Proceedings of the 2006 American Control Conference
Minneapolis, Minnesota, USA, June 14-16, 2006

MILP Optimal Path Planning for Real-Time Applications

Cedric S. Ma and Robert H. Miller
Northrop Grumman Integrated Systems
One Hornet Way, El Segundo, CA

Abstract— **This paper presents several efficient solution techniques specific to the optimal path planning of an autonomous vehicle. Mixed-Integer Linear Programming (MILP) is the underlying problem formulation, from which an optimal solution can be obtained through the use of a commercially available MILP solver such as CPLEX. The solution obtained is optimal in terms of the cost function specified in terms of fuel, time, altitude, etc. Several techniques are introduced to reduce the complexity of the underlying mathematical problems as to help make the path planning approach suitable for running in a mission-critical real-time environment. Some of these techniques may be applicable to other optimal path planning approaches.**

I. INTRODUCTION

Mixed-integer linear programming (MILP) is a recently developed approach to path planning that has proven to be exceptionally suitable for obstacle avoidance subject to dynamic constraints. The strength of the approach is that MILP can efficiently solve global optimization problems common in obstacle avoidance, while incorporating linear vehicle dynamic constraints that may lead to an overly complex problem in other methods, particularly in combinatorial approaches to path planning based on graph formulations and dynamic programming. Cell-based and potential field approaches [1] may not take the global properties of the problem into account, and usually have minimal or nonexistent dynamic constraints compared to the MILP approach. For a survey of other common approaches to path planning see [2].

A mixed-integer linear programming (MILP) problem differs from a linear programming (LP) problem only in that some of the variables are restricted to be integers and thus the solution space for a MILP problem is a subset of that of a similar LP problem (the MILP problem without the integer restrictions). Our work is based on research by Professors How and Feron at Massachusetts Institute of Technology. In [3], a basic MILP problem formulation is presented, combining fuel-optimal path planning for multiple vehicles with double-integrator vehicle dynamics and obstacles in two-dimensional space. More advanced implementations of this MILP approach include: safety guarantees [4], and a receding-horizon approach to avoid entrapment behind large obstacles [5].

In this paper, we will first build up our basic MILP path planning problem similar to that of [3], introduce a terrain flight formulation, and present the motivations for and the techniques that help make the path planning approach suitable for a real-time applications environment. We then discuss the issues that are encountered when the path planning algorithm is to be integrated in flight-capable hardware and software, and conclude with our experimental setup and results.

II. BASIC OBSTACLE AVOIDANCE PROBLEM

A standard mixed-integer linear programming (MILP) problem has the following form:

$$\min_x f(x) = c^T x \tag{1}$$

$$b_L \leq Ax \leq b_U \tag{2}$$

$$x_L \leq x \leq x_U \tag{3}$$

where $A \in \mathbb{R}^{m \times n}$; $c, x, x_L, x_U \in \mathbb{R}^n$; $b_L, b_U \in \mathbb{R}^m$, and a subset of x is restricted to be integers. For constraints (2), the rows corresponding to the portion(s) of x that are restricted to be integers are *integer constraints*, while the rest of the rows are *continuous constraints*.

An optimal solution x, if a feasible solution exists, is a vector of variables for which the value under the objective function (1) is smallest, such that x satisfies the constraints (2) and bounds (3). That is, for any other $x* \neq x$, either $f(x*) \geq f(x)$, or x does not satisfy at least one of (2), (3), and the integer restriction.

Mixed-integer linear programming is especially suited for path planning in an environment with obstacles because the continuous constraints can represent the dynamics of the vehicle while the integer constraints can represent collision constraints with obstacles.

A. Basic LP Trajectory Planning Problem

We start with linear, time invariant (LTI), continuous time plant dynamics:

$$\dot{s} = A_c s + B_c u \tag{4}$$

and define our initial and final conditions to be s_0 and s_F, respectively. In particular, and for the rest of the paper, our plant dynamics would consist of a double integrator model with acceleration as input:

$$\begin{aligned} \ddot{x} &= u_x \\ \ddot{y} &= u_y \\ \ddot{z} &= u_z \end{aligned} \tag{5}$$

By defining $v_x = \dot{x}$, $v_y = \dot{y}$, $v_z = \dot{z}$, we can easily rewrite (5) in the form of (4) by setting $s =$

$\begin{bmatrix} x & y & z & v_x & v_y & v_z \end{bmatrix}^T$ and $u = \begin{bmatrix} u_x & u_y & u_z \end{bmatrix}^T$ and get $\dot{s} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} s + \begin{bmatrix} 0 \\ I \end{bmatrix} u$.

For the purpose of formulating trajectory planning problem as an LP or MILP problem, the plant dynamics need to be discretized with a chosen period to yield:

$$s[k+1] = As[k] + Bu[k] \tag{6}$$

Using equation (6) recursively, and assuming the notation $s_k = s[k], u_k = u[k]$, we obtain:

$$s_k = A^k s_0 + A^{k-1}Bu_0 + \cdots + ABu_{k-2} + Bu_{k-1} \tag{7}$$

Assuming that we are solving a problem with N time steps, we have $s_F = s_N$. We apply this to equation (7) to yield a set of *destination constraints*:

$$s_F - A^N s_0 = A^{N-1}Bu_0 + \cdots + ABu_{N-2} + Bu_{N-1} \tag{8}$$

In the absence of other constraints, a solution to equation (8) is a set of N piecewise constant controls $\{u_0, \cdots, u_{N-1}\}$ that will steer the continuous system (5) from s_0 to s_F in N time periods.

Velocity constraints on (v_x, v_y, v_z) can be expressed as the following:

$$v_{min} \leq v_k \leq v_{max} \tag{9}$$

where we define $v = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}^T$, $v_k = v[k] = Vs_k$, $V = \begin{bmatrix} 0 & I \end{bmatrix}$, $v_{min} = \begin{bmatrix} v_{x,min} & v_{y,min} & v_{z,min} \end{bmatrix}^T$, $v_{max} = \begin{bmatrix} v_{x,max} & v_{y,max} & v_{z,max} \end{bmatrix}^T$

We apply equation (7) to (9) to obtain the *velocity constraints* for time $k \in \{0, \cdots, N-1\}$:

$$\begin{aligned} v_{min} - VA^k s_0 \leq VA^{k-1}Bu_0 + \cdots + VABu_{k-2} + \\ VBu_{k-1} \leq v_{max} - VA^k s_0 \end{aligned} \tag{10}$$

Control limits on (u_x, u_y, u_z) can be expressed as the following, for $k \in \{0, \cdots, N-1\}$:

$$u_{min} \leq u_k \leq u_{max} \tag{11}$$

where: $u_{min} = \begin{bmatrix} u_{x,min} & u_{y,min} & u_{z,min} \end{bmatrix}^T$, $u_{max} = \begin{bmatrix} u_{x,max} & u_{y,max} & u_{z,max} \end{bmatrix}^T$

For a solution that is both fuel and time optimal, we would like the objective function to be the following:

$$f(x) = \sum_{k=0}^{N-1} r\|u_k\|_1 + \sum_{k=1}^{N-1} q\|X_k - X_F\|_1 \tag{12}$$

where we define $X = \begin{bmatrix} x & y & z \end{bmatrix}^T$, $X_k = X[k] = Ws_k$, $X_F = Ws_F$, $W = \begin{bmatrix} I & 0 \end{bmatrix}$.

$\|X_k - X_F\|_1$ is the 1-norm distance in (x, y, z) between s_k and s_F. r and q are non-negative weight factors. The first and second term in (12) are known as *fuel cost* and *non-arrival cost*, respectively. The non-arrival cost in the objective function is the key to a minimum-time formulation.

Since a 1-norm function is not linear, we need to introduce additional slack variables v_k and w_k to create the desired effect with the following constraints:

$$-v_k \leq u_k \leq v_k \tag{13}$$

$$-w_k \leq Ws_k - Ws_F \leq w_k \tag{14}$$

When the constraints (13) and (14) are combined with nonnegative weights q and r, they also guarantee that $v_k = \|u_k\|_1$ and $w_k = \|WS_k - WS_F\|_1 = \|X_k - X_F\|_1$, respectively, in the optimal solution.

Constraints (13) can be decomposed into two separate inequalities as MILP constraints for $k \in \{0, \cdots, N-1\}$:

$$\begin{aligned} u_k - v_k &\leq 0 \\ -u_k - v_k &\leq 0 \end{aligned} \tag{15}$$

Similarly, constraints (14) can be decomposed into two separate inequalities, expanded out using equation (7), and written as MILP constraints for $k \in \{1, \cdots, N-1\}$:

$$\begin{aligned} W(A^{k-1}Bu_0 + \cdots + ABu_{k-2} + Bu_{k-1}) - w_k \\ \leq W(s_F - A^k s_0) \\ -W(A^{k-1}Bu_0 + \cdots + ABu_{k-2} + Bu_{k-1}) - w_k \\ \leq -W(s_F - A^k s_0) \end{aligned} \tag{16}$$

We can make use of the slack variables we defined in (13) and (14), formulated respectively as MILP constraints (15) and (16), to conveniently express the objective function (12) as:

$$f(x) = r\sum_{k=0}^{N-1} v_k + q\sum_{k=1}^{N-1} w_k \tag{17}$$

Our basic LP trajectory planning problem thus consists of the destination constraints (8), velocity constraints (10), slack variable constraints for energy (15) and time (16), control limits (11), and objective function (17). An optimal solution under these conditions is said to be a set of control actions that would take the system (5) from s_0 to s_F in N time periods with minimum cost as defined by (17).

B. MILP Trajectory Planning Problem with Obstacles Avoidance

We now expand our existing LP problem to include obstacle collision constraints.

It can be shown that for any arbitrary convex polyhedral bounded by n number of planes in 3-D, the interior of the polyhedral can be represented by the combination of the following set of inequalities:

$$\begin{aligned} a_1x + b_1y + c_1z &> d_1 \\ &\vdots \\ a_nx + b_ny + c_nz &> d_n \end{aligned} \tag{18}$$

The collision constraints requiring that s_k be outside the region defined by (18) can be rewritten as:

$$\begin{bmatrix} a_1 & b_1 & c_1 & 0 & 0 & 0 \\ & & \vdots & & & \\ a_n & b_n & c_n & 0 & 0 & 0 \end{bmatrix} s_k - M \begin{bmatrix} b_{k_1} \\ \vdots \\ b_{k_n} \end{bmatrix} \leq \begin{bmatrix} d_1 \\ \vdots \\ d_n \end{bmatrix} \tag{19}$$

$$b_{k_1} + \cdots + b_{k_n} \leq n - 1 \tag{20}$$

where $M >> 0$ is an arbitrarily large number, and b_{k_i} are binary variables. The constraints (19) and (20) essentially negates at least one set of the inequalities in (18) and forces s_k to be outside the region defined by (18).

By writing (19) in short form as $\theta_i s_k - M b^i_k \leq \theta'_i$ for each obstacle i, and by substituting s_k with equation (7), the final MILP obstacle collision constraints for obstacle $i \in \{1, \cdots, p\}$ would be, for $k \in \{1, \cdots, N\}$:

$$\begin{aligned} \theta_i(A^{k-1}Bu_0 + \cdots + ABu_{k-2} + Bu_{k-1}) - Mb^i_k \\ \leq \theta'_i - \theta_i A^k s_0 \\ \begin{bmatrix} 1 \cdots 1 \end{bmatrix}_{1 \times n_i} b^i_k \leq n_i - 1 \\ 0 \leq b^i_k \leq 1 \\ b^i_k \in integers \end{aligned} \tag{21}$$

Our MILP trajectory planning problem with obstacle avoidance consists of the constraints in the basic LP trajectory planning problem, i.e. (8), (10), (11), (15), (16), (17), plus obstacle collision constraints (21).

Assuming that N is the number of time steps, the above MILP problem will consist of $6+3N*2+3(N-1)*2+3N+N\sum_{i=1}^{p}(n_i+1)$ constraints, $3N+3N+3(N-1)$ continuous variables, and $N\sum_{i=1}^{p} n_i$ discrete variables, where p is the number of obstacles and n_i is the number of planes that define obstacle i.

III. Terrain Flight

"Terrain flight" is the tactic of using the terrain's characteristics and man-made objects to degrade the enemy's ability to visually, optically, or electronically detect or locate the aircraft. Nap-of-the-Earth (NOE) flight is a category of terrain flight characterized by maneuvers as close to the earth's surface as vegetation, obstacles, or ambient light will permit.

In Section II, we presented path planning with obstacle avoidance as a MILP problem. In this section, we consider terrain flight as a variation of obstacle avoidance, with some important differences: (1) obstacles are generally scattered in various locations, while terrain is constantly under the vehicle; (2) it usually suffices to avoid collision with the obstacles, but in terrain flight, the objective is to stay low and out of sight of the enemy.

Our approach to basic terrain flight is to construct our terrain as a series of obstacles that act as three-dimensional "floor tiles" that the vehicle travels over, while introducing an altitude cost to keep the trajectory close to the terrain:

$$\begin{aligned} f(x) = \sum_{k=0}^{N-1} r\|u_k\|_1 + \sum_{k=1}^{N-1} q\|X_k - X_F\|_1 \\ + p\|X_N - X_F\|_1 + \sum_{k=1}^{N} a_z z_k \end{aligned} \tag{22}$$

where $z_k = z[k] = Zs_k, Z = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}, a_z$ is a non-negative weight factor. To make the objective function (22) implementable in MILP, we need to introduce slack variables z_k and the following constraints by substituting equation (7) in $z_k = Zs_k$ to get, for $k \in \{1, \cdots, N\}$:

$$\begin{aligned} ZA^{k-1}Bu_0 + \cdots + ZABu_{k-2} + ZBu_{k-1} - z_k \\ = -ZA^k s_0 \end{aligned} \tag{23}$$

Together, (22) and (23) associate a cost to the trajectory proportional to the altitude at each point. By tuning the weight factor a_z against other weight factors r, q, and p in the original objective function (24), we can achieve NOE flight of varying aggressiveness.

A. Terrain Modeling

For this paper, we have chosen triangulated irregular networks (TIN) as our terrain representation since it is probably the most straightforward way of modeling grid-based terrain. TIN is simply a pattern of similarly oriented triangles that is laid on top of a regular square-based grid pattern, with an elevation assigned to each point on the grid, such as the one shown in Figure 1a. A straightforward way of implementing terrain flight in MILP would be to model each triangle in a TIN terrain as an obstacle with three vertical side walls, each of which is positioned adjacent to and directly facing the side wall associated with an adjacent triangle, as illustrated in Figure 1b. The points s_0 and s_F would be positioned somewhere above the terrain and the trajectory would be required to avoid collision with the terrain while staying as close to the terrain as possible per the objective function.

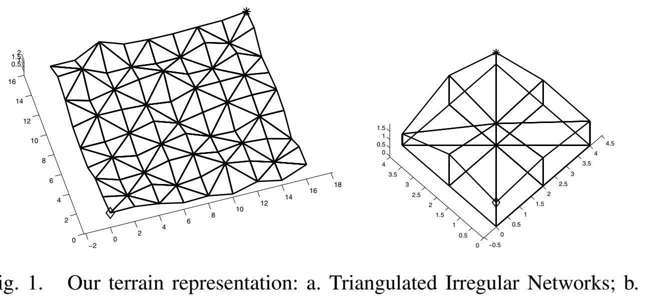

Fig. 1. Our terrain representation: a. Triangulated Irregular Networks; b. Obstacle representation of terrain tiles

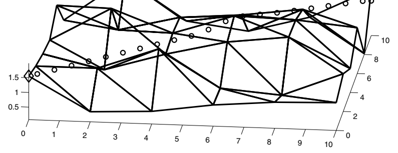

Fig. 2. Terrain Flight Trajectory

IV. Computational Techniques

In this section, we present several different techniques designed to reduce the computational requirements for a real-time implementation of the MILP path planning approach.

A. Receding Horizon

The main purpose of implementing a MILP trajectory planning problem in a receding horizon framework is to reduce the computational effort required to solve the MILP problem by reducing the problem size. Instead of planning the entire optimal trajectory from s_0 to s_F, which may be located very far apart and may require many time steps to reach, we break the problem down into many subproblems of smaller size with fewer time steps N, such that within each subproblem there is no longer the requirement to reach a particular destination state s_F (destination constraint (8)). Instead, the first subproblem uses s_0 as the initial state, and only a subset of the planned trajectory from the first subproblem is executed (i.e. only s_1 and s_2). Then s_2 from the first subproblem is used as the initial state in the second subproblem, and so on, until a subproblem with $s_N = s_F$ is calculated, in which case the entire trajectory of the subproblem is executed and the vehicle reaches the destination.

To encourage the overall trajectory as well as the individual trajectories of each subproblem to get closer to the destination, an additional term known as *terminal cost* enters the objective function (12):

$$f(x) = \sum_{k=0}^{N-1} r\|u_k\|_1 + \sum_{k=1}^{N-1} q\|X_k - X_F\|_1 + p\|X_N - X_F\|_1 \qquad (24)$$

The objective function thus becomes:

$$f(x) = r\sum_{k=0}^{N-1} v_k + q\sum_{k=1}^{N-1} w_k + pw_N \qquad (25)$$

and to accommodate the term $\|X_N - X_F\|_1$, an additional slack variable w_N is introduced according to equation (14) so that constraints (16) still holds but is now applied through the range $k \in \{1, \cdots, N-1\}$.

B. Multiple Timescales

Due to the complexity of a typical obstacle avoidance problem and the need to solve it in real-time, the overall obstacle avoidance problem is best solved in smaller parts. Consider an all-encompassing MILP obstacle avoidance problem with a small time discretization value (T) and long planning horizon (N), so that the problem can account for small obstacles and still maintain a big picture view of the obstacle field and terrain. This problem would be very large and would require a long solve time.

We can achieve more reasonable solve times while retaining the desired effects of small time discretization and long planning horizon by constructing and solving a number of smaller, simpler MILP problems in series. In doing so, we are in effect taking advantage of an important property of an obstacle avoidance problem, in that it is inherently of multiple timescales. That is, using the concept of abstraction, the problem can be naturally decomposed into subproblems of largest timescales (terrain navigation), medium timescales (urban obstacle avoidance), and smallest timescales (rounding corners of buildings, skimming surface of terrain, and avoiding trees and small objects). The longer timescale problems would perform longer range planning such as terrain navigation, while the shorter timescale problems would refine the solutions provided by the longer timescale problems to account for smaller obstacles.

Figure 3 illustrates the concept of multiple timescales with a two timescale example. We introduce the notion of an *execution horizon*, which denotes the subset of the planned trajectory to be executed. In general, the execution horizon is shorter than the planning horizon, but they can be equal. In conjunction with time discretization (T) and planning horizon (N), these three parameters determine how the problems from different timescales will be integrated. The sequence starts when the longest timescale MILP path-planning problem is formulated and solved with s_0 at the current vehicle location, and s_F the true destination. The solution at the end of its execution horizon is taken as the destination s_F for the next shorter timescale problem, reusing the same s_0. The solution at the end of the new problem's execution horizon is then taken as the s_F of the next one, and so on, while reusing the same s_0. This sequence ends when the solution of the shortest timescale solution is given as a series of waypoints for the vehicle to follow. When the vehicle reaches the waypoint corresponding to the end of the shortest timescale problem's execution horizon, this location is designated the new s_0 and the entire sequence starts anew, until the true destination is reached or changed.

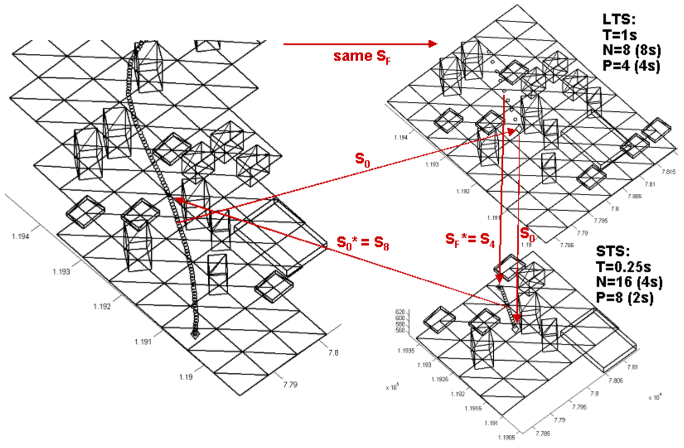

Fig. 3. Multiple Timescales

C. Non-uniform Time-Step

Notice that many sets of MILP constraints presented in the previous sections (such as equation (8), (10), (16), (21), and (23)) are based on the enumeration of equation (7):

$$\begin{aligned} s_1 &= As_0 + Bu_1 \\ s_2 &= A^2 s_0 + ABu_1 + Bu_2 \\ s_3 &= A^3 s_0 + A^2 Bu_1 + ABu_2 + Bu_3 \\ &\vdots \\ s_k &= A^k s_0 + A^{k-1} Bu_1 + \cdots + ABu_{k-1} + Bu_k \end{aligned}$$

Equation (7) is simply a mechanism for the propagation of constraints, with the premise that the future trajectory of a vehicle depends on its initial conditions and subsequent control inputs. In this manner, every type of MILP constraints (i.e. for altitude) is instantiated for each time step $i \in \{1, \cdots, N\}$ of the MILP problem. This is because the MILP problem is posed as an optimization problem for a path consisting of the trajectory point s_k for each time step $k \in \{1 \cdots N\}$.

This uniform stepping of time is not required, and not always desirable. For instance, for an identically-sized MILP problem one might prefer a nonlinear scaling of time so that time is more sparsely captured further into the future and more finely represented closer to the present. For example, instead of planning for s_k, $k \in \{1, \cdots, N\}$ we can have $k \in \{1, 2, 4, 8, \cdots\}$ We can achieve this by doing the following:

$$\begin{aligned} s_1 &= As_0 + Bu_1 \\ s_2 &= A^2 s_0 + ABu_1 + Bu_2 \\ s_4 &= A^4 s_0 + A^3 Bu_1 + A^2 Bu_2 + ABu_4 + Bu_4 \\ s_8 &= A^8 s_0 + A^7 Bu_1 + A^6 Bu_2 + \ldots + ABu_7 + Bu_8 \end{aligned}$$

By choosing a coarser representation of time further into the future, the problem is now simplified by having to plan for fewer time steps because the controls u_k for multiple time steps are combined into one (i.e. $u_3 = u_4$). With this method of non-uniform time-steps, we are able to construct a MILP problem with the same planning horizon with fewer time steps and therefore smaller problem size, or alternatively, for a problem with the same problem size, we can have a longer planning horizon.

V. Implementation Issues

Figure 4 illustrates the resulting nap-of-the-earth flight trajectory computed by our path planning algorithm. Since we have incorporated a combination of the computational techniques we have introduced in Section IV into our path planning algorithm, the trajectory is actually a composite from a collection of optimal sub-trajectories that differ in time (receding horizon), scale (multiple time scales), and time gridding (non-uniform time step). It is precisely because of the successful application of the techniques in Section IV that one is able to compute such a detailed, long-duration trajectory as in Figure 5 within a reasonable computation time.

In a typical implementation, the path planning algorithm is written in C code which formulates the MILP problems

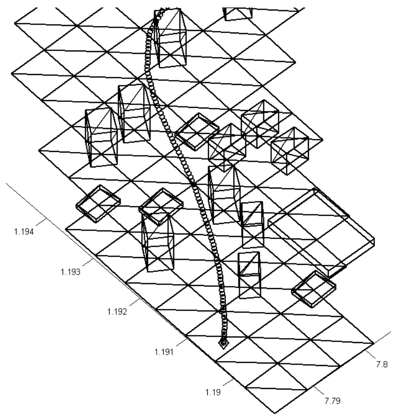

Fig. 4. Composite snapshot showing computed trajectory and all relevant obstacles and terrain considered in the MILP subproblems

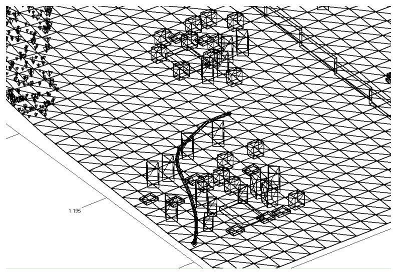

Fig. 5. Complete obstacles and terrain environment showing computed composite trajectory

as we have detailed throughout this paper. The application of the techniques introduced in Section IV requires that many subproblems be generated and their results reapplied to subsequent problems. Therefore, the overall algorithm needs to have the means to manage the sequence and dataflow between the many MILP subproblems that are generated.

Up to this point, we have not yet addressed how one goes about implementing the MILP planned trajectory such as the one shown Figure 5, on an actual flight trajectory. This is where we need to consider the interfaces between MILP and other components in the actual flight hardware and software.

In Section II we have briefly considered the time-discretization of a continuous time LTI system leading to equation (6). The issue that the time-discretization introduces is the lack of continuity between the points in the planned trajectory. As a result, the obstacle as well as other constraints that are specified in the problem formulation are only enforced at these collocation points in the planned trajectory, and not in between. In the case of obstacle avoidance, in particular, an optimal solution tends to be one that barely skirts the obstacles in order to minimize the objective function. Therefore, if one were to follow a series of straight-lines that connect the collocation points, one is likely collide with the obstacles. The solution to the problem lies in modelling the obstacles as larger than their actual sizes to account for the distance between the collocation points. This technique also addresses a similar issue that arises for a vehicle that is not a point mass and has been a standard technique in path planning literature [6]: each obstacle can be similarly expanded according to the dimensions of the vehicle.

Another fundamental issue that arises is the mechanism with which a vehicle follows the trajectory produced by the MILP problem. This issue is closely related to the selection of the length of period for time discretization, whose tradeoffs are an increased problem size for the same planning horizon vs. a sparser separation of trajectory points. The simplest way to implement a MILP trajectory is to have a waypoint follower track the planned trajectory points one after another. A more advanced technique would involve a nonlinear outer loop [7] that considers the equations of motion as well as the dynamics of the vehicle. We believe that a combination of this advanced nonlinear technique together with a MILP-based approach to path-planning provides the best overall closed-loop solution for autonomous NOE flight.

An important element of the interface between the MILP path planner and other system components is the input to the MILP path planner. Since the algorithm presented in this paper primarily performs the obstacle avoidance function, there must be means to incorporate obstacle data into a MILP-based path planner so that the appropriate MILP problem can be constructed. Most likely, the information would come from an off-line database, processed data from an optical sensor, or a combination of both. However, from the discussion in the previous section, we are well aware of the complexity challenge for a real-time MILP implementation, so it is paramount for any constraints to a MILP problem to be as relevant as possible. Therefore, there must be methods devised specifically to select the obstacles that are most relevant to the current MILP problem, that allows a tradeoff of problem size with levels of detail in modelling the environment.

VI. Experimental Setup and Results

Our experimental setup includes a scenario similar to the one shown in Figure 4, and the hardware consists of a 1.8GHz Linux PC and a 200MHz PowerPC 604 with 32MB of RAM running VxWorks real-time operating system. While the former is equipped with a standard CPLEX MILP solver, the latter uses an unreleased version of CPLEX ported to the VxWorks RTOS by ILOG. Multiple timescales is implemented as two loops: a long timescale (LT) loop running on the Linux PC and a short timescale (ST) loop running on the PowerPC. The Linux PC serves as the main computer for path planning purposes and provides ST subproblems for the PowerPC to solve. Communication is handled via TCP/IP through Ethernet, and since the amount of problem data sent over to the PowerPC and that of the solutions sent back from it are relatively small, solve time is still the dominant factor in determining the overall speed of the ST loop.

With a LT subproblem of 10 time steps at 0.5 second intervals, the solve time is on the order of 1 second. The ST subproblem has 10 time steps at 0.2 second intervals, and the solve time is on the order of 5 seconds. The obstacle counts for typical LT and ST subproblems are 20 and 10, respectively. Although the ST solve time is relatively long, it is consistent with the speed of the processor and the amount of memory available, when compared to the Linux setup. This is true even with consideration to the relative complexity of the LT and ST subproblems.

Although these solve times may be longer than required for immediate real-time applications, they are merely one order of magnitude away from the acceptable limit. With the rapid advancement of microprocessor, memory, and solver technologies, and with the use of parallel solvers as well as additional solver tuning, we could soon solve even more complex path planning problems in a true real-time environment.

VII. Conclusion

In this paper, we have presented a MILP obstacle avoidance path-planning algorithm with terrain flight capabilities. This problem setup serves as our motivating example for the computational techniques that we introduced. These techniques help reduce the computational intensity of a very large path planning problem by exploiting the structures of the path planning problem, breaking it down into multiple layers of subproblems that are far simpler to solve. The solutions of these subproblems are then combined together into a composite trajectory.

References

[1] J. Barraquand, B. Langlois, and J. C. Latombe. Numerical potential field techniques for robot path planning. In *IEEE Transactions on Systems, Man, and Cybernetics*, 1992.

[2] Yong K. Hwang and Narendra Ahuja. Gross motion planning - a survey. In *ACM Computing Surveys, Volume 24, Issue 3*, pages 560–570, September 1992.

[3] T. Schouwenaars, B. De Moor, E. Feron, and J. How. Mixed integer programming for multi-vehicle path planning. In *2001 European Control Conference*, September 2001.

[4] T. Schouwenaars, E. Feron, and J. How. Safe receding horizon path planning for autonomous vehicles. In *40th Annual Allerton Conference on Communication, Control, and Computing*, October 2002.

[5] J. Bellingham, A. Richards, and J. How. Receding horizon control of autonomous aerial vehicles. In *2002 American Control Conference*, May 2002.

[6] Toms Lozano-Prez and Michael A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. In *Communications of the ACM, Volume 22, Issue 10*, pages 560–570, October 1979.

[7] M.B. Milam. *Real-Time Optimal Trajectory Generation for Constrained Dynamical Systems*. PhD thesis, Department of Control and Dynamical Systems, California Institute of Technology, Pasadena, California,, 2003.

VIII. Acknowledgements

This work is partly supported by DARPA/AFRL under the Software Enabled Control program, contract number F33615-99-C-3613. Some methods presented in this paper have patents pending.