A comparison between MILP and NLP Techniques for Centralized Trajectory Planning of Multiple Unmanned Air Vehicles

Francesco Borrelli\textsuperscript{0,2}, Dharmashankar Subramanian\textsuperscript{2}, Arvind U. Raghunathan\textsuperscript{1} and Lorenz T. Biegler\textsuperscript{1}

Abstract—We consider the problem of optimal cooperative three-dimensional conflict resolution involving multiple Unmanned Air Vehicles (UAVs) using numerical trajectory optimization methods. The conflict problem is posed as an optimal control problem of finding trajectories that minimize a certain objective function while maintaining the safe separation between each UAV pair. We assume the origin and destination of the UAV are known and consider UAV models with simplified linear kinematics.

The main objective of this report is to compare two different approaches to the solution of the problem. In the first approach, the optimal control is converted to a finite dimensional Nonlinear Program (NLP) by using collocation on finite elements and by reformulating the disjunctions involved in modeling the protected zones by using continuous variables. In the second approach the optimal control is converted to a finite dimensional Mixed Integer Linear Program (MILP) using Euler discretization and reformulating the disjunctions involved with the protected zones by using binary variables and Big-M techniques. Based on results of extensive random simulations, we compare time complexity and optimality of the solutions obtained with the MILP approach and the NLP approach.

NLPs are essential to enforce flyability constraints on more detailed UAV models. The main objective of this paper is to open the route to the use of MILP solutions (based on simple linear UAV models) in order to initialize NLP solvers which allow the use of dynamic UAV models at any desired level of detail.

I. INTRODUCTION

The approaches to aircraft control for conflict resolution can differ on various criteria and objectives. A good review can be found in [20], [12], [13]. Our interest in resolution of conflicts between Unmanned Air Vehicles (UAVs) in free flight stems from the centralized computation of flyable trajectories for a flock of UAVs in an urban terrain.

In this paper we consider the problem of optimal cooperative three-dimensional conflict resolution involving multiple UAVs using mathematical programming techniques. We formulate a continuous-time optimal control problem where we minimize the inputs to a set of UAVs in order to let them move from a known initial configuration to a known final configuration without colliding with each other and without colliding with obstacles [14], [15], [16], [6]. UAVs are modeled as simple linear point-mass models with constraints on states and inputs. The protection zones associated with each UAV pair and each UAV-obstacle pair are modeled using disjunctions. The main objective of this report is to compare two different approaches to the numerical solution of the optimal control problem: the Nonlinear programming approach presented in [20] and the Mixed Integer Linear programming approach presented in [18]. We do not make any attempt to improve the computational time of the respective mathematical programming solvers with formulation-specific heuristics. Such a study is beyond the scope of this paper.

There are three key steps required for the numerical solution of the optimal control problem: (i) the time discretization of the continuous-time optimal control problem, (ii) the formulation of the disjunctions involved in modeling protection zones associated with each vehicle pair and vehicle-obstacle pair, and (iii) the choice of the mathematical programming solver used.

In the nonlinear programming approach [20] we employ collocation on finite elements to discretize the optimal control problem in time. In particular, the profiles of the optimization variables are approximated by a family of polynomials on finite elements. The disjunctions involved in modeling the protection zones are reformulated by using continuous variables and non-convex constraints. The resulting model is a nonconvex finite dimensional nonlinear program (NLP) which is solved by using an Interior Point algorithm that incorporates a novel line search method [20], [8]. The interior point algorithm is initialized as proposed in [7].

In the mixed integer programming approach [18] we
employ a simple Euler discretization of the UAV dynamics. The disjunctions involved in modeling the protection zones are reformulated by using binary variables and big-M techniques. The resulting model is a finite dimensional Mixed-Integer Linear Program (MILP) which is solved by using a commercial branch and bound algorithm included in the CPLEX [17] package.

Based on results of extensive random simulations, we compare time complexity and optimality of the solution obtained with MILP approach and the NLP approach. The results of the comparison are summarized in Section VIII. The comparative study reported in this paper has several objectives.

NLPs are essential to enforce flyability constraints on more detailed UAV models. The first objective of this paper is to open the route to the use of MILP solutions (based on simple linear UAV models) in order to initialize NLP solvers which allow the use of dynamic UAV models at any desired level of detail.

The second objective of the study is to test the performance of the NLP solver presented in [20] in terms of optimality of the solution. As a last objective, we study the relation between time complexity of the optimal solution (both MILP and NLP) and some parameters of the optimization problem such as density of the obstacle field or the number of collisions when the vehicles fly along straight lines joining initial and final destination, at any speed.

The paper is organized as follows. In Section II we formulate the continuous time optimal control problem that applies to conflict resolution among multiple UAVs. In Sections III and IV we describe the NLP and the MILP approach, respectively, which have been used to numerically solve the optimal control problem. Section VI describes our method for generating random problem instances. Finally we conclude in Section VII with the results of the comparative study.

II. Problem Formulation

In this section we describe the optimal control problem that applies to conflict resolution among multiple UAVs. We consider the conflict resolution of UAVs with dynamics described by a simple kinematic model

\[
\begin{align*}
\frac{dx}{dt} &= v_x(t) \\
\frac{dy}{dt} &= v_y(t) \\
\frac{dz}{dt} &= v_z(t)
\end{align*}
\]

where \(x, y\) and \(z\) represent the position of the UAV along the different axes. The velocities of the UAV in the \(x, y\) and \(z\) directions at time \(t\) are respectively \(v_x(t), v_y(t)\) and \(v_z(t)\). The maneuvers available to the UAV for conflict avoidance are changes in these velocities. The optimal conflict free trajectories are obtained as a result of minimizing the following objective,

\[
\min \sum_{i=1}^{n} J_i(v_{x,i}(t_0), v_{y,i}(t_0), v_{z,i}(t_f))
\]

subject to

\[
\frac{dx_i}{dt} = v_{x,i}(t); \quad x_i(t_0) = x_{i,0}; \quad x_i(t_f) = x_{i,f}
\]

\[
\frac{dy_i}{dt} = v_{y,i}(t); \quad y_i(t_0) = y_{i,0}; \quad y_i(t_f) = y_{i,f}
\]

\[
\frac{dz_i}{dt} = v_{z,i}(t); \quad z_i(t_0) = z_{i,0}; \quad z_i(t_f) = z_{i,f}
\]

\(i = 1, \ldots, n\)

\(\forall 1 \leq i < j \leq n\)

\(g_{\text{ineq}}(x_i(t), y_i(t), z_i(t), x_j(t), y_j(t), z_j(t)) \leq 0\)

\(v_{x,i}(t)^L \leq v_{x,i}(t) \leq v_{x,i}(t)^U\)

\(v_{x,i}(t)^L \leq v_{y,i}(t) \leq v_{x,i}(t)^U\)

\(v_{y,i}(t)^L \leq v_{z,i}(t) \leq v_{y,i}(t)^U\)

\(i = 1, \ldots, n\)

where \(i\) indexes the UAV involved, \(t_o\) is the initial time, \(t_f\) is the final time, \((x_{i,0}, y_{i,0}, z_{i,0})\) and \((x_{i,f}, y_{i,f}, z_{i,f})\) respectively denote the origin and destination of UAV \(i\).

The objective function in (2) weights the input profiles \(v_{x,i}(t), v_{y,i}(t), v_{z,i}(t)\) along the different axes from the initial time \(t_0\) to the final time \(t_f\) of each vehicle and is related to the amount of fuel required for the maneuver. The set of constraints \(g_{\text{ineq}} \leq 0\) are separation constraints between different pairs of UAVs. The set of constraints \(g_{\text{ineq}} \geq 0\) are separation constraints between UAV and static obstacles. \(v_{x,i}(t)^L, v_{x,i}(t)^U\), \(v_{y,i}(t)^L, v_{y,i}(t)^U\), \(v_{z,i}(t)^L, v_{z,i}(t)^U\) denote point-wise upperbound and lower-bound on the velocity of the UAV \(i\) along the \(x, y\) and \(z\) axis, respectively.

In general, the optimal control problem (2) cannot be solved analytically. Numerical approximations are used to convert the infinite dimensional problem to a finite dimensional problem. In the next sections, we will briefly describe two approaches that we have used to numerically solve problem (2).

III. Solution via Nonlinear Programming

In this section we briefly review the approach used in [20] to convert the infinite dimensional optimal control problem (2) into a finite dimensional nonlinear program. First, we detail the form of the cost function and of the constraints in (2) used in this approach. The cost function weights the one-norm of the UAV velocity vectors components

\[
J_i(v_x, v_y, v_z) = \int_{t_0}^{t_f} \left( ||v_x(t)||_1 + ||v_y(t)||_1 + ||v_z(t)||_1 \right) dt
\]
The inequality constraints $g_{ineq}$ are chosen as
\[
|x_i(t) - x_j(t)| \geq R \vee |y_i(t) - y_j(t)| \geq R \vee |z_i - z_j(t)| \geq H
\]
where $2R$ and $2H$ are base-length and height of square-base parallelepiped protection zones around each UAV, respectively. The inequality constraints $g_{ineq}$ are chosen as
\[
\|x_i(t) - Ox_j\| \geq R_j \vee \|y_i(t) - Oy_j\| \geq R_j \vee \|z_i - Oz_j\| \geq H_j
\]
where $(Ox_j, Oy_j, Oz_j)$ are the centers of square-base parallelepiped protection zones around the $j$th static obstacle. $2R_j$ and $2H_j$ are base-length and height of the parallelepipeds. $N_o$ is the number of obstacles.

The key challenge in modeling the problem (2) is represented by the disjunctions in (4)-(5) that model the protection zone around each UAV and each obstacle. The disjunctive equation can be formulated using continuous variables as described below. Suppose for simplicity that the disjunctive equation is given as,
\[
g_1(t) \geq 0 \vee g_2(t) \vee g_3(t) \geq 0.
\]
The above disjunction can be posed using continuous variables as follows
\[
\begin{align*}
\lambda_1(t)g_1(t) + \lambda_2(t)g_2(t) + \lambda_3(t)g_3(t) & \geq 0 \\
\lambda_1(t) + \lambda_2(t) + \lambda_3(t) & = 1
\end{align*}
\]
(7)
The second equation in the above formulation is critical to show equivalence to the disjunction (6). The equation denotes a normalization constraint for $\lambda_1(t)$, $\lambda_2(t)$, $\lambda_3(t)$. Of course, the right hand side could be any positive constant. In its absence, $g_1(\hat{t}) < 0$, $g_2(\hat{t}) < 0$, $g_3(\hat{t}) < 0$, $\lambda_1(\hat{t}) = \lambda_2(\hat{t}) = \lambda_3(\hat{t}) = 0$ for some $\hat{t}$, will satisfy the continuous formulation but violate the disjunction. Nevertheless, the first equation in (7) is nonconvex, and can induce local minima. Using the above formulation we can now discretize the optimal control problem to obtain a finite dimensional Nonlinear Program (NLP). In particular we use the monomial basis representation [1] for the differential, algebraic and control variable profiles. The collocation points within each element are chosen to be at Radau points. In Section III-A we give more details on the discretization procedure. The resulting NLP is solved using an interior point nonlinear programming solver, IPOPT [9], [10] discussed in Section III-B.

A. Collocation on Finite Elements

We employ collocation on finite elements to discretize the optimal control problem (2). The profiles of the variables are approximated by a family of polynomials on finite elements. The time interval of interest $[t_0, t_f]$ is divided into $ne$ finite elements of length $h_i$ such that
\[
\sum_{i=1}^{ne} h_i = t_f - t_0.
\]
Further we may define the time at the start of the intervals as, $t_i := t_0 + \sum_{i=1}^{t_i} h_i | i = 1, ..., ne$. A monomial basis representation [1] is used for the differential profiles, as follows:
\[
x_n(t) = x_{n,i-1} + h_i \sum_{q=1}^{ncol} \Omega_q \left( \frac{t-t_{i-1}}{h_i} \right) \frac{dx_n}{dt_{i,q}}
\]
(8)
where $x_{n,i-1}$ approximates the value of the differentiable variable at the beginning of element $i$, $x_n(t_{i-1})$, $\frac{dx_n}{dt_{i,q}}$ approximates the first derivative in element $i$ at the collocation point $q$ and $\Omega_q$ is a polynomial of order $ncol$, satisfying,
\[
\Omega_q(\rho_r) = 0 \quad \forall q = 1, ..., ncol \\
\Omega_q(\rho_r) = \delta_{q,r} \quad \forall q = 1, ..., ncol
\]
where $\rho_r$ is the $r$th collocation point within each element, and $\delta_{q,r}$ is a Kronecker delta. We use Radau points as they allow to set constraints easily at the end of each element, and stabilize the system efficiently when high index constraints are present. In addition the monomial representation leads to smaller condition numbers and rounding errors [1]. The control and algebraic profiles are approximated using the monomial basis representation as,
\[
y_n(t) = \sum_{q=1}^{ncol} \psi_q \left( \frac{t-t_{i-1}}{h_i} \right) y_{n,i,q}
\]
(9)
\[
u_n(t) = \sum_{q=1}^{ncol} \psi_q \left( \frac{t-t_{i-1}}{h_i} \right) u_{n,i,q}
\]
(10)
where $y_{n,i,q}$ and $u_{n,i,q}$ represent the values of the algebraic and control variables, respectively, in each element $i$ at collocation point $q$. $\psi_q$ is a Lagrange polynomial of order $ncol$ satisfying
\[
\psi_q(\rho_r) = \delta_{q,r}.
\]
The collocation formulation imposes continuity across the time elements for the differential variables (8), while the algebraic and control variables are allowed to have discontinuities at the boundaries of the elements.

We assume that the number of finite elements $ne$, and their lengths are pre-determined. With this assumption
the substitution of the profile approximations and slacks for the inequality constraints yields the following non-linear programming problem,

$$J_{NLP} = \min_{\xi \in \mathbb{R}^{\mathbb{Z}}} \phi(\xi)$$

s.t. 

$$c(\xi) = 0, \quad \xi^L \leq \xi \leq z^U$$

where $$\xi = (\alpha_{n,i,q}, y_{n,i,q}, u_{n,i,q}), \phi : \mathbb{R}^{\mathbb{Z}} \rightarrow \mathbb{R}$$ and $$c : \mathbb{R}^{\mathbb{Z}} \rightarrow \mathbb{R}^{n_c}$$.

We employ a barrier algorithm to solve the NLP (11). The main idea of the algorithm is described in the next section. Optimal solutions for joint maneuvers typically bring UAVs to within the minimum separation. Different separation constraints are active depending on which pair of UAVs are at the minimum separation. Given the large number of separation constraints $$N(N-1)/2(nc)(ncol)$$ an active set algorithm can be expensive if a good guess for the optimal active set is not readily available. Barrier algorithms [4], [10] have been shown to overcome the combinatorial bottleneck of current Sequential Quadratic Programming (SQP) methods of identifying the active inequality constraints. Encouraging results have been obtained on a number of process engineering problems with a large number of inequality constraints [3].

B. NLP solver

The NLP (11) is solved using IPOPT, an interior point algorithm. The algorithm follows a barrier approach, where the bounds on the variables $$z$$ are replaced by a logarithmic barrier term which is added to the objective to yield the following problem ($$P_{\mu}$$)

$$\min_{\xi} \phi(\xi) - \mu \sum_{j=1}^{n_c} \ln(\xi^{(j)} - \xi^{L(j)}) - \mu \sum_{j=1}^{n_c} \ln(\xi^{U(j)} - \xi^{(j)})$$

s.t. 

$$c(\xi) = 0$$

with a barrier parameter $$\mu > 0$$. Here, $$\xi^{(j)}$$ denotes the $$j$$-th component of the vector $$\xi$$. The barrier algorithm attempts to solve the NLP (11) by solving a sequence of barrier problems $$\{P_{\mu}\}$$ with decreasing barrier parameter, $$\{\mu_k\} \rightarrow 0$$. For a given barrier parameter $$\mu > 0$$ the logarithmic term in the objective function of barrier problem ($$P_{\mu}$$) serves to push the iterates into the strict interior of the region defined by the bounds. As the barrier parameter $$\mu$$, is driven to zero the influence of the logarithmic term is diminished and under certain assumptions, we are also able to recover solutions that make a bound active. In essence, the barrier algorithm approaches the solution to the barrier problem from the interior of the feasible region.

The interior point approach avoids having to identify the active inequalities at the solution and the combinatorial difficulty associated with it. The algorithm, IPOPT solves for the stationary conditions of problem, ($$P_{\mu}$$) whose solution is obtained by the application of Newton’s method. The interested reader is referred to the work of Wächter and Biegler [9], [10] and the work of Biegler et. al. [4] for details on the convergence and implementation. IPOPT has been developed for solving large-scale nonlinear programming problems and has also been interfaced to a modeling language, AMPL [5]. The algorithm was initialized by using the initialization strategies presented in [20].

IV. SOLUTION VIA MIXED INTEGER LINEAR PROGRAMMING

In this section we describe the approach used to convert the infinite dimensional optimal control problem (2) into a finite dimensional Mixed Integer Linear Program (MILP) [18].

The objective function and constraints on the control inputs used in this approach are identical to the nonlinear programming approach. In the first step the UAV dynamics and cost function are discretized with a sampling time $$T_s$$ by using a Euler discretization

$$\begin{align*}
    x_i(k+1) &= x_i(k) + T_s v_{x,i} \\
    y_i(k+1) &= y_i(k) + T_s v_{y,i} \\
    z_i(k+1) &= z_i(k) + T_s v_{z,i}
\end{align*}$$

The disjunctions describing vehicles protection zones (4) are reformulated by using binary variables and “big-M” technique as described next. Consider two vehicles $$i$$ and $$j$$, and six binary variables $$b_{i,j}^1 \in \{0, 1\}$$, ..., $$b_{i,j}^6 \in \{0, 1\}$$. Constraint (4) is reformulated into the following set of mixed integer inequalities

$$\begin{align*}
    x_i(k) &\geq x_j(k) + R - m_x b_{i,j}^1(k) \\
    x_i(k) &\leq x_j(k) - R + M_x b_{i,j}^2(k) \\
    y_i(k) &\geq y_j(k) + R - m_y b_{i,j}^3(k) \\
    y_i(k) &\leq y_j(k) - R + M_y b_{i,j}^4(k) \\
    z_i(k) &\geq z_j(k) + H - m_z b_{i,j}^5(k) \\
    z_i(k) &\leq z_j(k) - H + M_z b_{i,j}^6(k) \\
    \sum_{k=1}^6 b_{i,j}^k(k) &\leq 5
\end{align*}$$

where we assume that $$x_i(k) \in \mathcal{X} \subset \mathbb{R}^n$$, $$y_i(k) \in \mathcal{Y} \subset \mathbb{R}^n$$, $$z_i(k) \in \mathcal{Z} \subset \mathbb{R}^n \forall i = 1, \ldots, n$$ and $$\mathcal{X}$$, $$\mathcal{Y}$$, $$\mathcal{Z}$$ are given bounded sets. The “bigM” constants $$m_x, m_y, m_z, M_x, M_y, M_z$$ are lowerbounds and upperbounds on the difference between the position of
two UAVs

\[ M_x \geq \sup_{x_1, x_2 \in X} (x_1 - x_2 + R), \]
\[ m_x \leq \inf_{x_1, x_2 \in X} (x_2 - x_1 + R), \]
\[ M_y \geq \sup_{y_1, y_2 \in Y} (y_1 - y_2 + R), \]
\[ m_y \leq \inf_{y_1, y_2 \in Y} (y_2 - y_1 + R), \]
\[ M_z \geq \sup_{z_1, z_2 \in Z} (z_1 - z_2 + H), \]
\[ m_z \leq \inf_{z_1, z_2 \in Z} (z_2 - z_1 + H), \]

In the same way constraint (5) describing obstacles protection zones are reformulated into a set of mixed integer inequalities by using six binary variables \( O_{k,j}^b \in \{0, 1\}, \ldots, O_{k,j}^b \in \{0, 1\} \) for each obstacle-vehicle pair \((i, j)\) and computing six constants \( O_{x,i}, O_{y,i}, O_{z,i}, O_{M_x}, O_{M_y}, O_{M_z} \).

Problem (2) is reformulated into the following finite dimensional mathematical program:

\[
\min_U \sum_{k=0}^{N-1} \sum_{i=0}^{N-1} |x_{i}(k)| + |y_{i}(k)| + |z_{i}(k)| \quad (14)
\]

\[
\begin{aligned}
& x_{i}(k) \geq x_{j}(k) + R - m_x b_{i,j}^1(k) \\
& x_{i}(k) \leq x_{j}(k) - R + M_x b_{i,j}^2(k) \\
& y_{i}(k) \geq y_{j}(k) + R - m_y b_{i,j}^3(k) \\
& y_{i}(k) \leq y_{j}(k) - R + M_y b_{i,j}^4(k) \\
& z_{i}(k) \geq z_{j}(k) + H - m_z b_{i,j}^5(k) \\
& z_{i}(k) \leq z_{j}(k) - H + M_z b_{i,j}^6(k) \\
& \sum_{i=1}^{6} |b_{i,j}(k)| \leq 5, \quad i = 1, \ldots, n, \\
& \sum_{i=1}^{6} |b_{i,j}(k)| \leq 5, \quad j = 1, \ldots, N, \quad k = 0, \ldots, N-1
\end{aligned}
\]

The mathematical programming formulation (15) can be rewritten as a mixed-integer linear program by using a standard approach which make uses of additional slack variables. We will not detail the step of the translation of problem (15) into a MILP, details can be found in [21], [19]. We solved the mathematical programming formulation (15) by using the mathematical programming language AMPL [5] which transforms (15) it into a MILP of the form

\[
\begin{aligned}
& \min_{\epsilon_c, \epsilon_b} \quad f_c \epsilon_c + f_b \epsilon_b \\
& \text{subj. to} \quad G_c \epsilon_c + G_b \epsilon_b \leq b \quad (16)
\end{aligned}
\]

where \([\epsilon_b, \epsilon_c]\) is the optimization vector, which includes the inputs, the states and the binary variables associated with the vehicles and obstacles at all time steps, plus additional slack variables needed for the translation of problem (15) in to an MILP. The mathematical programming language AMPL translates the problem (15) into an MILP of the form (16). Then, the MILP (16) is solved by a Branch and Bound algorithm provided by the software CPLEX [17] which is interfaced with AMPL.

V. NLP/MILP FORMULATION DIFFERENCE

Before comparing the nonlinear and the mixed-integer approach presented in the previous sections, we highlight the only difference between the formulations used in the two approaches. The discretization in time of the original optimal control problem (2) is uniform over the horizon \( t_f - t_o \) in the MILP approach, while it is a function of the position of the Radau points (see Section III-A) in the NLP approach. This implies that the NLP and the MILP optimal trajectories will fulfill the constraints at different points in time. For small sampling times (for the MILP) and big number of collocation points (for the NLP) this difference can be neglected. Also, since we are interested in the difference between the computational burden of the two techniques, the number of discretization points in time is more important than their collocation. We have used a sampling time \( T_s \) in (12) of 3s which results into 30 samples over an horizon of 90s. Accordingly in the MILP approach the number of Radau points over the same horizon have been chose equal to 30.

We want to mention here that we have also performed extensive simulations with NLP approach with cylindrical protection zones around each UAV, and an objective function of minimizing the integral over time of a squared Euclidean norm of the UAV velocity vector (thus improving the smoothness of cost and constraint functions). For such formulation, the NLP solver performance was comparable to the formulation with a one norm in the objective function and parallelepipedal protection zones.
VI. RANDOM SCENARIOS GENERATOR

The comparison between the two solution methods presented in Section III and Section IV has been carried out on a set on problem instances that were randomly generated. In this section we give some details on the instance-set generator. A instance-set generator provides a set of random problem instances. For each problem instance belonging to the set, number of vehicles \( n \) and number of obstacles \( N_o \) are fixed. Each problem instance belonging to the set will be uniquely identified by the triplet \((i,j,k)\), \( i \in \{n_{\min}, \ldots, n_{\max}\} \), \( j \in \{N_{o_{\min}}, \ldots, N_{o_{\max}}\} \), \( k \in \{1, \ldots, N_{\text{max}}\} \). The problem \((i,j,k)\) is the \( k \)-th random problem with \( n = i \) vehicles and \( N_o = j \) obstacles. Therefore, a set will be composed of \((n_{\max} - n_{\min})(N_{o_{\max}} - N_{o_{\min}} + 1)\) groups. Each group \((i,j)\) is composed of \( N_{\text{max}} \) random problems with \( n = i \) vehicles and \( N_o = j \) obstacles. \((i,j)\) will be the called the group number. We have considered random sets of problem instances with \( n_{\min} = 2, n_{\max} = 2, N_{o_{\min}} = 5, n_{max} = 10, N_{o_{\max}} = 6 \), for a total of 216 random problems for each instance-set.

For each problem instance belonging to the set, the algorithm randomly generates: (i) initial and terminal position of each vehicle: \((x_{i,0}, y_{i,0}, z_{i,0})\) and \((x_{i,f}, y_{i,f}, z_{i,f})\) for \( i = 1, \ldots, n \); (ii) \( N_o \) obstacles where the \( i \)-th obstacle \( O_i \) is described through the quintuple \((Ox_i, Oy_i, Oz_i, Ri, Hi)\). The algorithm makes sure that initial and terminal positions respect vehicles and obstacles protection zones.

In order to explore the possibility of defining a simple measure for time complexity of a problem instance, we associated a vector, \([a, b, c]\), to each problem instance. The component \( a \) denotes the density of the obstacle field measured as total volume of the obstacles field divided by the volume of field where UAVs are allowed to fly, \( 0 \leq a \leq 1 \), \( a \in \mathbb{R} \). The component \( b \) is the number of vehicle/vehicle collisions when vehicles fly straight paths from their initial position to their end position at any speed, \( 0 \leq b \leq n(n-1), b \in \mathbb{N} \). The component \( c \) is the number of vehicle/obstacle collisions when vehicles fly straight paths from their initial position to their end position at any speed, \( 0 \leq c \leq nN_o, c \in \mathbb{N} \).

VII. COMPARISON RESULTS

Based on results of extensive random simulations we compared time complexity and optimality of the solution of the NLP approach presented in Section III and of the MILP approach presented in Section IV. A summary of the observation reported in this section is based on more than 10 random instance-sets (more than 2160 problem instances). We compared only problem instances that were feasible for both approaches. The number of infeasible problem instances was not significant compared to the total number of feasible instances. There are three relevant results we have obtained:

1) MILP was faster than the NLP for all problem instances we solved. As the problem size grows (in terms of number of vehicles and obstacles), the gap between the MILP solution time and the NLP solution time increases. Also the NLP reports much higher oscillations in solution times for larger problem instances, compared to the MILP solutions.

2) MILP and NLP always provide optimal solutions with similar costs. We have computed the cost formulated for the NLP approach, for both MILP and NLP optimal trajectories. Then, we have defined a relative cost error as \[ J(z_{\text{MILP}}) = J(z_{\text{NLP}}) \]

\[ J = J(v_{x,1}, v_{y,1}, v_{z,1}, \ldots, v_{x,n}, v_{y,n}, v_{z,n}) = \sum_{i=1}^{n} \int_{t_0}^{t_f} \left[ \|v_x(t)\|^2 + \|v_y(t)\|^2 + \|v_z(t)\|^2 \right] dt \]

and \( z_{\text{MILP}} \) and \( z_{\text{NLP}} \) are the MILP and NLP optimizers, respectively. The average relative cost error was always on the order of \( 10^{-1} \), which implies a good performance by the NLP algorithm.

3) There is no correlation between the vector \([a, b, c]\) associated to a problem instance and the corresponding time complexity. We were expecting that for a fixed number of vehicles and obstacles the time complexity of a problem instance is related to parameters like density of the obstacles field or number of collision when vehicles fly along straight lines joining initial and final destination at any speed: this was not the case. We have found problem instances where the density of the field was small and where straight flight is a feasible solution which were more complex than problem instances with higher density of the obstacle field and where a straight flight would cause several collisions.

Next we report some data for one random instance-set with 216 problem instances, out of which 214 were feasible (for both the MILP and the NLP solvers). The MILP and NLP minimum solution times were 1.091 s and 1.422, respectively. The MILP and NLP maximum solution times were 18.20 s and 506.18 s, respectively. The MILP and NLP average solution times were 4.32 and 46.53, respectively. The relative maximum differ-
ence between the MILP and NLP optimal cost was 0.45%. The relative average difference between the MILP and NLP optimal cost was 0.22%. The data for all the groups of the instance-set are reported in Figure 1. Detailed data for some subgroups of the instance-set are reported in Tables I, II. Visualization of trajectories obtained for both approaches will be included in the final version of the paper.

<table>
<thead>
<tr>
<th>Group Number</th>
<th>2-2</th>
<th>5-5</th>
<th>8-2</th>
<th>10-4</th>
<th>10-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feasible cases</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Both infeasible cases</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Min sol. time MILP</td>
<td>1.16</td>
<td>2.40</td>
<td>4.44</td>
<td>10.00</td>
<td>12.85</td>
</tr>
<tr>
<td>Min sol. time NLP</td>
<td>1.24</td>
<td>7.25</td>
<td>22.25</td>
<td>97.63</td>
<td>51.93</td>
</tr>
<tr>
<td>Max sol. time MILP</td>
<td>1.26</td>
<td>3.09</td>
<td>5.14</td>
<td>13.66</td>
<td>18.20</td>
</tr>
<tr>
<td>Max sol. time NLP</td>
<td>8.07</td>
<td>250.96</td>
<td>168.15</td>
<td>506.18</td>
<td>206.14</td>
</tr>
<tr>
<td>Ave. sol. time MILP</td>
<td>1.19</td>
<td>2.59</td>
<td>4.70</td>
<td>11.56</td>
<td>14.31</td>
</tr>
<tr>
<td>Ave. sol. time NLP</td>
<td>3.83</td>
<td>24.53</td>
<td>29.99</td>
<td>244.35</td>
<td>129.48</td>
</tr>
<tr>
<td>Max rel. cost error</td>
<td>0.45</td>
<td>0.30</td>
<td>0.22</td>
<td>0.23</td>
<td>0.21</td>
</tr>
<tr>
<td>Ave. rel. cost error</td>
<td>0.27</td>
<td>0.23</td>
<td>0.19</td>
<td>0.21</td>
<td>0.18</td>
</tr>
</tbody>
</table>

**TABLE I**

**GROUPS COMPARISON**

VIII. CONCLUSIONS

In this study, we have compared the time complexity and optimality of solutions obtained for multi-UAVs conflict resolution and obstacle avoidance with MILP and NLP approaches, based on the results of extensive random simulations. First, we have noticed that the MILP approach is always faster than the NLP approach, across the extensive set of random problem instances that we solved. In particular, when the optimal control problem becomes large, in terms of number of vehicles and obstacles (causing an explosion of combinatorics in the MILP case, and of non-convex constraints in the NLP case) the MILP approach outperforms the NLP approach. We used linear dynamical models for the UAVs. Conflict resolution algorithms that work with simple linear models may generate trajectories that are not necessarily flyable in practice. The issue of flyability of the optimal trajectories that are generated by any solution approach is critical to both safety and performance, and necessitates the use of more realistic and detailed dynamic models. Therefore, the first result of this paper opens the route to the use of MILP solutions (based on simple linear UAV models) in order to initialize NLP solvers which allow the use of more detailed dynamic UAV models. The second objective of the study was to test the performance of the NLP solver presented in [20] in terms of optimality of the solution. In fact, while the NLP solver in [20] does not guarantee to converge to the global optimum, as opposed to the MILP approach, we notice that the costs of the solutions obtained with the MILP and the NLP algorithms are always comparable. Lastly, we have studied the relation between time complexity of the optimal solution (both MILP and NLP) and some physical parameters of the optimization problem such as density of the obstacle field or the number of collisions when the vehicles fly at any speed straight lines joining initial and final destination. Interestingly, we did not find any correlation correlation between such physical parameters associated to a problem instance and its time complexity from our set of random problem instances. The results appear promising in the context of using linear MILP models, as well as for utilizing MILP-based solutions to provide a good initial guess to NLP formulations that can handle detailed dynamics. This has potential implications for on-line flyable solutions to the multi-UAVs conflict resolution problem.

**REFERENCES**

Fig. 1. Comparison between the MILP and the NLP approach for a set of 216 random problem instances.