# Iterative MILP Methods for Vehicle-Control Problems

Matthew G. Earl, *Member, IEEE*, and Raffaello D'Andrea, *Senior Member, IEEE*

***Abstract*—Mixed-integer linear programming (MILP) is a powerful tool for planning and control problems because of its modeling capability and the availability of good solvers. However, for large models, MILP methods suffer computationally. In this paper, we present iterative MILP algorithms that address this issue. We consider trajectory-generation problems with obstacle-avoidance requirements and minimum-time trajectory-generation problems. These problems involve vehicles that are described by mixed logical dynamical equations, a form of hybrid system. The algorithms use fewer binary variables than standard MILP methods, and require less computational effort.**

***Index Terms*—Hybrid systems, mathematical optimization, mixed-integer linear programming (MILP), obstacle avoidance, path and trajectory planning, robot motion planning, trajectory generation.**

## I. Introduction

MIXED-integer linear programming (MILP) methods have attracted attention because of their modeling capability and because powerful solvers are available commercially. The use of MILP for modeling and control problems is described in [3]. MILP methods have been used for cooperative reconnaissance [16], air traffic management [17], coordinated robot motion [1], [18], unmanned aerial vehicle (UAV) path planning [5], and cooperative control [2], [9].

Powerful software packages such as CPLEX [14] can solve MILPs efficiently for problems in which the number of binary variables is of reasonable size. However, a major disadvantage of MILP is its computational complexity. Because MILP is NP-hard [12], computational requirements can grow significantly as the number of binary variables needed to model the problem increases. Motivated to generate efficient MILP problem formulations, we have developed several iterative techniques that require fewer binary variables than standard MILP methods.

The MILP obstacle-avoidance methods from [9] and [19] specify a uniformly distributed set of discrete times at which obstacle avoidance is enforced. We call this approach uniform gridding. In this approach, there is no avoidance guarantee between time steps. In addition, many of the avoidance times are unnecessary, resulting in large MILPs that require a significant computational effort to solve. Here, we present an iterative MILP obstacle-avoidance algorithm that can be used alone, or in combination with the uniform gridding approach. The algorithm guarantees obstacle avoidance over the entire trajectory and distributes avoidance times efficiently, resulting in smaller MILPs that can be solved faster. We also present an iterative MILP obstacle-growing algorithm that allows the use of a coarse set of uniformly distributed obstacle-avoidance times. In this approach, collision-free trajectories are found by artificially increasing the size of the obstacles that collide with the trajectory generated by the MILP, iterating until the resulting trajectory is collision-free. Note that an iterative MILP algorithm for the removal of redundant constraints in spacecraft trajectory-planning problems with avoidance constraints is described in [19].

Next, we consider the minimum-time trajectory-generation problem using MILP. The MILP approach to this problem presented in [20] and [21] generates an approximate solution. Time is discretized uniformly, and an auxiliary binary variable and a set of inequality constraints are added for each discrete time. This approach gives an estimate to the time-optimal solution that depends on the sampling time chosen. For more accuracy, the sampling time is reduced, which results in a larger number of binary variables in the MILP formulation, and thus increases the computation time, possibly exponentially. Here, we present an iterative MILP algorithm that solves for the time-optimal solution to the problem. The algorithm uses binary search. At each iteration, the feasibility of a MILP with only one discrete time (for the minimum-time part of the problem) needs to be determined.

The iterative methods presented in this paper apply to mixed logical dynamical (MLD) systems, which are governed by a mixture of logical rules (or state machines) and linear dynamical equations. This form of hybrid system is equivalent to piecewise affine systems, as shown in [13]. In this paper, we use an omnidirectional vehicle in an obstacle field, which is an MLD system, to motivate our methods. See [3] to learn how to represent various types of hybrid systems in MLD form.

The paper is organized as follows. In Section II, we describe the dynamics of the vehicles we use to motivate our methods. In Section III, we describe two iterative MILP algorithms for obstacle avoidance, and we perform an average case computational complexity study, comparing the performance of the iterative time-step selection algorithm with the uniform gridding approach. Finally, in Section IV, we describe an iterative MILP algorithm for minimum-time control problems. All files for generating the plots found in this paper are available online [10].

## II. Vehicle Dynamics

We motivate our methods using the wheeled robots of the Cornell RoboCup Team [6]. In this section, we show how to

Manuscript received December 6, 2004; revised April 18, 2005. This paper was recommended for publication by Associate Editor J. P. Merlet and Editor S. Hutchinson upon evaluation of the reviewers' comments. This work was supported by the Air Force Office of Scientific Research under Grant F49620-01-1-0361. This paper was presented in part at the 43rd IEEE Conference on Decision and Control, Paradise Island, Bahamas, December 2004.

M. G. Earl is with BAE Systems, Advanced Information Technologies, Burlington, MA 01803 USA (e-mail: matthew.earl@baesystems.com).

R. D'Andrea is with the Mechanical and Aerospace Engineering Department, Cornell University, Ithaca, NY 14853 USA (e-mail: rd28@cornell.edu).

Digital Object Identifier 10.1109/TRO.2005.853499

simplify their nonlinear governing equations using a procedure from [15]. The result is a linear set of governing equations coupled by a nonlinear constraint on the control input. This procedure allows real-time calculation of many near-optimal trajectories, and is a major factor of Cornell's success in the RoboCup competition. We then show how to represent the simplified system in a MILP problem formulation. The result is a set of linear discrete-time governing equations subject to a set of linear inequality constraints.

Each vehicle is equipped with a three-motor omnidirectional drive, which allows it to move along any direction, irrespective of its orientation. This allows superior maneuverability compared with traditional nonholonomic (car-like) vehicles. The nondimensional governing equations of each vehicle are given by

$$\begin{bmatrix} \ddot{x}(t) \\ \ddot{y}(t) \\ \ddot{\theta}(t) \end{bmatrix} + \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \frac{2mL^2}{I}\dot{\theta}(t) \end{bmatrix} = \mathbf{u}\left(\theta(t), t\right) \tag{1}$$

where $\mathbf{u}(\theta(t), t) = \mathbf{P}(\theta(t))\mathbf{U}(t)$

$$\mathbf{P}(\theta) = \begin{bmatrix} -\sin(\theta) & -\sin\left(\frac{\pi}{3} - \theta\right) & \sin\left(\frac{\pi}{3} + \theta\right) \\ \cos(\theta) & -\cos\left(\frac{\pi}{3} - \theta\right) & -\cos\left(\frac{\pi}{3} + \theta\right) \\ 1 & 1 & 1 \end{bmatrix}$$

and $\mathbf{U}(t) = (U_1(t), U_2(t), U_3(t)) \in \mathcal{U}$. In these equations, $(x(t), y(t))$ are the coordinates of the vehicle, $\theta(t)$ is its orientation, $\mathbf{u}(\theta(t), t)$ is the $\theta(t)$-dependent control input, $m$ is the mass of the vehicle, $I$ is its moment of inertia, $L$ is the distance from the drive to the center of mass, and $U_i(t)$ is the voltage applied to motor $i$. The set of admissible voltages $\mathcal{U}$ is the unit cube, and the set of admissible control inputs is given by $P(\theta)\mathcal{U}$.

These governing equations are coupled and nonlinear. To simplify them, we replace the set $P(\theta)\mathcal{U}$ with the maximal $\theta$-independent set found by taking the intersection of all possible sets of admissible controls. The result is a $\theta$-independent control set defined by control input $(u_x(t), u_y(t), u_\theta(t))$ and the inequality constraints $u_x(t)^2 + u_y(t)^2 \leq (3 - |u_\theta(t)|)^2/4$ and $|u_\theta(t)| \leq 3$. Using the restricted set as the allowable control set, the governing equations decouple, and are given by

$$\begin{bmatrix} \ddot{x}(t) \\ \ddot{y}(t) \\ \ddot{\theta}(t) \end{bmatrix} + \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \frac{2mL^2}{I}\dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} u_x(t) \\ u_y(t) \\ u_\theta(t) \end{bmatrix}. \tag{2}$$

The constraints on the control input couple the degrees of freedom.

To decouple the $\theta$ dynamics, we further restrict the admissible control set to a cylinder defined by the following two inequalities: $|u_\theta(t)| \leq 1$, and

$$u_x(t)^2 + u_y(t)^2 \leq 1. \tag{3}$$

Now, the equations of motion for the translational dynamics of the vehicle are given by

$$\begin{aligned} \ddot{x}(t) + \dot{x}(t) &= u_x(t) \\ \ddot{y}(t) + \dot{y}(t) &= u_y(t) \end{aligned} \tag{4}$$

subject to (3). In state-space form, (4) is $\dot{\mathbf{x}}(t) = \mathbf{A}_c\mathbf{x}(t) + \mathbf{B}_c\mathbf{u}(t)$, where $\mathbf{x} = (x, y, \dot{x}, \dot{y})$ is the state, and $\mathbf{u} = (u_x, u_y)$ is the control input.

TABLE I
VARIABLES DEFINITIONS

| | |
|---|---|
| $N_u$ | number of control input discretization steps |
| $M_u$ | number of inequalities defining the admissible control polygon |
| $N_o$ | number of obstacle avoidance times |
| $M_o$ | number of inequalities defining the polygon that approximates the obstacles |

To represent the governing equations in a MILP framework, we discretize the control input in time. We require the control input be constant between time steps. The result is a set of linear discrete-time governing equations, which we derive next. Variables are defined in Table I.

Let $N_u$ be the number of discretization steps for the control input $\mathbf{u}(t)$. Let $t_u[k]$ be the time at step $k$. Let $T_u[k] > 0$ be the time between steps $k$ and $k+1$ for $k \in \{0, \ldots, N_u - 1\}$. The discrete-time governing equations are given by

$$\mathbf{x}_u[k+1] = \mathbf{A}[k]\mathbf{x}_u[k] + \mathbf{B}[k]\mathbf{u}[k] \tag{5}$$

where $\mathbf{x}_u[k] = \mathbf{x}(t_u[k])$, $\mathbf{u}[k] = \mathbf{u}(t_u[k])$, $\mathbf{x}_u[k] = (x_u[k], y_u[k], \dot{x}_u[k], \dot{y}_u[k])$, and $\mathbf{u}[k] = (u_x[k], u_y[k])$. The coefficients $\mathbf{A}[k]$ and $\mathbf{B}[k]$ are functions of $k$, because we have allowed for nonuniform time discretizations. They can be calculated explicitly in the usual way [7]. Because there will be several different time discretizations used in this paper, we use subscripts to differentiate them. In this section, we use the subscript $u$ to denote variables associated with the discretization in the control input $\mathbf{u}(t)$.

The discrete-time governing equations can be solved explicitly in the usual way [7]. In later sections of this paper, it will be necessary to represent the position of the vehicle at times between control discretization steps in terms of the control input. Because the set of governing equations is linear, given the discrete state $\mathbf{x}_u[k]$ and the control input $\mathbf{u}[k]$, we can calculate the vehicle's state at any time $t$ using the following equations:

$$\begin{aligned} x(t) &= x_u[k] + \left(1 - e^{t_u[k]-t}\right)\dot{x}_u[k] \\ &\quad + \left(t - t_u[k] - 1 + e^{t_u[k]-t}\right)u_x[k] \\ \dot{x}(t) &= \left(e^{t_u[k]-t}\right)\dot{x}_u[k] + \left(1 - e^{t_u[k]-t}\right)u_x[k] \end{aligned} \tag{6}$$

where $k$ satisfies $t_u[k] \leq t \leq t_u[k+1]$. If the time discretization of the control input is uniform, $T_u[k_u] = T_u$ for all $k_u$, then $k_u = \lfloor t/T_u \rfloor$. The components of the vehicle's state, $y(t)$ and $\dot{y}(t)$, can be calculated in a similar way.

The control input constraint given by (3) cannot be expressed in a MILP framework because it is nonlinear. To incorporate this constraint, we approximate it with a set of linear inequalities that define a polygon. The polygon inscribes the region defined by the nonlinear constraint. We take the conservative inscribing polygon to guarantee that the set of allowable controls defined by the region is feasible. Similar to work in [20], we define the polygon by the set of $M_u$ linear inequality constraints

$$u_x[k]\sin\frac{2\pi m}{M_u} + u_y[k]\cos\frac{2\pi m}{M_u} \leq \cos\frac{\pi}{M_u} \quad \forall m \in \{1, \ldots, M_u\} \tag{7}$$

for each step $k \in \{1, \ldots, N_u\}$.

To illustrate the approach, consider the following minimum-control-effort trajectory-generation problem. Given a vehicle

governed by (5) and (7), find the sequence of control inputs $\{\mathbf{u}[k]\}_{k=0}^{N_u-1}$ that transfers the vehicle from starting state $\mathbf{x}(0) = \mathbf{x}_s$ to finishing state $\mathbf{x}(t_f) = \mathbf{x}_f$, and minimizes the cost function

$$J = \sum_{k=0}^{N_u-1} \left(|u_x[k]| + |u_y[k]|\right). \tag{8}$$

To convert the absolute values in the cost function to linear form, we introduce auxiliary continuous variables $z_x[k]$ and $z_y[k]$ and the inequality constraints

$$\begin{aligned} -z_x[k] \le u_x[k] \le z_x[k] \\ -z_y[k] \le u_y[k] \le z_y[k]. \end{aligned} \tag{9}$$

Minimizing $z_x[k]$, subject to the inequalities $u_x[k] \le z_x[k]$ and $u_x[k] \ge -z_x[k]$, is equivalent to minimizing $|u_x[k]|$ (similarly for $|u_y[k]|$) [4]. Using the auxiliary variables, the cost function can be written as a linear function

$$J = \sum_{k=0}^{N_u-1} \left(z_x[k] + z_y[k]\right). \tag{10}$$

The resulting optimization problem (minimize (10), subject to (5), (7), (9), and the boundary conditions) is in MILP form. Because binary variables do not appear in the problem formulation, it is a linear program and is easily solved to obtain the optimal sequence of control inputs.

## III. Obstacle Avoidance

In vehicle control, it is necessary to avoid other vehicles, stationary and moving obstacles, and restricted regions. In this section, we show how to use MILP to solve obstacle-avoidance problems, we present two iterative MILP obstacle-avoidance algorithms that are more computationally efficient than standard methods, and we perform an average case computational-complexity study.

### A. MILP Formulation

We start by showing a MILP method to guarantee circular obstacle avoidance at $N_o$ discrete times. A version of this method for uniformly distributed obstacle-avoidance times is presented in [9] and [19]. The method we present here allows nonuniform distributions of obstacle-avoidance times, which we take advantage of in our iterative algorithm presented in the next section. We use subscript $o$ to denote variables associated with the time discretization for obstacle avoidance. For step $k$, taken to be an element of the set $\{1, \ldots, N_o\}$, let $t_o[k]$ be the time at which obstacle avoidance is enforced. Let $R_{\text{obst}}$ denote the radius of the obstacle, and let $(x_{\text{obst}}[k], y_{\text{obst}}[k])$ denote the coordinates of its center at time $t_o[k]$. We approximate the obstacle with a polygon, denoted $\mathcal{O}[k]$, defined by a set of $M_o$ inequalities. The polygon is given by

$$\mathcal{O}[k] = \left\{ (\bar{x}, \bar{y}) : (\bar{x} - x_{\text{obst}}[k]) \sin\frac{2\pi m}{M_o} + (\bar{y} - y_{\text{obst}}[k]) \times \cos\frac{2\pi m}{M_o} \le R_{\text{obst}} \quad \forall m \in \{1, \ldots, M_o\} \right\}. \tag{11}$$

To guarantee obstacle avoidance at time $t_o[k]$, the coordinates of the vehicle must be outside the region $\mathcal{O}[k]$. This avoidance condition can be written as $(x_o[k], y_o[k]) \notin \mathcal{O}[k]$, where $(x_o[k], y_o[k])$ are the coordinates of the vehicle at time $t_o[k]$. Here $x_o[k] = x(t_o[k])$ and $y_o[k] = y(t_o[k])$ are expressed in terms of the control inputs using (6).

Because at least one constraint defining the region $\mathcal{O}[k]$ must be violated in order to avoid the obstacle, the avoidance condition is equivalent to the following condition. There exists an $m$ such that $(x_o[k] - x_{\text{obst}}[k]) \sin(2\pi m/M_o) + (y_o[k] - y_{\text{obst}}[k]) \cos(2\pi m/M_o) > R_{\text{obst}}$.

To express this avoidance constraint in a MILP problem formulation, it must be converted to an equivalent set of linear inequality constraints. We do so by introducing auxiliary binary variable $b_m[k] \in \{0, 1\}$ and the following $M_o$ inequality constraints:

$$(x_o[k] - x_{\text{obst}}[k]) \sin\frac{2\pi m}{M_o} + (y_o[k] - y_{\text{obst}}[k]) \cos\frac{2\pi m}{M_o} > R_{\text{obst}} - H b_m[k] \quad \forall m \in \{1, \ldots, M_o\} \tag{12}$$

where $H$ is a large positive number taken to be larger than the maximum dimension of the vehicle's operating environment plus the radius of the obstacle. If $b_m[k] = 1$, the right side of the inequality is a large, negative number that is always less than the left side. In this case, the inequality is inactive because it is trivially satisfied. If $b_m[k] = 0$, the inequality is said to be active, because it reduces to an inequality from the existence condition above. For obstacle avoidance, at least one of the constraints in (12) must be active. To enforce this, we introduce the following inequality constraint into the problem formulation:

$$\sum_{m=1}^{M_o} b_m[k] \le M_o - 1. \tag{13}$$

Therefore, to enforce obstacle avoidance at time $t_o[k]$, the set of binary variables $\{b_m[k]\}_{m=1}^{M_o}$ and the constraints given by (12) and (13) are added to the MILP problem formulation.

Consider the example problem from Section II, adding obstacles that must be avoided. In this problem, we want to transfer the vehicle from start state $\mathbf{x}_s$ to finish state $\mathbf{x}_f$ in time $t_f$ using minimal control effort and avoiding obstacles. To enforce obstacle avoidance at each time in the set $\{t_o[k]\}_{k=1}^{N_o}$, we augment the MILP formulation in Section II with the set of binary variables $\{b_m[k]\}_{m=1}^{M_o}$, constraints (12), and constraint (13) for all $k$ in the set $\{1, \ldots, N_o\}$.

Distributing the avoidance times uniformly (uniform gridding) results in a trajectory that avoids obstacles at each discrete time in the set. However, the trajectory can collide with obstacles between avoidance times. This is shown, for instance, in Fig. 1(a).

A simple method to reduce this behavior is to take a finer discretization, which increases the number of avoidance times, as shown in Fig. 1(b). However, this is not desirable in MILP, because an increase in the number of avoidance times increases the number of binary variables in the problem.

### B. Iterative MILP Time-Step Selection Algorithm

It is advantageous to use as few avoidance times as possible. Next, we propose an iterative algorithm to do so. The method

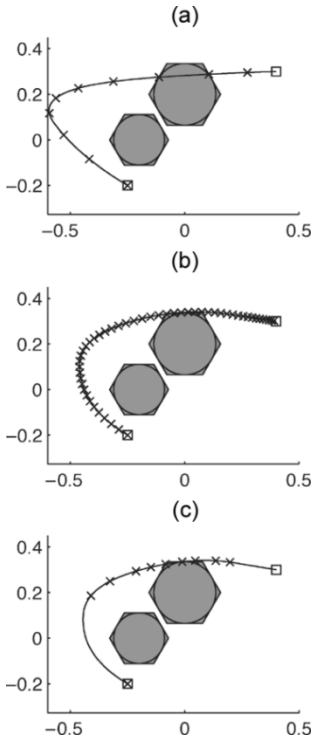


Fig. 1. (a) Resulting trajectory using uniform gridding ($N_o = 10$). (b) Trajectory using a finer uniform gridding ($N_o = 50$). (c) Trajectory using iterative MILP avoidance ($N_o = 9$). The circles denote obstacles, and the polygons denote the buffer regions used in the MILP formulation. The values of the parameters are $M_o = 6$, $M_u = 4$, $N_u = 4$, $(x_s, y_s, \dot{x}_s, \dot{y}_s) = (-0.25, -0.2, -0.5, 0.3)$, and $(x_f, y_f, \dot{x}_f, \dot{y}_f) = (0.4, 0.3, 0, 0)$.

distributes avoidance times where they are needed most, as shown in Fig. 1(c), and guarantees obstacle avoidance if an obstacle-free trajectory exists. The idea is to first solve the MILP with no obstacle-avoidance times (or with a coarse set of avoidance times), and check the resulting trajectory for collisions. Then, if there are collisions, augment the MILP formulation with an avoidance time (and the corresponding binary variables and constraints) for each collision. The avoidance time for each collision is taken from the interval of time that the trajectory is within the obstacle. Next, solve the augmented MILP and check the resulting trajectory for collisions, repeating the procedure until a collision-free trajectory is found.

The algorithm is outlined in Table II and proceeds as follows. First, formulate the vehicle-control problem as a MILP, and choose an initial set of avoidance times $\{t_o[k]\}_{k=1}^{N_o}$. This set is usually taken to be the empty set or a coarsely distributed set of times. Next, introduce a buffer zone for each obstacle $j$ with radius $R_{\text{buff}}^{(j)} = \alpha R_{\text{obst}}^{(j)}$, where $\alpha > 1$ is the buffer factor. Radius $R_{\text{buff}}^{(j)}$ is larger than $R_{\text{obst}}^{(j)}$ (usually taken slightly larger), and is used as the radius of obstacle $j$ in the MILP formulation. This is done to guarantee obstacle avoidance and termination of the algorithm, which we show later in this section. Next, solve the MILP using the buffer regions as the obstacles. Then, check the resulting trajectory for collisions (Appendix I) using each obstacle's true radius $R_{\text{obst}}^{(j)}$ for each obstacle $j$.

If there are no collisions, terminate the algorithm. Otherwise, for each collision $i$, compute the time interval $[t_1^{(i)}, t_2^{(i)}]$ in which the trajectory is within the obstacle. This interval

TABLE II
ITERATIVE MILP TIME-STEP SELECTION ALGORITHM

1: Formulate vehicle control problem as a MILP with the set of obstacle avoidance times $\{t_o[k]\}_{k=1}^{N_o}$.
2: Set obstacle buffer zone for each obstacle $j$, $R_{buff}^{(j)} := \alpha R_{obst}^{(j)}$ where $\alpha > 1$.
3: Solve MILP with obstacles of radius $R_{buff}^{(j)}$ for each obstacle $j$.
4: Check resulting trajectory for collisions with obstacles of radius $R_{obst}^{(j)}$ for each obstacle $j$.
5: **while** there are collisions **do**
6: For each collision $i$, compute time interval $[t_1^{(i)}, t_2^{(i)}]$.
7: For each collision $i$, augment the MILP formulation with obstacle avoidance constraints at time $t_o^{new} \in [t_1^{(i)}, t_2^{(i)}]$.
8: Solve augmented MILP with obstacles of radius $R_{buff}^{(j)}$ for each obstacle $j$.
9: Check resulting trajectory for collisions with obstacles of radius $R_{obst}^{(j)}$ for each obstacle $j$.
10: **end while**

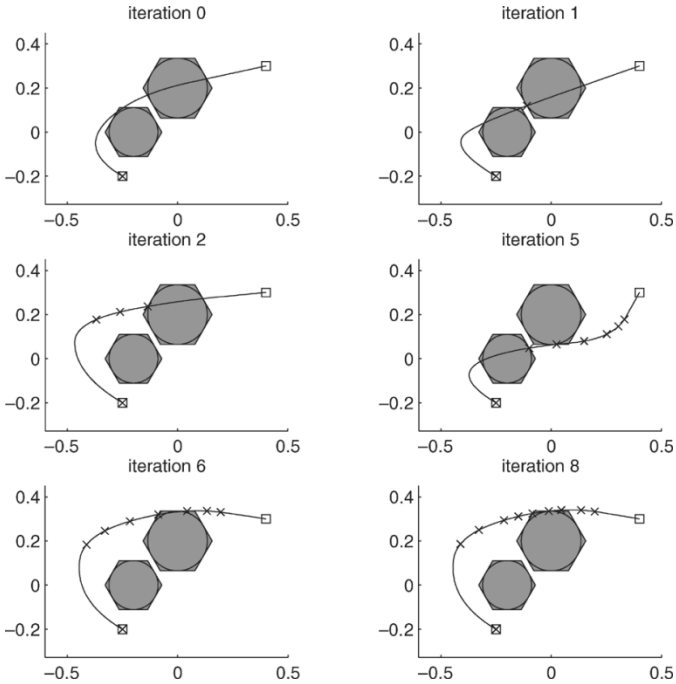


Fig. 2. Snapshots of the iterative MILP obstacle-avoidance algorithm. The circular regions denote obstacles, and the polygons denote the buffer regions used in the MILP formulation. Each cross "×" denotes a time at which obstacle avoidance is enforced. The values of the parameters are $M_o = 6$, $M_u = 4$, $N_u = 4$, $(x_s, y_s, \dot{x}_s, \dot{y}_s) = (-0.25, -0.2, -0.5, 0.3)$, and $(x_f, y_f, \dot{x}_f, \dot{y}_f) = (0.4, 0.3, 0, 0)$.

can be computed efficiently using a bisection routine and the collision-check routine. Then, for each collision $i$, augment the MILP problem formulation with avoidance constraints at time $t_o^{\text{new}}$, taken to be in the interval $[t_1^{(i)}, t_2^{(i)}]$. In this paper, we take $t_o^{\text{new}} = (t_1^{(i)} + t_2^{(i)})/2$. Next, solve the augmented MILP and check the resulting trajectory for collisions. If there are no collisions, terminate the algorithm. Otherwise, repeat the procedure until there are no collisions.

Snapshots of intermediate steps in the iterative algorithm are shown in Fig. 2. The procedure adds obstacle-avoidance points where they are needed most, thus avoiding unnecessary and computationally costly constraints and binary variables.

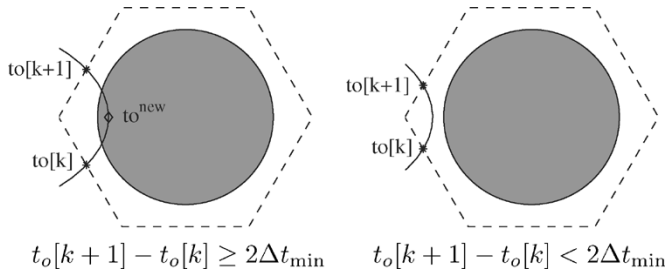


Fig. 3. These diagrams help show that the iterative MILP time-selection algorithm terminates. If the difference between consecutive obstacle-avoidance times, denoted $t_o[k+1] - t_o[k]$, is greater than $2\Delta t_{\min}$, the trajectory can intersect the obstacle as shown in the figure on the left. In this case, the algorithm will add a new avoidance constraint at time $t_o^{\text{new}}$. If the difference is less than $2\Delta t_{\min}$, the trajectory can not intersect the obstacle, and a new avoidance constraint does not need to be added in between.

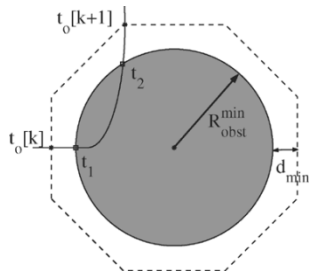


Fig. 4. This diagram helps show the minimum time step that can be added by the iterative MILP time-step selection algorithm. The trajectory intersects the obstacle in the interval $[t_1, t_2]$. Assume the vehicle is moving at $v_{\max}$ between times $t_o[k]$ and $t_1$. If the algorithm selects $t_1$ as the new obstacle-avoidance time, the difference $t_1 - t_o[k]$ is equal to $\Delta t_{\min}$. This is the minimum possible time interval between avoidance times, because the vehicle can not move any faster.

Now we show that the iterative algorithm in Table II terminates. The minimum distance between the boundary of a buffer zone and the boundary of the obstacle it surrounds is $d = R_{\text{buff}} - R_{\text{obst}} = (\alpha - 1)R_{\text{obst}}$. For a problem involving multiple obstacles, the minimum of these distances is given by $d_{\min} = (\alpha - 1)R^{\min}_{\text{obst}}$, where $R^{\min}_{\text{obst}}$ is the radius of the smallest obstacle in the environment. The minimum time it takes the vehicle to travel between the boundary of a buffer zone and its corresponding obstacle is given by $\Delta t_{\min} = d_{\min}/v_{\max} = (\alpha - 1)R^{\min}_{\text{obst}}/v_{\max}$, where $v_{\max}$ is the maximum velocity of the vehicle. Consider two consecutive obstacle-avoidance times, denoted $t_o[k]$ and $t_o[k+1]$. The vehicle must be located outside all buffer zones at these two times, because we have enforced this as a hard constraint in the MILP. If the difference $t_o[k+1] - t_o[k]$ is less than $2\Delta t_{\min}$, the vehicle's trajectory can not intersect the obstacle, because there is not enough time to enter the buffer zone, collide with the obstacle, then exit the buffer zone (see Fig. 3). In order for the trajectory to intersect the obstacle in the interval between these two times, the difference $t_o[k+1] - t_o[k]$ must be greater than $2\Delta t_{\min}$. In summary, the algorithm will not add an obstacle-avoidance time in the interval if $t_o[k+1] - t_o[k] < 2\Delta t_{\min}$, but it can add an obstacle-avoidance time if $t_o[k+1] - t_o[k] \geq 2\Delta t_{\min}$. Therefore, in the worst case, once the algorithm reaches a point where the time interval between each obstacle-avoidance time is less than $2\Delta t_{\min}$, the algorithm must terminate.

Next we bound the number of steps it takes for the algorithm to terminate. The smallest possible time interval between consecutive obstacle-avoidance times is $\Delta t_{\min}$. This can be seen by looking at Fig. 4, where $t_o[k+1]$ and $t_o[k]$ are two consecutive avoidance times, $t_1$ is the time at which the trajectory enters the obstacle, and $t_2$ is the time it exits the obstacle. Suppose the vehicle is moving at its maximum velocity from time $t_o[k]$ to $t_1$. The algorithm will detect this intersection, compute times $t_1$ and $t_2$, and pick a new obstacle-avoidance time $t_o^{\text{new}}$ in the interval $[t_1, t_2]$. Suppose the algorithm picks $t_o^{\text{new}} = t_1$, then $t_o^{\text{new}} - t_o[k] = \Delta t_{\min}$. The time interval can not be any less, because the vehicle can not pass through the buffer zone in time less than $\Delta t_{\min}$. In the trajectory-generation problem, if $t_s$ is the vehicle's starting time and $t_f$ is its finishing time, the maximum number of time intervals added by the algorithm is $\lfloor (t_f - t_s)/\Delta t_{\min} \rfloor$. Therefore, the algorithm will terminate in a maximum of $\lfloor (t_f - t_s)/\Delta t_{\min} \rfloor = \lfloor (t_f - t_s)v_{\max}/((\alpha - 1)R^{\min}_{\text{obst}}) \rfloor$ steps.

This is a worst-case bound on the number of steps, and increases significantly as $\alpha$ is decreased toward one. However, in practice, the number of steps does not appear to increase significantly as $\alpha$ approaches one. For example, for the case with three obstacles, we varied $\alpha$ from 1.001 to 2.5. For each value of $\alpha$ tested, we solved 100 randomly generated instances of the problem. The median number of steps required for algorithm termination was 2 for all values of $\alpha$ tested. The minimum average number of steps was 1.74 for $\alpha = 2.3$, and the maximum average number of steps was 2.26 for $\alpha = 1.01$.

Upon termination of the algorithm, there are three possible results. (1) If a path around the buffer zones to the goal exists, the algorithm outputs a feasible path to the goal. (2) If a path around the obstacles to the goal does not exist, the algorithm indicates the infeasibility. In this case, the final MILP before termination is infeasible, which is indicated by the MILP solver. (3) If a path around the obstacles to the goal exists, but a path around the buffer zones to the goal does not, the algorithm either outputs a feasible path to the goal or terminates with an infeasible MILP. In this rarely encountered case, there is no guarantee that the algorithm will find a feasible path to the goal.

### C. Iterative MILP Obstacle-Growing Algorithm

Being consistent with our goal to reduce the number of obstacle-avoidance times in our MILP problem formulations, we propose another iterative MILP algorithm for obstacle avoidance. This algorithm iteratively grows the buffer zones surrounding the obstacles until a collision-free trajectory is found. The idea is to first solve the MILP with a coarse set of avoidance times, and an initial set of buffer zones surrounding each obstacle. Then, check the resulting trajectory for collisions. If there are collisions, increase the size of each buffer zone that surrounds an obstacle with which the trajectory collides. Next, solve the MILP with these new buffer zones, and check the resulting trajectory for collisions. This process is repeated until there are no collisions.

The details of the algorithm are listed in Table III. Snapshots of intermediate steps of the algorithm are shown in Fig. 5. The crosses denote the coarse set of times at which obstacle avoidance is enforced in the MILP. As the figure shows, the size of

TABLE III
ITERATIVE MILP OBSTACLE-GROWING ALGORITHM

1: Formulate vehicle control problem as a MILP with the set of obstacle avoidance times $\{t_o[k]\}_{k=1}^{N_o}$.
2: Set obstacle buffer zone for each obstacle $j$, $R_{buff}^{(j)} := \alpha R_{obst}^{(j)}$ where $\alpha > 1$.
3: Solve MILP with obstacles of radius $R_{buff}^{(j)}$ for each obstacle $j$.
4: Check resulting trajectory for collisions with obstacles of radius $R_{obst}^{(j)}$ for each obstacle $j$.
5: **while** there are collisions **do**
6: For each obstacle $j$ that collides with the trajectory, increase buffer region by setting $R_{buff}^{(j)} := \alpha R_{buff}^{(j)}$.
7: Solve MILP with obstacles of radius $R_{buff}^{(j)}$ for each obstacle $j$.
8: Check resulting trajectory for collisions with obstacles of radius $R_{obst}^{(j)}$ for each obstacle $j$.
9: **end while**

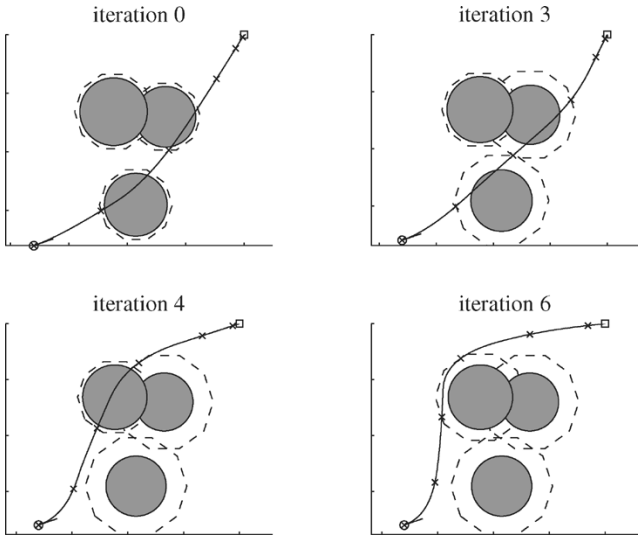


Fig. 5. Snapshots of the iterative MILP obstacle-growing algorithm. The circular regions denote obstacles, and the polygons denote the buffer regions used in the MILP formulation. Each cross denotes a time at which obstacle avoidance is enforced. For each iteration, there are five times at which obstacle avoidance is enforced ($N_o = 5$).

the buffer regions surrounding the obstacles with which the trajectory intersects is increased until the resulting trajectory, generated by the MILP, avoids all obstacles.

If the buffer zones become very large, this method will have performance issues, as shown in Fig. 6(d), where the resulting trajectory is suboptimal, compared with those generated by the other methods. The situation in which this algorithm is most useful is when uniform gridding is used and the resulting trajectory clips an obstacle, barely intersecting it. In this case, the algorithm will push the trajectory away from the clipped obstacle in a few iterations, resulting in a collision-free trajectory. However, if the initial distribution of avoidance times is too coarse, the algorithm could have problems. In this case, the buffer regions could grow to be large and engulf the initial or final position, which results in an infeasible MILP.

The benefits of the growing algorithm over the time-step selection algorithm occur in cases where we already have a large number of obstacle-avoidance constraints, and we do not want

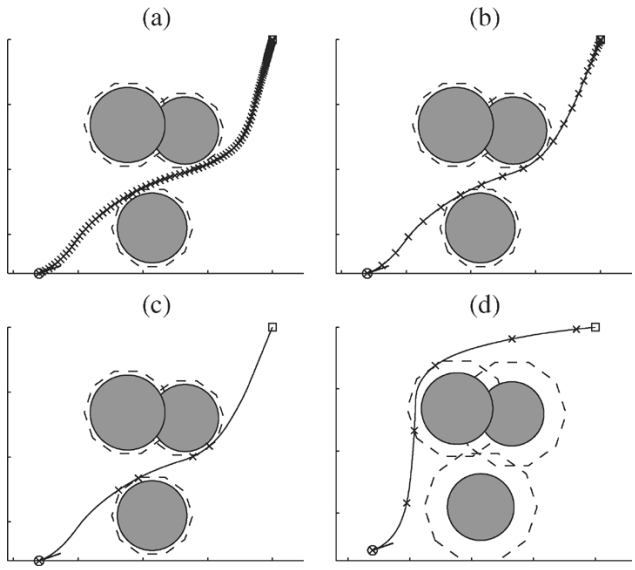


Fig. 6. Solutions to an instance of the obstacle-avoidance problem. (a) Using uniform gridding with sample time $2\Delta t_{\min}$ ($N_o = 115$). (b) Using uniform gridding with sample time $\Delta t_c$ ($N_o = 25$). (c) Using iterative time-step selection ($N_o = 4$). (d) Using iterative obstacle growing ($N_o = 5$). The straight-line segment denotes the initial velocity $(\dot{x}_s, \dot{y}_s) = (0.497, 0.172)$, the circular regions denote obstacles, the dashed regions denote the polygonal buffer zones, and each cross denotes a time along the trajectory at which obstacle avoidance is enforced.

to add any more. For such cases, the growing algorithm can generate a feasible path without adding any new constraints to the problem.

### D. Performance Analysis

In this section, we explore the average case computational complexity of the iterative MILP time-step selection algorithm by solving randomly generated problem instances. Each instance is generated by randomly picking parameters from a uniform distribution over the intervals defined below. Each MILP is solved using AMPL [11] and CPLEX [14] on a PC with Intel PIII 550 MHz processor, 1024 KB cache, 3.8 GB RAM, and Red Hat Linux.

For comparison, we solve the same instances using uniform gridding with sample time $\Delta t_c = 2R_{\text{obst}}^{\min}\sqrt{\alpha^2 - 1}/v_{\max}$. This sample time is the maximum sample time that guarantees obstacle avoidance, assuming the vehicle travels in a straight line between sample times. This is a good approximation, since $\Delta t_c$ is small for the instances we solve. See Appendix II for details. Each obstacle-avoidance time is given by $t_o[k] = k\Delta t_c$, where $k = 1, \ldots, N_o$ and $N_o = \lceil t_f/\Delta t_c \rceil$.

The instances are generated as follows. The start state is taken to be $\mathbf{x}_s = (x_s, y_s, r_v \cos\theta_v, r_v \sin\theta_v)$, where $(x_s, y_s)$ is constant, and $r_v$ and $\theta_v$ are random variables chosen uniformly from the intervals $[r_v^{\min}, r_v^{\max}]$ and $(0, 2\pi]$, respectively. The final state is fixed with zero velocity, $\mathbf{x}_f = (x_f, y_f, 0, 0)$. We generate $N_{\text{obst}}$ obstacles each with position $(x_{\text{obst}}, y_{\text{obst}}) = (r\cos\theta, r\sin\theta)$ and radius $R_{\text{obst}}$. The parameters $R_{\text{obst}}$, $r$, and $\theta$ are random variables chosen uniformly from the respective intervals $[R_{\min}, R_{\max}]$, $[r_{\min}, r_{\max}]$, and $(0, 2\pi]$, such that no obstacle overlaps the circle of radius $R_s$ with position $(x_s, y_s)$ or the circle of radius $R_f$ with position $(x_f, y_f)$.

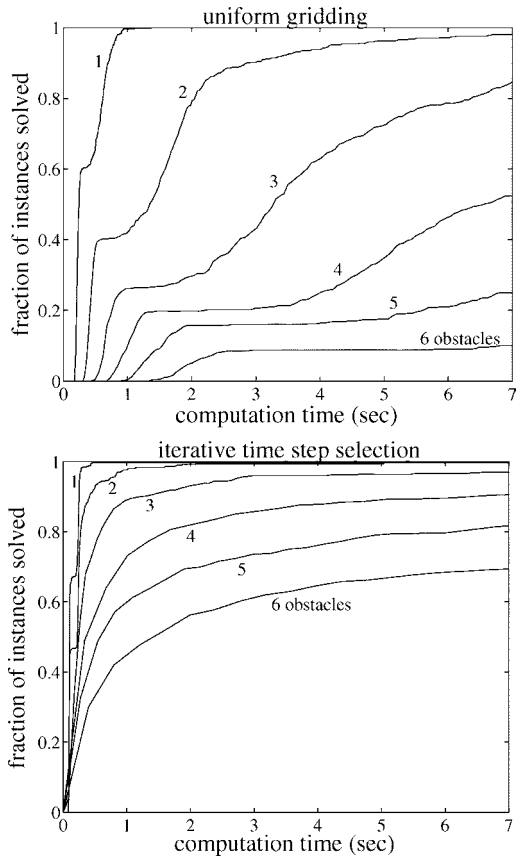


Fig. 7. Fraction of instances solved versus computation time for uniform gridding (top) and the iterative MILP time-step selection algorithm (bottom). We consider $N_{\text{obst}} = 2, 3, 4, 5, 6$. For each curve, 500 random instances were solved. The values of the parameters are $N_u = 10$, $M_u = 10$, and $M_o = 10$.

For the instances generated in this paper, we set the intervals to be $r_v \in [0.5, 1.0]$, $R_{\text{obst}} \in [0.2, 0.3]$, and $r \in [0.0, 1.0]$. The constant parameters are taken to be $(x_s, y_s) = (-0.8, -0.8)$, $(x_f, y_f) = (1.0, 1.0)$, $R_s = 0.5$, and $R_f = 0.1$.

The solution to an instance of the obstacle-avoidance problem with three obstacles is shown in Fig. 6 for the uniform gridding method and for the iterative MILP methods. Each cross denotes the time along the trajectory at which obstacle avoidance is enforced. The uniform gridding method with sample time $\Delta t_c$ requires $N_o = 25$ obstacle-avoidance times, shown in Fig. 6(b), while the iterative MILP time-step selection algorithm requires only $N_o = 4$ avoidance times, shown in Fig. 6(c). Notice the efficiency with which the iterative algorithm distributes the avoidance times. For comparison, we also solve this instance using uniform gridding with sample time $2\Delta t_{\min}$ [Fig. 6(a)] and using the iterative obstacle-growing algorithm [Fig. 6(d)]. For uniform gridding, choosing sample time $2\Delta t_{\min}$ guarantees obstacle avoidance, as discussed in Section III-B. However, as shown in the figure, this dense set of obstacle-avoidance times is very conservative.

In Fig. 7, we plot the fraction of instances solved versus computation time for the two methods. As these figures show, the iterative MILP method is less computationally intensive than the uniform gridding method for the instances solved. For example, 70% of the instances are solved in 0.4 s or less using the iterative MILP algorithm for the three-obstacle case. In contrast, no instances are solved in 0.4 s or less using uniform gridding for the three-obstacle case.

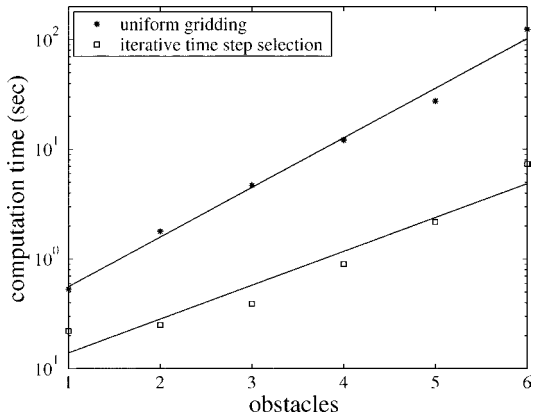


Fig. 8. Computation time necessary to solve 70% of the randomly generated instances versus the number of obstacles. Each square denotes a data point for the uniform gridding method. Each asterisk denotes a data point for the iterative MILP method. The solid lines denote curves fitted to these data.

In Fig. 8, we plot the computation time necessary to solve 70% of the randomly generated instances versus the number of obstacles on the field. Data are plotted for the uniform gridding method and the iterative MILP method. The computational requirements for both methods grow exponentially with the number of obstacles. However, as the figure shows, the iterative MILP method is less computationally intensive, and the computation time grows at a slower rate.

## IV. Minimum-Time Problems

In this section, we present an iterative MILP algorithm for solving minimum-time problems using a vehicle trajectory-generation problem as motivation. In [20] and [21], MILP methods for this problem are presented. Time is discretized uniformly, and the sampling interval that contains the optimal time is found using MILP. To get better bounds on the optimal time, the sample time of the discretization must be reduced, which results in a larger number of binary variables. In Appendix III, this method is outlined in the context of the vehicle considered in this paper.

Here we propose an iterative algorithm that converges to the optimal time using binary search. At each iteration, the feasibility of a MILP is determined using a solver such as CPLEX [14]. In each MILP, the number of binary variables and the number of constraints are much fewer than those for other techniques, because only one discrete time step is needed.

To motivate the iterative algorithm, we consider a minimum-time vehicle-control problem in an obstacle field. Given a vehicle governed by (5) and (7) and subject to obstacle-avoidance constraints (12) and (13), find the sequence of control inputs $\{\mathbf{u}[k]\}_{k=0}^{N_u-1}$ that transfers the vehicle from initial state $\mathbf{x}(0) = \mathbf{x}_s$ to final state $\mathbf{x}(t_f) = \mathbf{x}_f$ in minimum time.

Suppose we know that the optimal time, denoted $t^*$, is within the time interval $(t_L, t_R]$. Let time $t_M = (t_L + t_R)/2$. Consider the MILP given by (5), (7), constraint $\mathbf{x}(0) = \mathbf{x}_s$, and constraint $\mathbf{x}(t_f) = \mathbf{x}_f$, with the final time taken to be $t_f = t_M$. We use (6) to express $\mathbf{x}(t_f)$ in terms of the control inputs. To determine if there exists a sequence of control inputs that transfers the vehicle

TABLE IV
ITERATIVE MINIMUM-TIME MILP ALGORITHM

1: Formulate problem as a MILP without objective function.
2: Set $t_L := t_{lb}$ and $t_R := t_{ub}$.
3: Set $t_M := (t_R + t_L)/2$.
4: **while** $(t_R - t_L) > \epsilon$ **do**
5: Determine feasibility of MILP with final time $t_f = t_M$.
6: **if** feasible **then** set $t_R := t_M$.
7: **else** set $t_L := t_M$.
8: Set $t_M := (t_R + t_L)/2$.
9: **end while**

from start state to finish state, we solve the MILP without an objective function (this is a feasibility problem).

If the MILP is feasible, $t^*$ must be within the interval $(t_L, t_M]$. Otherwise, the MILP is infeasible, and $t^*$ must be within the interval $(t_M, t_R]$. By determining the feasibility of the MILP, we have cut the bound on the optimal time $t^*$ in half. This suggests an iterative binary-search procedure that converges to $t^*$.

The iterative algorithm is outlined in Table IV, and proceeds as follows. First, pick a time interval $(t_{\text{lb}}, t_{\text{ub}}]$ that bounds the optimal time $t^*$. The lower bound is taken to be $t_{\text{lb}} = d_{\min}/v_{\max}$, where $d_{\min}$ is the straight-line distance from the initial position to the final position, and $v_{\max}$ is the maximum velocity of the vehicle. The upper bound is taken to be a feasible time in which the vehicle can reach the destination. A simple way to compute a feasible time is to try time $\alpha t_{\text{lb}}$, where $\alpha > 1$, increasing $\alpha$ until a feasible time is found. Set $t_L := t_{\text{lb}}$, $t_R := t_{\text{ub}}$, and $t_M := (t_R + t_L)/2$.

Next, set the final time in the MILP problem formulation to be $t_f = t_M$, and determine if the resulting MILP is feasible using the MILP solver. If the MILP is feasible, the optimal time $t^*$ must be within the interval $(t_L, t_M]$. In this case, set $t_R := t_M$. Otherwise, the MILP is infeasible and the vehicle can not reach the destination in time $t_M$. The optimal time $t^*$ must be within the interval $(t_M, t_R]$. In this case, set $t_L := t_M$. Then, update $t_M$ by setting $t_M := (t_R + t_L)/2$. If the difference $t_R - t_L$ is less than some desired tolerance for our calculation of $t^*$, denoted $\epsilon$, the algorithm terminates. Otherwise, repeat the process by setting the final time to $t_f = t_M$ and continue with the steps outlined previously until the computed value of $t^*$ is within the desired tolerance $\epsilon$.

After the $k$th iteration, the time interval containing optimal time $t^*$ has length $(t_{\text{ub}} - t_{\text{lb}})/2^k$.

The solid lines of Fig. 9 show the solution to an instance of the minimum-time problem without obstacles (note that for the case without obstacles, the MILP for the iterative approach reduces to an LP). The iterative procedure was stopped after 13 iterations, which took approximately 1 s on our Pentium III 550 MHz computer. To achieve the same accuracy using the uniform time discretization method, solving one large MILP with a small sampling time, it took 5 min on the same computer.

Our iterative procedure converges to the time-optimal solution of the problem stated in the beginning of this section. This solution is an approximate solution to the continuous-time version of the minimum-time vehicle-control problem. In the continuous-time version of the problem without obstacles, the vehicle is governed by (4) and (3). We wish to transfer the vehicle

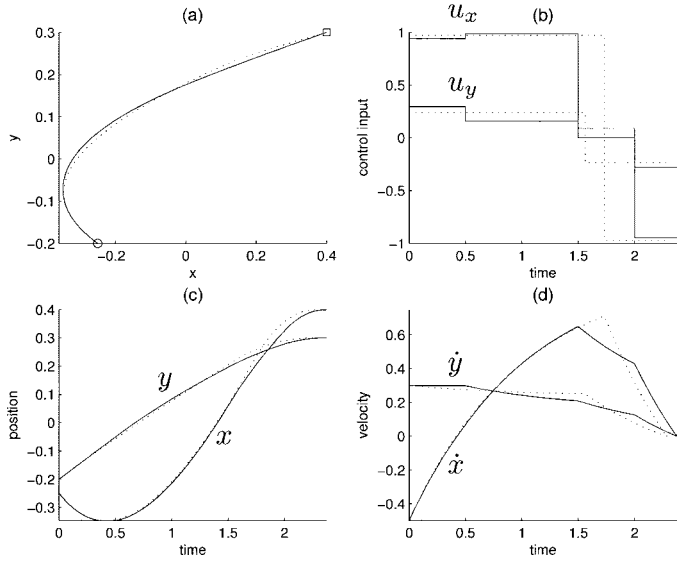


Fig. 9. Approximate time-optimal solution (solid lines) to an instance of the minimum-time vehicle-control problem of Section IV given by the iterative MILP algorithm. For comparison, we plot the near-optimal solution (dotted lines) for the continuous-time version of the problem obtained using techniques from [15]. The parameters are $M_u = 20$, $N_u = 10$, $(x_s, y_s, \dot{x}_s, \dot{y}_s) = (-0.25, -0.2, -0.5, 0.3)$, $(x_f, y_f, \dot{x}_f, \dot{y}_f) = (0.4, 0.3, 0, 0)$.

from starting state $\mathbf{x}_s$ to finishing state $\mathbf{x}_f$ in minimum time. In Fig. 9, we compare our near-optimal solution with the continuous-time problem (solid lines) to another technique (dotted lines) for generating near-optimal solutions from [15], which was used successfully in the RoboCup competition.

In addition to being used on its own, our iterative approach can be combined with the uniform discretization approach. In this case, the uniform approach is run first with a coarse discretization (large sampling time $T$). The output is a time interval of size $T$, which contains the optimal time $t^*$. We use this time interval as the input to our iterative algorithm. The $k$th step of the iterative algorithm outputs a time interval of length $T/2^k$ containing the optimal time $t^*$, and thus quickly converges to the optimal time.

## V. DISCUSSION

We have presented iterative MILP algorithms for obstacle avoidance and for minimum-time control problems. The iterative MILP time-selection algorithm picks obstacle-avoidance times, and intelligently distributes them where they are needed most. The iterative MILP obstacle-growing algorithm allows a coarse set of obstacle-avoidance times to be used instead of a dense distribution, which is required to guarantee obstacle avoidance for standard MILP methods. Both of these algorithms reduce the number of binary variables needed to formulate and solve obstacle-avoidance trajectory-generation problems using MILP. To demonstrate the computational benefits of the iterative MILP time-step selection algorithm, we performed an average case computational-complexity analysis. For comparison, we also performed the analysis on the standard uniform gridding method. The iterative algorithm significantly outperformed the uniform gridding method. In addition, we also present an iterative algorithm for solving minimum-time problems using MILP.

We found that the algorithm significantly outperforms standard techniques for minimum-time problems using MILP.

Due to the reduced computational requirements of these methods, they can be applied more widely in practice. Computational efficiency is especially important for real time control in dynamically changing environments, where new control plans need to be generated often and in real time, using a strategy such as model predictive control. In our research, we use these methods to solve cooperative control problems [7]. However, there is much room for improvement, including decreasing computation time further and developing methods that scale better with increased numbers of obstacles and vehicles. We feel that intelligent time-step selection methods, such as those presented in this paper, can be very useful in reducing computational requirements and should be pursued further. One aspect that needs inspection is the intelligent selection of the discretization for the control input to the vehicle.

## APPENDIX I

Here we present a collision-detection method that is guaranteed to find all collisions. It is conservative, and may determine false positives if a trajectory is very close to an obstacle.

Form a collision-check buffer zone of size $\alpha_c R_{\text{obst}}^{(i)}$, where $\alpha_c > 1$, around each obstacle $i$. Discretize the trajectory in time with sample time less than $2\Delta t_{\min}^c$, where $\Delta t_{\min}^c = (\alpha_c - 1)R_{\text{obst}}^{\min}/v_{\max}$. For each sample, check for collisions with the buffered obstacles. Because $2\Delta t_{\min}^c$ is a lower bound on the minimum time for the vehicle to travel from the buffer zone to the obstacle and back out of the buffer zone (similar to arguments in Section III-B), the method is guaranteed to find all collisions if the sampling time is less than this. However, the downside of this approach is that it may return false positives if the trajectory intersects the buffer zone but not the obstacle. To reduce this, $\alpha_c$ must be decreased, which increases the number of samples. We have developed an efficient nonconservative approach to collision detection for our vehicles, but its description is beyond the scope of this paper.

## APPENDIX II

Here we derive the minimum sample time, denoted $\Delta t_c$, that guarantees obstacle avoidance between sample times, assuming the vehicle moves in a straight-line path between sample times. This is a good approximation, because $\Delta t_c$ is small for the problems we solve.

Let $d$ be the straight-line distance the vehicle can travel between any two consecutive avoidance times. The chord of the smallest buffer region that is tangent to the obstacle it surrounds is denoted the critical chord. The critical chord length is given by $d_c = 2((R_{\text{buff}}^{\min})^2 - (R_{\text{obst}}^{\min})^2)^{1/2} = 2R_{\text{obst}}^{\min}\sqrt{\alpha^2 - 1}$, because $R_{\text{buff}}^{\min} = \alpha R_{\text{obst}}^{\min}$.

If $d < d_c$, the vehicle is guaranteed to avoid the obstacle between avoidance times. If $d \geq d_c$, the vehicle can collide with the obstacle between avoidance times. The critical time interval is given by $\Delta t_c = d_c/v_{\max} = (2R_{\text{obst}}^{\min}\sqrt{\alpha^2 - 1})/v_{\max}$, where $v_{\max}$ is the maximum velocity of the vehicle.

## APPENDIX III

Here we consider a minimum-time trajectory-generation problem using a uniform gridding MILP approach. We are given a vehicle governed by the discrete time system (5) and subject to the constraints (7). The objective is to find the sequence of control inputs $\{\mathbf{u}[k]\}_{k=0}^{N_u-1}$ that transfers the system from the initial state $\mathbf{x}(0) = \mathbf{x}_s$ to the final state $\mathbf{x}(t_f) = \mathbf{x}_f$ in minimum time.

Applying the techniques of [20] and [21], we introduce a uniform time discretization with constant sampling time $T$. The solution of the resulting MILP gives a feasible time that is within $T$ of the optimal time. Discretize time into $N_T$ times given by $t_T[k] = kT$, where $k$ is an element of the set $\{1, \ldots, N_T\}$. The discretization must be chosen so that $t_T[N_T] = N_T T$ is larger than the optimal time. Next, introduce auxiliary binary variable $\delta[k] \in \{0, 1\}$ and the inequality constraints

$$\begin{aligned}
x\left(t_T[k]\right) - x_f &\leq H\left(1 - \delta[k]\right) \\
x\left(t_T[k]\right) - x_f &\geq -H\left(1 - \delta[k]\right) \\
y\left(t_T[k]\right) - y_f &\leq H\left(1 - \delta[k]\right) \\
y\left(t_T[k]\right) - y_f &\geq -H\left(1 - \delta[k]\right) \\
\dot{x}\left(t_T[k]\right) - \dot{x}_f &\leq H\left(1 - \delta[k]\right) \\
\dot{x}\left(t_T[k]\right) - \dot{x}_f &\geq -H\left(1 - \delta[k]\right) \\
\dot{y}\left(t_T[k]\right) - \dot{y}_f &\leq H\left(1 - \delta[k]\right) \\
\dot{y}\left(t_T[k]\right) - \dot{y}_f &\geq -H\left(1 - \delta[k]\right)
\end{aligned} \tag{14}$$

for each $k$ in the set $\{1, \ldots, N_T\}$. Here, the state $\mathbf{x}(t_T[k])$ is written in terms of the control inputs using (6), and $H$ is a large positive constant taken to be greater than the largest dimension of the operating environment.

If $\delta[k] = 0$, every constraint in (14) is trivially satisfied because, for example, $x(t_T[k]) - x_f$ is always less than $H$. Otherwise, $\delta[k] = 1$, and the constraints in (14) enforce the condition $\mathbf{x}(t_T[k] = \mathbf{x}_f)$. To require that the final condition be satisfied at only one discrete time $t_T[k]$, the following constraint is introduced: $\sum_{i=1}^{N_T} \delta[i] = 1$.

Finally, we introduce the cost function to be minimized, $J = \sum_{i=1}^{N_T} i\delta[i]$. By minimizing this cost, the final state $\mathbf{x}_f$ is reached at the earliest discrete time possible, $t_T[k]$. The output after solving the resulting MILP is a single $k_{\text{sol}}$ such that $\delta[k_{\text{sol}}] = 1$. The optimal time is, therefore, within the interval $(t_T[k_{\text{sol}} - 1], t_T[k_{\text{sol}}])$.

## ACKNOWLEDGMENT

The authors would like to thank the reviewers for providing many valuable comments.

## REFERENCES

[1] S. Akella and S. Hutchinson, "Coordinating the motions of multiple robots with specified trajectories," in *Proc. IEEE Int. Conf. Robot. Autom.*, Washington, DC, May 2002, pp. 624–631.

[2] J. S. Bellingham, M. Tillerson, M. Alighanbari, and J. P. How, "Cooperative path planning for multiple UAVs in dynamic and uncertain environments," in *Proc. IEEE Conf. Decision Control*, Las Vegas, NV, Dec. 2002, pp. 2816–2822.

[3] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, pp. 407–428, 1999.

[4] D. Bertsimas and J. N. Tsitsiklis, *Introduction to Linear Optimization*. Belmont, MA: Athena Scientific, 1997.

[5] A. I. Chaudhry, K. M. Misovec, and R. D'Andrea, "Low observability path planning for an unmanned air vehicle using mixed integer linear programming," in *Proc. IEEE Conf. Decision Control*, Paradise Island, Bahamas, Dec. 2004, pp. 3823–3829.

[6] R. D'Andrea, T. Kalmár-Nagy, P. Ganguly, and M. Babish, "The Cornell RoboCup Team," in *Robot Soccer WorldCup IV, Lecture Notes in Artificial Intelligence*, G. Kraetzschmar, P. Stone, and T. Balch, Eds. New York: Springer, 2001.

[7] M. G. Earl and R. D'Andrea. Multi-Vehicle Cooperative Control Using Mixed Integer Linear Programming. [Online] Preprint available: http://arxiv.org/abs/cs.RO/0501092

[8] ——, A Decomposition Approach to Multi-Vehicle Cooperative Control. [Online] Preprint available: http://arxiv.org/abs/cs.RO/0504081

[9] ——, "Modeling and control of a multi-agent system using mixed integer linear programming," in *Proc. IEEE Conf. Decision Control*, Las Vegas, NV, Dec. 2002, pp. 107–111.

[10] All files for generating the plots found in this paper are available online by following the link. [Online]http://arxiv.org/abs/cs.RO/0505042 or http://control.mae.cornell.edu/earl/milp2

[11] R. Fourer, D. M. Gay, and B. W. Kernighan, *AMPL-A Modeling Language for Mathematical Programming*. Boston, MA: Boyd & Fraser, 1993.

[12] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: Freeman, 1979.

[13] W. P. M. H. Heemels, B. De Schutter, and A. Bemporad, "Equivalence of hybrid dynamical models," *Automatica*, vol. 37, no. 7, pp. 1085–1091, Jul. 2001.

[14] *ILOG AMPL CPLEX System Version 7.0 User's Guide*, 2000.

[15] T. Kalmár-Nagy, R. D'Andrea, and P. Ganguly, "Near-optimal dynamic trajectory generation and control of an omnidirectional vehicle," *Robot. Auton. Syst.*, vol. 46, pp. 47–64, 2004.

[16] J. Ousingsawat and M. E. Campbell, "Establishing optimal trajectories for multi-vehicle reconnaissance," in *Proc. AIAA Guid., Navig., Control Conf.*, Providence, RI, 2004, [CD-ROM].

[17] L. Pallottino, E. Feron, and A. Bicchi, "Conflict resolution problems for air traffic management systems solved with mixed integer programming," *IEEE Trans. Intell. Transp. Syst.*, vol. 3, no. 1, pp. 3–11, Mar. 2002.

[18] J. Peng and S. Akella, "Coordinating the motions of multiple robots with kinodynamic constraints," in *Proc. IEEE Int. Conf. Robot. Autom.*, Taipei, Taiwan, Sep. 2003, pp. 4066–4073.

[19] A. Richards, T. Schouwenaars, J. P. How, and E. Feron, "Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming," *J. Guid., Control, Dynam.*, vol. 25, pp. 755–764, Jul.–Aug. 2002.

[20] A. Richards and J. P. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," in *Proc. Amer. Control Conf.*, Anchorage, AK, May 8–10, 2002, pp. 1936–1941.

[21] H. P. Rothwangl, "Numerical synthesis of the time optimal nonlinear state controller via mixed integer programming," in *Proc. Amer. Control Conf.*, Arlington, VA, Jun. 2001, pp. 3201–3205.

**Matthew Earl** was born in New Haven, CT, and grew up in Warwick, NY. He received the B.S. degree in mechanical engineering from the University of Rochester, Rochester, NY, in 1997, and the Ph.D. degree in dynamics and control from Cornell University, Ithaca, NY, in 2005.

While working toward the B.S. degree, he was a co-op student at the NASA Langley Research Center, Hampton, VA. Since 2005, he has been with BAE Systems, Advanced Information Technologies, Burlington, MA, where he is a Senior Analyst.

Dr. Earl received a Cornell University McMullen Fellowship and an NSF graduate research fellowship.

**Raffaello D'Andrea** was born in Pordenone, Italy. He received the BASc. degree in engineering science from the University of Toronto, Toronto, ON, Canada, in 1991, and the M.S. and Ph.D. degrees in electrical engineering from the California Institute of Technology, Pasadena, in 1992 and 1997, respectively.

Since 1997, he has been with the Department of Mechanical and Aerospace Engineering at Cornell University, Ithaca, NY, where he is an Associate Professor. He is also Vice President of Systems Architecture for Kiva Systems Incorporated. He was the system architect and faculty advisor of the RoboCup world-champion Cornell Autonomous Robot Soccer team in 1999 (Stockholm, Sweden), 2000 (Melbourne, Australia), 2002 (Fukuoka, Japan), and 2003 (Padova, Italy), and third-place winner in 2001 (Seattle, WA). His collaboration with Canadian artist Max Dean, "The Table," an interactive installation, appeared in the Biennale di Venezia in 2001. It was on display at the National Gallery of Canada in Ottawa, ON, in 2002 and 2003, and is now part of its permanent collection.

Dr. D'Andrea was the recipient of a University of Toronto W. S. Wilson Medal in Engineering Science, an American Control Council O. Hugo Schuck Best Paper award, an NSF CAREER Award, and a DOD-sponsored Presidential Early Career Award for Scientists and Engineers.