_j &:= \sum_{k\in K}(v_ju_{jk})^2\\
C_{ij} &:= \sum_{k\in K}(v_iu_{ik})(v_ju_{jk})\\
D^i_{ij} &:= \sum_{k\in K}(x^0_{ik}-x^0_{jk})(v_iu_{ik})\\
D^j_{ij} &:= \sum_{k\in K}(x^0_{ik}-x^0_{jk})(v_ju_{jk}).
\end{aligned}$$

Thus,

$$p_{ij}(t)=(B_iq_i^2+B_jq_j^2-2C_{ij}q_iq_j)t^2+2(D^i_{ij}q_i-D^j_{ij}q_j)t+A_{ij}-d^2 \tag{2}$$

is a polynomial of second degree in t.

We can now rewrite the SIP formulation in (1a)–(1b) as:

$$\left.\begin{array}{rl} \min\limits_{q^{\min}\le q\le q^{\max}} & \sum\limits_{i\in A}(q_i-1)^2\\ \forall i<j\in A,\ \forall t\in[0,T] & p_{ij}(t)\ge 0.\end{array}\right\} \tag{3}$$

Eq. (3) is the minimization of $\sum_{i\in A}(q_i-1)^2$ subject to the second degree polynomial $p_{ij}(t)$ being non-negative on $t\in[0,T]$.

3 Reformulation to polynomial programming

We introduce now a reformulation of Eq. (3) based on a result from [12]. This allows us to obtain a (finite) PP problem of the same degree of the original formulation (3).

In particular, the following proposition is an immediate corollary of [12, Lemma 2.1].

3.1 Proposition
For any $i<j\in A$, the polynomial $p_{ij}(t)$ is non-negative on $[0,T]$ iff there is a 2×2 positive semidefinite matrix

$$M_{ij}=\begin{pmatrix} M^{11}_{ij} & M^{12}_{ij}\\ M^{21}_{ij} & M^{22}_{ij}\end{pmatrix}\succeq 0$$

(with $M^{12}_{ij} = M^{21}_{ij}$) and a nonnegative scalar $\mu_{ij} \geq 0$ such that:

$$p_{ij}(t) = (1\ t)M^{ij}\begin{pmatrix} 1 \\ t \end{pmatrix} + (T-t)t\mu_{ij}. \tag{4}$$

We use Proposition 3.1 to introduce an exact reformulation of the SIP in Eq. (3), as shown in Theorem 3.2.

3.2 Theorem
The following formulation:

$$\left.\begin{array}{rrcl} \min\limits_{M,\mu,q} & \sum\limits_{i\in A}(q_i-1)^2 & & \\ \forall i<j\in A & M^{22}_{ij}-\mu_{ij} & = & (B_iq_i^2+B_jq_j^2-2C_{ij}q_iq_j) \\ \forall i<j\in A & 2M^{12}_{ij}+T\mu_{ij} & = & 2(D^i_{ij}q_i-D^j_{ij}q_j) \\ \forall i<j\in A & M^{11}_{ij} & = & A_{ij}-d^2 \\ \forall i<j\in A & (M^{12}_{ij})^2 & \leq & M^{11}_{ij}M^{22}_{ij} \\ \forall i<j\in A & M^{11}_{ij}, M^{22}_{ij}, \mu_{ij} & \geq & 0 \\ & q^{\mathsf{min}} \leq & q & \leq q^{\mathsf{max}} \end{array}\right\} \tag{5}$$

is an exact reformulation of Eq. (1a)–(1b).

Proof. Note that $p_{ij}(t)$ is given in two different forms Eq. (2) and Eq. (4). We can therefore match coefficients of terms in t. This yields the following system:

$$\begin{array}{rrcl} \forall i<j\in A & M^{22}_{ij}-\mu_{ij} & = & (B_iq_i^2+B_jq_j^2-2C_{ij}q_iq_j) \\ \forall i<j\in A & 2M^{12}_{ij}+T\mu_{ij} & = & 2(D^i_{ij}q_i-D^j_{ij}q_j) \\ \forall i<j\in A & M^{11}_{ij} & = & A_{ij}-d^2, \end{array}$$

which is independent of t by construction. We now have to impose the constraints $M^{ij} \succeq 0$ and $\mu_{ij} \geq 0$ given in the statement of Prop. 3.1. For the former, we observe that a 2×2 matrix Γ is positive semidefinite iff

$$(\Gamma_{12})^2 \leq \Gamma_{11}\Gamma_{22}$$

and $\Gamma_{11}, \Gamma_{22} \geq 0$, which yields the corresponding constraints in Eq. (5). The latter is simply copied from Eq. (3) to Eq. (5). □

We observe that Eq. (5) is a quadratic PP problem, and note that the degree is the same as in the original formulation Eq. (1a)–(1b). We also observe that Eq. (5) is nonconvex in q because of the constraints

$$\forall i<j\in A \quad M^{ij}_{22}-\mu_{ij} = (B_iq_i^2+B_jq_j^2-2C_{ij}q_iq_j) = \sum_{k\in K}(v_iu_{ik}q_i - v_ju_{jk}q_j)^2. \tag{6}$$

We remark that a convex relaxation can be readily obtained by relaxing Eq. (6) to

$$\forall i<j\in A \quad M^{ij}_{22}-\mu_{ij} \geq (B_iq_i^2+B_jq_j^2-2C_{ij}q_iq_j). \tag{7}$$

4 Computational results

We tested our new MP formulation in Eq. (5) of the SRADP in k dimensions on some 2D and 3D instances.

The set of 2D instances, already used in [3], is taken from [2]. It consists of *circle instances* where n aircraft are placed on a circle of a given radius r, and *non-circle instances* where aircraft move along straight trajectories intersecting in n_c conflict points.

The set of 3D instances is introduced in [4] and it includes both *sphere instances*, where n aircraft are placed on a sphere of a given radius r, and instances in which aircraft move along straight 3D trajectories (named *non-sphere instances* in Tab 1), which intersect in at least $\frac{n}{2}$ conflict points.

For the 2D and the sphere instances the planned speed is $v_i = 400\,\text{NM/h}$ for each $i \in A$ and parameters x^0_{ik} and u_{ik} are given by

$$u_{i1} = \cos(\phi_i)\sin(\gamma_i), \quad u_{i2} = \sin(\phi_i)\sin(\gamma_i), \quad u_{i3} = \cos(\gamma_i), \quad x^0_{ik} = -r\,u_{ik},$$

where γ_i is the angle that the vector of the direction u_i forms with the axis k_3 and ϕ_i is the angle between the projection of u_i onto the k_1k_2-plane and the axis k_1. The bounds q_i^{min} and q_i^{max} are set to 0.94 and 1.03 respectively, following the weaker bounds proposed by the ERASMUS project [9].

We implemented the PP formulation (5) using the AMPL modeling language [5] and solved it with the global optimization solver Baron [7] (B in the Table 1). For cases in which Baron exceeded its time-limit (set to 3600 seconds), we used a Multistart algorithm (MS in Table 1), which performs 1000 calls to the IPOPT [10] local NLP solver from randomly sampled starting points.

In all our tests, we considered a time horizon of $T = 2$ hours and safety distance $d = 5\,\text{NM}$. All the solvers were run with their default settings. The tests were performed on a 2.53GHz Intel(R) Xeon(R) CPU with 48 GB RAM.

It is clear that the proposed formulation often improves the best available objective function value for the instance, and is therefore useful.

Table 1: Numerical results

Instances		Literature	SOS		
n	*r*	*Best obj*	*obj*	*time(s)*	*solver*
Circle					
2	100	0.002531	0.002531	0.58	B
3	200	0.001667	**0.001666**	0.68	B
4	200	**0.004009**	0.004028	198	B
5	300	**0.003033**	0.003056	46.1	MS
6	300	**0.006033**	0.006058	962	MS
Non-circle					
6	-	0.001295	**0.001254**	53.7	MS
7	-	0.001617	**0.001591**	72.4	MS
7	-	0.001579	**0.001566**	72.7	MS
8	-	**0.002384**	**0.002384**	83.5	MS
10	-	0.001470	**0.001397**	139	MS
Spheric					
2	100	**0.002220**	0.002227	0.84	B
3	200	**0.001404**	0.001407	1.12	B
4	200	**0.003703**	0.003714	35.3	MS
5	300	**0.002959**	**0.002959**	47.8	MS
6	300	**0.005847**	**0.005847**	66.3	MS
7	500	**0.002855**	0.002856	80.7	MS
8	500	0.004549	**0.004513**	104	MS
9	500	**0.006987**	**0.006987**	127	MS
10	600	0.006410	**0.006393**	161	MS
12	700	0.008404	**0.008380**	222	MS
Non-spheric					
2	-	**0.000305**	**0.000305**	0.09	B
4	-	**0.003278**	0.003282	3.00	B
6	-	**0.006003**	0.006004	52.1	MS
8	-	**0.011704**	0.011706	85.4	MS
10	-	**0.001503**	0.01503	203	MS

References

[1] A. Bahr, J. Leonard, and M. Fallon. Cooperative localization for autonomous underwater vehicles. *International Journal of Robotics Research*, 28(6):714–728, 2009.

[2] S. Cafieri and C. D'Ambrosio. Feasibility pump for aircraft deconfliction with speed regulation. *Journal of Global Optimization, Springer Verlag*, 71(3):501–515, 2018.

[3] M. Cerulli, C. D'Ambrosio, and L. Liberti. Flying safely by bilevel programming. *Advances in Optimization and Decision Science for Society, Services and Enterprises: ODS, Genoa, Italy, September 4-7, 2019*, 3:197–206, 2019. Available at `https://hal.archives-ouvertes.fr/hal-02869682`.

[4] M. Cerulli, C. D'Ambrosio, L. Liberti, and M. Pelegrín. Detecting and solving aircraft conflicts using bilevel programming. *Working paper*, 2020. Available at `https://hal.archives-ouvertes.fr/hal-02869699`.

[5] R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Cengage Learning, 2002.

[6] C. Lizon, C. D'Ambrosio, L. Liberti, M. Le Ravalec, and D. Sinoquet. A mixed-integer nonlinear optimization approach for well placement and geometry. In *Proceedings of the 14th European Conference on the Mathematics of Oil Recovery*, volume XIV of *ECMOR*, page A38, Houten, 2014. EAGE.

[7] N. V. Sahinidis. *BARON 17.8.9: Global Optimization of Mixed-Integer Nonlinear Programs,* User's Manual, 2017.

[8] P. Tabaghi, I. Dokmanić, and M. Vetterli. Kinetic Euclidean distance matrices. *IEEE Transactions on Signal Processing*, 68:452–465, 2020.

[9] J. Villiers. Automatisation du contrôle de la circulation aèrienne: "ERASMUS", une voie conviviale pour franchir le mur de la capacitè. *Institut du Transport Aèrien*, 58, 2004.

[10] A. Wächter and L. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:25–57, 2006.

[11] L. Wang and F. Guo. Semidefinite relaxations for semi-infinite polynomial programming. *Computational Optimization and Applications*, 58:133–159, 2014.

[12] Y. Xu, W. Sun, and L. Qi. On solving a class of linear semi-infinite programming by SDP method. *Optimization*, 64(3):603–616, 2015.