Polynomial programming prevents aircraft (and other) conflicts[☆]

Martina Cerulli, Leo Liberti

LIX - CNRS, École Polytechnique, Institut Polytechnique de Paris, Palaiseau, 91120, France

Abstract

Using a known algebraic result, we obtain a finite polynomial programming reformulation of a semi-infinite program modeling the aircraft deconfliction problem via subliminal speed regulation. Solving the reformulation yields better results than the state of the art for most of the tested instances.

Keywords: semi-infinite programming, quadratic programming, aircraft deconfliction, 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 available to pursue this goal: changing aircraft altitudes, heading angles, 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.

Given the pandemic situation, the air traffic system is currently not too congested. Therefore, this may be the best time to test and introduce new types of automation mechanisms in aircraft deconfliction, which would be useful in urban air mobility too. The speed regulation (SR) strategy has been studied in the context of the European project ERASMUS [1], 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, thus reducing ATC's workload.

In this paper, we focus on ADP via subliminal SR 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).

SR is one of the most used strategies for aircraft deconfliction by Mathematical Programming. In Section 4, we compare our approach with the one proposed in [2], where a Mixed Integer Nonlinear Programming (MINLP) formulation for ADP via SR in 2 dimensions is considered, and small instances of the problem are solved to global optimality by means of an existing exact solver, while for bigger instances a heuristic procedure (where the problem is decomposed and locally exactly solved) is proposed. In [3] another MINLP formulation of the same problem is proposed and solved using a feasibility pump heuristic. This algorithm, at each iteration, solves alternatively two subproblems obtained by two relaxations of the original problem, minimizing the distance between their solutions. In [4], two Mixed Integer Linear Programming formulations are presented: one using only speed changes, and the other one heading angles changes (HAC). Both [5] and [6] focus only on this last strategy. In particular, in [5] trajectories are modeled with B-splines and a SIP formulation of ADP via HAC is presented, reformulated with an exact penalty function, and solved using local optimization methods. A two-step approach is presented in [6]. The first step consists of a nonconvex MINLP model based on geometric constructions, which aims to minimize the total HAC cost (potentially the angle variation of each aircraft has a cost, even if in the experiments reported in the paper such costs are set to 1) to obtain the new conflict-free flight configuration. The second step consists of a set of unconstrained quadratic optimization models solved as a post-processing step to return each aircraft to its original flight plan as soon as possible after conflict resolution. In fact, SR strategy cannot solve face-to-face conflicts, and it may not be enough to guarantee safety if speed bounds are tight. That is why several works in the literature focus on combined maneuvers too. For instance, in [7] a complex number formulation with disjunctive linear constraints is introduced to model ADP using both SR and HAC; different relaxations of the resulting MINLP are then proposed, solved, and compared.

As already said, we formulate the ADP via SR as a SIP problem having an infinite number of quadratically parametrized (w.r.t the time) constraints. For an introduction to SIP, see, for instance, [8]. The classical discretization approach [9] to solve SIP problems consists in replacing the infinite constraint parameter set by a finite subset. Such discretization methods lead to a relaxation of the original problem, the optimal value of which converges to the SIP value as the subset approximates better the original feasible set. Instead of using a fixed subset of constraints, another approach is the cutting plane method [10], which consists in iteratively adding constraints, defining

[☆]This work was partly funded by the European Unions Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement n. 764759 ETN "MINOA".

Email addresses: `mcerulli@lix.polytechnique.fr` (Martina Cerulli), `liberti@lix.polytechnique.fr` (Leo Liberti)

the original feasible set.

The application scope of our new PP reformulation for the SIP model of ADP extends to all application settings where distance constraints must be imposed at each of uncountably many time instants. We offer three examples. Trajectories in a fleet of underwater autonomous vehicles cannot come exceedingly close [11] 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 [12]. When we focus on a three-dimensional space, our approach is more relevant in the context of urban air mobility. In fact, Unmanned Aerial Vehicles (UAVs) have different dynamics w.r.t aircraft and the minimum distance between them can be ensured by using SR also during altitude changes (therefore the need of a third dimension to take into account). Considering more than three dimensions may be used for modelling, for example, fleets of objects with pairwise distance.

The fact that such SIP problems could be reformulated to PP ones (either via Lasserre-type semidefinite relaxations [13] or kinetic distance matrices [14]) 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 formulation having the same polynomial degree as the original SIP problem, which can be solved in practice to derive feasible solutions.

The rest of this paper is organized as follows. In Section 2 we introduce the SIP formulation of the ADP via SR. In Section 3 we propose our new PP reformulation of the SIP problem. In Section 4 we present some computational results showing the practical applicability of our PP reformulation. Some concluding comments are given in Section 5.

2. Semi-infinite formulation

In this section we formulate the ADP via SR using Mathematical Programming. 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 ADP was already presented in [15].

The following 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 taken from [2].

Sets:

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

Parameters:

- $[0, T]$ is the time interval 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. $q_i^{\min} < 1 < q_i^{\max}$.

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 interval considered, namely that each aircraft starts flying with the new implemented speed.

We now present objective function and constraints.

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

$$\forall i \in A \quad q_i^{\min} \le q_i \le q_i^{\max} \tag{1b}$$

$$\forall i < j \in A, \forall t \in [0, T]$$

$$\sum_{k \in K} \left[\left(x^0_{ik} - x^0_{jk} \right) + t \left(q_i v_i u_{ik} - q_j v_j u_{jk} \right) \right]^2 \ge d^2. \tag{1c}$$

We remark that (1c) contains uncountably many constraints quantified over t. For this reason formulation (1a)–(1c) is a SIP program. Constraints (1c) ensure aircraft separation requiring 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 $[0, T]$. Constraints (1b) ensure that variable q_i is in $[q_i^{\min}, q_i^{\max}]$ for every aircraft i.

2.1. Polishing the polynomial

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

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

in function of t. We have:

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

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

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

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 (1a)–(1c) as:

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

Problem (3) is the minimization of $\sum\limits_{i\in A}(q_i-1)^2$ subject to the second degree polynomial $p_{ij}(t)$ being non-negative on $t\in[0,T]$, and q_i being in $[q_i^{\min}, q_i^{\max}]$ for each aircraft i.

3. Reformulation to polynomial programming

We introduce now a reformulation of (3) based on a result from [16]. 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 [16, Lemma 2.1].

Proposition 1 (corollary of Lemma 2.1 from [16]) *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*

$$M_{ij}=\begin{pmatrix} m_{ij} & r_{ij}\\ r_{ij} & g_{ij}\end{pmatrix}\ge 0$$

and a nonnegative scalar $\mu_{ij}\ge 0$ 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 1 to introduce an exact reformulation of the SIP problem (3), as shown in Theorem 2.

Theorem 2 *The following formulation:*

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

is an exact reformulation of (1a)–(1c).

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

$$\begin{array}{rcll}
g_{ij}-\mu_{ij} & = & B_iq_i^2+B_jq_j^2-2C_{ij}q_iq_j & \forall i<j\in A\\
2r_{ij}+T\mu_{ij} & = & 2(D^i_{ij}q_i-D^j_{ij}q_j) & \forall i<j\in A\\
m_{ij} & = & A_{ij}-d^2 & \forall i<j\in A,
\end{array}$$

which is independent of t by construction. We now have to impose the constraints $M_{ij}\ge 0$ and $\mu_{ij}\ge 0$ given in the statement of Prop. 1. For the former, we observe that the 2×2 matrix M_{ij} is positive semidefinite iff $(r_{ij})^2\le m_{ij}g_{ij}$ and $m_{ij}, g_{ij}\ge 0$, which yields the corresponding constraints in formulation (5). The latter is simply copied from formulation (3) to (5). □

We observe that problem (5) is a quadratic PP problem, and the degree is the same as in the original formulation (1a)–(1c). We also observe that formulation (5) is nonconvex in q because of the constraints $\forall i<j\in A$

$$g_{ij}-\mu_{ij}=B_iq_i^2+B_jq_j^2-2C_{ij}q_iq_j=\sum_{k\in K}(v_iu_{ik}q_i-v_ju_{jk}q_j)^2. \tag{6}$$

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

$$g_{ij}-\mu_{ij}\ge B_iq_i^2+B_jq_j^2-2C_{ij}q_iq_j \quad \forall i<j\in A. \tag{7}$$

4. Computational results

We tested the SIP formulation (1a)–(1c) using a native solver [17] which implements three types of discretization methods, but they could not solve most of the non-trivial instances ($n>2$). We report here, the results obtained testing our new formulation (5) of the ADP in k dimensions on some 2D and 3D instances.

The set of 2D instances, already used in [18], 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 [15] – see the public repository `https://github.com/MartinaCerulli/SRADP` – 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 Table 1), which intersect in at least $\frac{n}{2}$ conflict points. Such 3D instances, as already said, are more relevant for UAVs, which have different dynamics w.r.t aircraft and can modify their speed also while changing altitudes, always having a relative Euclidean distance greater than a safety threshold.

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

$$u_{i1} = \cos(\phi_i)\sin(\gamma_i), \quad u_{i2} = \sin(\phi_i)\sin(\gamma_i),$$

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

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 [1].

We implement the PP formulation (5) using the AMPL modeling language [19] and solve it with the global optimization solver Gurobi [20]. For cases in which Gurobi exceeds its time limit (t.l. in Table 1) set to 36000 seconds, we use a multistart algorithm (MS in Table 1), which performs 1000 calls to the IPOPT [21] local non-linear programming (NLP) solver from randomly sampled starting points.

In Table 1, for 2D instances we compare the results obtained by testing our PP reformulation on 2D instances with those that are obtained by solving the MINLP formulation proposed in [2] with the global solver Couenne [22] — Gurobi can't handle nonquadratic nonlinear constraints —, with default options but again with a time limit of 36000 seconds. Whenever this time limit is reached, we report in italics lower and upper bound (*inf* when no feasible solution is found within the time limit) computed by Couenne.

The benchmarks for 3D instances are taken from the implementation of the approaches proposed in [15], where the ADP is formulated using bilevel programming and then reformulated into a single level non linear problem in three different ways. We report in the second column of Table 1 the best results among those obtained by solving the three NLP reformulations proposed in [15] with Couenne — Gurobi can't handle nonquadratic nonlinear constraints —, when we use Gurobi for solving formulation (5), or with the multistart algorithm when we use it for formulation (5). Again, for the global solver Couenne we set a time limit of 36000 seconds, which is the same that we set for Gurobi. Whenever this time limit is reached, we report in italics lower and upper bound (*inf* when no feasible solution is found within the time limit) computed by the solver.

The headings of Table 1 are the following: n number of aircraft; r radius in NM of the circle or the sphere for 2D and 3D instances respectively; Literature "Best*LB–UB*" (column 3) is either the best optimal value found, or, when the time limit is exceeded, lower and upper bound of such optimal value; Formulation (5) "obj" is the objective function value returned by Gurobi or by the multistart algorithm;"time(s)" computing time in seconds; "solver" solver used.

In all our tests, run on NEOS Server [23, 24, 25], we consider a time interval of $T = 2$ hours and safety distance $d = 5\,\mathrm{NM}$.

The value of the objective function is always very low, given the tight speed variation bounds imposed by ERASMUS project ($q \in [q^{\min}, q^{\max}]$). Best values are reported in bold for each instance. We can note that the formulation (5) is the best for most of the instances in terms of both solution quality and efficiency.

In particular, for 2D instances Couenne can find the global optimum only for small instances, namely those with $n \leq 5$; for the larger circle instances, it reaches the time limit providing only a lower bound (*LB*), while for the non-circle ones, it provides a feasible solution too. On the contrary, with our approach we can always find an optimal solution within the time limit.

For 3D instances, when Couenne and Gurobi are used to solve NLP models proposed in [15] and formulation (5) respectively, Gurobi always outperforms Couenne, which cannot even find the global optimum for larger non-spheric instances. When the multistart algorithm is used, for half of the instances the value found with our approach is the best, but we have no proof of global optimality.

5. Conclusion

We address the aircraft deconfliction problem in k dimensions via speed regulation. We model it as a SIP program, which we reformulate using polynomial programming, and a theorem of Y. Xu et al. presented in [16]. The nonconvex polynomial programming problem obtained has the same degree of the original formulation. We solve it using state-of-the-art solvers, which provide better solutions compared with other approaches in the literature for most of the tested instances.

References

[1] ERASMUS Consortium, En route air traffic soft management ultimate system, Tech. Rep. proj. TREN/06/FP6AE/S07, deliv. D4.6, European Commission (2009).

[2] S. Cafieri, N. Durand, Aircraft deconfliction with speed regulation: New models from mixed-integer optimization, J. of Global Optimization 58 (2014) 613–629.

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

[4] L. Pallottino, E. Feron, A. Bicchi, Conflict resolution problems for air traffic management systems solved with mixed integer programming, IEEE transactions on intelligent transportation systems 3 (1) (2002) 3–11.

[5] C. Peyronne, A. R. Conn, M. Mongeau, D. Delahaye, Solving air traffic conflict problems via local continuous optimization, European Journal of Operational Research 241 (2) (2015) 502 – 512.

Table 1: Numerical results

Instances		Literature			Formulation (5)		
n	*r*	Best obj/*LB–UB*	time(s)	solver	obj	time(s)	solver
Circle							
2	100	**0.002531**	0.09	Couenne	**0.002531**	**0.07**	Gurobi
3	200	0.001667	1.15	Couenne	**0.001663**	**0.70**	Gurobi
4	200	**0.004019**	237	Couenne	0.004021	**19.0**	Gurobi
5	300	**0.003047**	20698	Couenne	0.003049	**46.1**	Gurobi
6	300	*0.005463 – inf*	t.l.	Couenne	**0.006042**	**87.4**	Gurobi
8	400	*0.000479 – inf*	t.l.	Couenne	**0.008247**	**34500**	Gurobi
Non-circle							
6	-	*0.000521 – 0.001254*	t.l.	Couenne	**0.001251**	**12.0**	Gurobi
7	-	*0.000023 – 0.001617*	t.l.	Couenne	**0.001589**	**18.6**	Gurobi
7	-	*0.000009 – 0.002147*	t.l.	Couenne	**0.001565**	**24.0**	Gurobi
8	-	*0.002372 – 0.002384*	t.l.	Couenne	**0.002381**	**444**	Gurobi
10	-	*0.000000 – 0.001543*	t.l.	Couenne	**0.001395**	**1472**	Gurobi
Spheric							
2	100	0.002227	0.16	Couenne	**0.002226**	**0.09**	Gurobi
3	200	0.001408	11.4	Couenne	**0.001405**	**0.98**	Gurobi
4	200	0.003714	119	Couenne	**0.003708**	**2.30**	Gurobi
5	300	0.002976	4890	Couenne	**0.002943**	**51.1**	Gurobi
6	300	**0.005320**	67.5	MS	0.005847	**59.1**	MS
7	500	0.002857	**161**	MS	**0.002855**	209	MS
8	500	0.004566	672	MS	**0.004513**	**104**	MS
9	500	**0.006457**	**91.0**	MS	0.006987	127	MS
10	600	**0.006333**	443	MS	0.006393	**161**	MS
12	700	0.008448	1727	MS	**0.008380**	**222**	MS
Non-spheric							
2	-	0.000305	**0.16**	Couenne	**0.000304**	0.33	Gurobi
4	-	0.003283	188	Couenne	**0.003282**	**1.72**	Gurobi
6	-	0.006004	6291	Couenne	**0.006002**	**4.40**	Gurobi
8	-	*0.000258 – 0.011705*	t.l.	Couenne	**0.011703**	**3.91**	Gurobi
10	-	*0.000838 – 0.015025*	t.l.	Couenne	**0.015022**	**13.4**	Gurobi

[6] A. Alonso-Ayuso, L. F. Escudero, F. J. Martn-Campo, Exact and approximate solving of the aircraft collision resolution problem via turn changes, Transportation Science 50 (2016) 263–274.

[7] D. Rey, H. Hijazi, Complex number formulation and convex relaxations for aircraft conflict resolution, IEEE 56th Annual Conference on Decision and Control (2017) 88 – 93.

[8] R. Reemtsen, J. Rückmann (Eds.), Semi-Infinite Programming, Vol. 25, Springer, Boston, MA, 1998.

[9] R. Hettich, An implementation of a discretization method for semi-infinite programming, Mathematical Programming 34 (3) (1986) 354–361.

[10] J. Kelley, Jr., The cutting-plane method for solving convex programs, Journal of the society for Industrial and Applied Mathematics 8 (4) (1960) 703–712.

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

[12] C. Lizon, C. D'Ambrosio, L. Liberti, M. L. Ravalec, 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, Vol. XIV of ECMOR, EAGE, Houten, 2014, p. A38.

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

[14] P. Tabaghi, I. Dokmanić, M. Vetterli, Kinetic Euclidean distance matrices, IEEE Transactions on Signal Processing 68 (2020) 452–465.

[15] M. Cerulli, C. D'Ambrosio, L. Liberti, M. Pelegrín, Detecting and solving aircraft conflicts using bilevel programming, Journal of Global Optimization Advance online publication.

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

[17] A. Vaz, E. Fernandes, M. Gomes, Nsips version 2.1 nonlinear semi-infinite programming solver, Tech. Rep. ALG/IV/5-2004, Universidade do Minho, Braga, Portugal (2004).

[18] M. Cerulli, C. D'Ambrosio, L. Liberti, Flying safely by bilevel programming, in: M. Paolucci, A. Sciomachen, P. Uberti (Eds.), Advances in Optimization and Decision Science for Society, Services and Enterprises (AIRO-ODS19), Springer, Cham, 2019, pp. 197–206.

[19] R. Fourer, D. M. Gay, B. W. Kernighan, AMPL: A Modeling Language for Mathematical Programming, Cengage Learning, Boston, MA, 2002.

[20] L. Gurobi Optimization, Gurobi optimizer reference manual (2020).

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

[22] P. Belotti, J. Lee, L. Liberti, A. Wächter, Branching and bounds tightening techniques for non-convex minlp, Optimization Methods and Software 24 (4-5) (2009) 597–634.

[23] J. Czyzyk, M. P. Mesnier, J. J. Moré, The NEOS server, IEEE Journal on Computational Science and Engineering 5 (3) (1998) 68 –75.

[24] E. D. Dolan, The NEOS server 4.0 administrative guide, Technical Memorandum ANL/MCS-TM-250, Mathematics and Computer Science Division, Argonne National Laboratory (2001).

[25] W. Gropp, J. J. Moré, Optimization environments and the NEOS server, in: Approximation Theory and Optimization, Cambridge University Press, 1997, pp. 167 – 182.