MIXED INTEGER PROGRAMMING FOR MULTI-VEHICLE PATH PLANNING

Tom Schouwenaars[*], Bart De Moor[*], Eric Feron[‡], Jonathan How[§ †]

[*] Esat-Sista, Katholieke Universiteit Leuven, Belgium
[‡] Laboratory for Information and Decision Systems,
[§] Space Systems Laboratory,
Massachusetts Institute of Technology,
Cambridge, MA 02139
fax: +(1) 617-253 5779
e-mail: feron@mit.edu

[†] ECC2001 Conference
e-mail: ecc2001@fe.up.pt
http://www.fe.up.pt/ecc2001/

Keywords: autonomous vehicles, path planning, collision avoidance

Abstract

This paper presents a new approach to fuel-optimal path planning of multiple vehicles using a combination of linear and integer programming. The basic problem formulation is to have the vehicles move from an initial dynamic state to a final state without colliding with each other, while at the same time avoiding other stationary and moving obstacles. It is shown that this problem can be rewritten as a linear program with mixed integer/linear constraints that account for the collision avoidance. A key benefit of this approach is that the path optimization can be readily solved using the CPLEX optimization software with an AMPL/Matlab interface. An example is worked out to show that the framework of mixed integer/linear programming is well suited for path planning and collision avoidance problems. Implementation issues are also considered. In particular, we compare receding horizon strategies with fixed arrival time approaches.

1 Introduction

The problem discussed in this paper is to use mathematical programming techniques to find the optimal path between two states for a single vehicle or a group of vehicles. This path will be optimized with respect to both fuel and/or time, and must ensure that the vehicles do not collide with each other or with obstacles, which can be fixed or moving. A classical approach to collision avoidance, especially in the field of robot motion planning, is the use of potential functions [1]. Other recently developed techniques include such approaches as randomized algorithms [2]. Path planning techniques using graph searching and surface covering [3] were developed earlier. This paper presents an alternative technique that uses integer programming as a modeling framework to formulate appropriate obstacle and inter-vehicle collision avoidance constraints on the motion of the dynamic system.

The paper first presents the basic problem formulation for a single vehicle, which is followed by the introduction of the constraints for obstacle avoidance. Note that we distinguish between (i) collision avoidance with stationary obstacles, (ii) collision avoidance with moving obstacles that have predefined paths, and (iii) mutual collision avoidance between multiple vehicles. A software implementation using AMPL and CPLEX with a Matlab interface to solve the mathematical program is presented. Section 6 compares different implementation strategies including receding horizon and fixed arrival time approaches.

2 Problem formulation

2.1 Single vehicle

The basic path planning problem discussed in this paper is the problem of finding a fuel-optimal path between the two states of one or more moving objects. Classical approaches to this optimization problem include the minimization of a quadratic cost function whose variables must satisfy the state space equations of the dynamical system $(\mathbf{A_c}, \mathbf{B_c})$ [4].

$$\min_{\mathbf{s},\mathbf{u}} J = \min_{\mathbf{s},\mathbf{u}} \int_0^\infty (\mathbf{s}'\mathbf{Q}\mathbf{s} + \mathbf{u}'\mathbf{R}\mathbf{u})\, dt \tag{1}$$

$$\text{subject to} \quad \dot{\mathbf{s}} = \mathbf{A_c}\mathbf{s} + \mathbf{B_c}\mathbf{u} \tag{2}$$

In this problem formulation, $\mathbf{s} \in \mathbb{R}^n$ is the state vector and $\mathbf{u} \in \mathbb{R}^{n_u}$ is the control. The system is assumed to start from some original state $\mathbf{s}_0$. The destination state is implicitly defined to be 0.

The cost function used in this paper uses a weighted one-norm. The (semi-)positive definite weighting matrices $\mathbf{Q}$ and $\mathbf{R}$ of the quadratic formulation are replaced by nonnegative weighting vectors $\mathbf{q}$ and $\mathbf{r}$, which gives the following convex cost function:

$$J = \int_0^\infty (\,\mathbf{q}'|\mathbf{s}| + \mathbf{r}'|\mathbf{u}|\,)dt \tag{3}$$

where $|\mathbf{v}|$ denotes the vector of absolute values of $\mathbf{v}$. When combined with the collision avoidance constraints (see Section 3) the 1-norm formulation yields a mixed integer/linear program, which is much easier to solve than the mixed integer/quadratic program that would be obtained using a quadratic cost.

In order to solve this fuel-optimal control problem numerically, one must discretize the system and use a finite time horizon T. The finite horizon T transforms into $N = T/\Delta t$ discrete time steps, where Δt is the sample time. To account for the remaining time $T \to \infty$ a terminal cost $f(\mathbf{s}_N)$ is introduced. The original optimization problem is thus transformed into the following one:

$$\min_{\mathbf{s}_i,\mathbf{u}_i} J_T = \min_{\mathbf{s}_i,\mathbf{u}_i} \sum_{i=0}^{N-1} \left(\mathbf{q}'|\mathbf{s}_i| + \mathbf{r}'|\mathbf{u}_i| \right) \Delta t + f(\mathbf{s}_N) \quad (4)$$

$$\text{subject to} \quad \mathbf{s}_{i+1} = \mathbf{A}\mathbf{s}_i + \mathbf{B}\mathbf{u}_i \quad (5)$$

where $\mathbf{A}$ and $\mathbf{B}$ are derived from $\mathbf{A_c}$ and $\mathbf{B_c}$ through appropriate discretizations. Neglecting the constant term $\mathbf{q}'|\mathbf{s}_0|$ and dividing the cost function by Δt gives:

$$J_T = \sum_{i=1}^{N-1} \mathbf{q}'|\mathbf{s}_i| + \sum_{i=0}^{N-1} \mathbf{r}'|\mathbf{u}_i| + f(\mathbf{s}_N) \quad (6)$$

A straightforward choice for the terminal cost $f(\mathbf{s}_N)$ is $\mathbf{p}'|\mathbf{s}_N|$, where the nonnegative weight vector $\mathbf{p}$ can be tuned appropriately. This yields a convex, piece-wise linear cost function that can be transformed into a linear form by introducing slack variables and additional constraints [5]. By introducing the slack variables w_{ij} $(i = 1,\ldots,N\ ,\ j = 1,\ldots,n)$ and v_{ik} $(i = 0,\ldots,N-1\ ,\ k = 1,\ldots,n_u)$, the optimization problem can be reformulated as:

$$\begin{aligned} \min_{\mathbf{w}_i,\mathbf{v}_i} J_T = \min_{\mathbf{w}_i,\mathbf{v}_i} & \sum_{i=1}^{N-1} \mathbf{q}'\mathbf{w}_i + \sum_{i=0}^{N-1} \mathbf{r}'\mathbf{v}_i + \mathbf{p}'\mathbf{w}_N \\ \text{subject to} \quad s_{ij} &\le w_{ij} \\ -s_{ij} &\le w_{ij} \\ u_{ik} &\le v_{ik} \\ -u_{ik} &\le v_{ik} \\ \mathbf{s}_{i+1} &= \mathbf{A}\mathbf{s}_i + \mathbf{B}\mathbf{u}_i \end{aligned} \quad (7)$$

where the indices j and k denote the components of the state and input vector respectively. The resulting optimization problem is a linear program, for which very efficient and highly optimized software packages exist. Several additional specifications can be added to this problem, including limits on control amplitude or constraints on the state location.

2.2 Multiple vehicles

The linear program (7) can easily be extended to a multi-vehicle case, where each vehicle has to move to a different final state. In order to specify an arbitrary final state, the cost function (6) is modified to minimize the difference between the state and the final state. Denote the state, input and final state vectors of the p^{th} vehicle as $\mathbf{s}_{pi}$, $\mathbf{u}_{pi}$ and $\mathbf{s}_{pf}$ respectively. In the most general case all vehicles can have different system matrices $(\mathbf{A}_p, \mathbf{B}_p)$. Moreover different weight vectors $\mathbf{q}_p$, $\mathbf{r}_p$ and $\mathbf{p}_p$ can be used for each vehicle. The cost function for the p^{th} vehicle is thus modified to:

$$J_{T,p} = \sum_{i=1}^{N-1} \mathbf{q}'_p |\mathbf{s}_{pi} - \mathbf{s}_{pf}| + \sum_{i=0}^{N-1} \mathbf{r}'_p |\mathbf{u}_{pi}| + \mathbf{p}'_p |\mathbf{s}_{pN} - \mathbf{s}_{pf}| \quad (8)$$

The total cost function for K vehicles can be defined as the sum of the separate cost functions. The linear program then becomes:

$$\begin{aligned} \min_{\mathbf{w}_{pi},\mathbf{v}_{pi}} \sum_{p=1}^{K} & \left(\sum_{i=1}^{N-1} \mathbf{q}'_p \mathbf{w}_{pi} + \sum_{i=0}^{N-1} \mathbf{r}'_p \mathbf{v}_{pi} + \mathbf{p}'_p \mathbf{w}_{pN} \right) \\ \text{subject to} \quad s_{pij} - s_{pf} &\le w_{pij} \\ -s_{pij} + s_{pf} &\le w_{pij} \\ u_{pik} &\le v_{pik} \\ -u_{pik} &\le v_{pik} \\ \mathbf{s}_{p,i+1} &= \mathbf{A}_p \mathbf{s}_{pi} + \mathbf{B}_p \mathbf{u}_{pi} \end{aligned} \quad (9)$$

In the next section, we introduce obstacle avoidance constraints and formulate them as mixed integer/linear constraints on the positions of the vehicles.

3 Binary Constraints for Collision Avoidance

3.1 Stationary obstacles

Consider for simplicity of exposition a single moving vehicle in a two-dimensional space that must avoid a rectangular obstacle. Let the position of the obstacle be denoted by the coordinates of its lower left and upper right corner points: $(x_{\min}, y_{\min})$ and $(x_{\max}, y_{\max})$. At every time step i the position (x_i, y_i) of the vehicle must lie in the area outside of the obstacle. This can be formulated as:

$$\begin{aligned} \forall i \in [1\ldots,N]: \quad x_i &\le x_{\min} \\ \text{or } x_i &\ge x_{\max} \\ \text{or } y_i &\le y_{\min} \\ \text{or } y_i &\ge y_{\max} \end{aligned} \quad (10)$$

Note that the vehicle is considered to be a moving point. This is a classical approach taken in robot motion planning: the obstacles are enlarged with the size of the moving object, such that the vehicle itself reduces to a moving point [1].

A way to transform the *or*-constraints (10) into more useful *and*-constraints is to introduce binary slack variables [6]. Let t_{ik} be a binary variable (0 or 1) and let M be a large arbitrary positive number. The constraints (10) may then be replaced by the following mixed-integer/linear constraints:

$$\begin{aligned} \forall i \in [1,\ldots,N]: \quad x_i &\le x_{\min} + Mt_{i1} \\ \text{and} \quad -x_i &\le -x_{\max} + Mt_{i2} \\ \text{and} \quad y_i &\le y_{\min} + Mt_{i3} \\ \text{and} \quad -y_i &\le -y_{\max} + Mt_{i4} \\ \text{and} \quad \sum_{k=1}^{4} t_{ik} &\le 3 \end{aligned} \quad (11)$$

If the k^{th} original *or*-constraint is not satisfied on the i^{th} time step, the corresponding binary variable t_{ik} equals 1. The last *and*-constraint ensures that at least one of the original *or*-constraints is satisfied, which ensures that the vehicle avoids the rectangle. The constraints (11) can be formulated for every obstacle situated in the manoeuvre space and for every vehicle in the multiple vehicle case. They are not enforced on the initial ($i = 0$) time step since these positions are fixed by the initial state.

This technique, however, is not restricted to rectangular planar obstacles. An arbitrarily shaped planar obstacle can be described by a surrounding polygon, of which the edges give rise to anti-collision constraints in both the x- and y-coordinates. The extension to 3D-movements with 3D-obstacles is straightforward: one only needs to formulate extra constraints in the z-coordinate and a 3D-obstacle can be described by a surrounding polyhedron.

3.2 Moving obstacles

An extension of the previous problem is to include moving obstacles with a predefined motion. A straightforward approach to this problem is to define new obstacles at every time step, corresponding to the positions of the moving obstacles at that instant. The coordinates of the obstacles change at every time step, according to their predefined motion. For translational motion of rectangular obstacles, the format of the binary constraints (11) remains the same. Rotational motion on the other hand and motion of non-rectangular obstacles yield constraints in both x- and y-coordinates. This strategy is readily implemented in the algorithms using AMPL and Matlab (Section 5).

3.3 Multiple vehicles

In the case in which multiple vehicles are moving to different destinations, collision avoidance between the vehicles can be dealt with in a way similar to the case of stationary obstacles. At each time step every pair of vehicles p and q must be a minimum distance apart from each other in the x- or y-coordinate. Considering planar motion again for simplicity of exposition, denote the positions of vehicle p and q at the i^{th} time step as (x_{pi}, y_{pi}) and (x_{qi}, y_{qi}) respectively. Let the safety distances be denoted by d_x and d_y. The constraints can then be described as follows:

$$\begin{array}{rrl} \forall i \in [1,\ldots,N] : \forall p,q \mid q>p : & |x_{pi}-x_{qi}| & \geq d_x \\ \text{or} & |y_{pi}-y_{qi}| & \geq d_y \end{array} \tag{12}$$

The condition $q > p$ avoids duplication of the constraints on the positions. Eliminating the absolute values transforms the constraints into:

$$\begin{array}{rrl} \forall i \in [1,\ldots,N] : \forall p,q \mid q>p : & x_{pi}-x_{qi} & \geq d_x \\ \text{or} & x_{qi}-x_{pi} & \geq d_x \\ \text{or} & y_{pi}-y_{qi} & \geq d_y \\ \text{or} & y_{qi}-y_{pi} & \geq d_y \end{array} \tag{13}$$

These constraints can again be formulated as mixed integer/linear constraints by introducing appropriate binary variables b_{pqik} :

$$\begin{array}{rrl} \forall i \in [1,\ldots,N] : \forall p,q \mid q>p : & & \\ & x_{pi}-x_{qi} & \geq d_x - Mb_{pqi1} \\ \text{and} & x_{qi}-x_{pi} & \geq d_x - Mb_{pqi2} \\ \text{and} & y_{pi}-y_{qi} & \geq d_y - Mb_{pqi3} \\ \text{and} & y_{qi}-y_{pi} & \geq d_y - Mb_{pqi4} \\ \text{and} & \sum_{k=1}^{4} b_{pqik} & \leq 3 \end{array} \tag{14}$$

Again the last constraint ensures that one of the original *or*-constraints is satisfied, which implies that the vehicles are a safe distance apart.

4 Mathematical program

The combination of the linear program (9) with the binary constraints for collision avoidance (11) and (14), yields a large non-convex mixed integer/linear program (MILP). To make the problem more realistic, one can also put linear or piece-wise linear –*i.e.* absolute value– constraints on maximum and minimum input and state [7].

Suppose that for all vehicles the state and input vectors are of the following form:

$$\mathbf{s}_p = [\, x_p \; y_p \; \dot{x}_p \; \dot{y}_p \,]^T \tag{15}$$

$$\mathbf{u}_p = [\, u_{xp} \; u_{yp} \,]^T \tag{16}$$

For simplicity only the position and the velocity are considered as state variables. Denote the minimum and maximum input and state vectors as $\mathbf{s}_{p,\min}$, $\mathbf{s}_{p,\max}$, $\mathbf{u}_{p,\min}$ and $\mathbf{u}_{p,\max}$. Constraints on the position should be interpreted as a rectangular area, between the boundaries of which the vehicles are supposed to stay.

Suppose there are K vehicles and L stationary obstacles. Let the complete time range be divided into N time steps. The complete mixed integer/linear program then becomes:

$$\min_{\mathbf{w}_{pi},\mathbf{v}_{pi}} \sum_{p=1}^{K} \left(\sum_{i=1}^{N-1} \mathbf{q}_p' \mathbf{w}_{pi} + \sum_{i=0}^{N-1} \mathbf{r}_p' \mathbf{v}_{pi} + \mathbf{p}_p' \mathbf{w}_{pN} \right) \tag{17}$$

subject to $\forall p \in [1,\ldots,K]$:

$$\begin{array}{rrl} \forall i \in [1,\ldots,N] : & x_{pi}-x_{pf} & \leq w_{pi1} \\ & -x_{pi}+x_{pf} & \leq w_{pi1} \\ & y_{pi}-y_{pf} & \leq w_{pi2} \\ & -y_{pi}+y_{pf} & \leq w_{pi2} \\ & \dot{x}_{pi}-\dot{x}_{pf} & \leq w_{pi3} \\ & -\dot{x}_{pi}+\dot{x}_{pf} & \leq w_{pi3} \\ & \dot{y}_{pi}-\dot{y}_{pf} & \leq w_{pi4} \\ & -\dot{y}_{pi}+\dot{y}_{pf} & \leq w_{pi4} \end{array} \tag{18}$$

$$\begin{array}{rrl} \forall i \in [0,\ldots,N-1] : & u_{xpi} & \leq v_{pi1} \\ & -u_{xpi} & \leq v_{pi1} \\ & u_{ypi} & \leq v_{pi2} \\ & -u_{ypi} & \leq v_{pi2} \end{array} \tag{19}$$

$$\forall i \in [0,\ldots,N-1]: \quad \begin{aligned} \mathbf{s}_{p(i+1)} &= \mathbf{A}_p \mathbf{s}_{pi} + \mathbf{B}_p \mathbf{u}_{pi} \\ \mathbf{s}_{p0} &= [\, x_{p0}\ y_{p0}\ \dot{x}_{p0}\ \dot{y}_{p0}\,]^T \end{aligned} \tag{20}$$

$$\forall\, i \in [0,\ldots,N-1]: \quad \begin{aligned} \mathbf{u}_{pi} &\geq \mathbf{u}_{p,\min} \\ \mathbf{u}_{pi} &\leq \mathbf{u}_{p,\max} \end{aligned} \tag{21}$$

$$\forall\, i \in [1,\ldots,N]: \quad \begin{aligned} \mathbf{s}_{pi} &\geq \mathbf{s}_{p,\min} \\ \mathbf{s}_{pi} &\leq \mathbf{s}_{p,\max} \end{aligned} \tag{22}$$

$$\forall\, c \in [1,\ldots,L], i \in [1,\ldots,N]: \quad \begin{aligned} x_{pi} &\leq x_{c,\min} + M t_{pci1} \\ -x_{pi} &\leq -x_{c,\max} + M t_{pci2} \\ y_{pi} &\leq y_{c,\min} + M t_{pci3} \\ -y_{pi} &\leq -y_{c,\max} + M t_{pci4} \\ \sum_{k=1}^{4} t_{pcik} &\leq 3 \\ t_{pcik} &= 0 \text{ or } 1 \end{aligned} \tag{23}$$

$$\forall i \in [1,\ldots,N]:\ q \in [p+1,\ldots,K]: \quad \begin{aligned} x_{pi} - x_{qi} &\geq d_x - M b_{pqi1} \\ x_{qi} - x_{pi} &\geq d_x - M b_{pqi2} \\ y_{pi} - y_{qi} &\geq d_y - M b_{pqi3} \\ y_{qi} - y_{pi} &\geq d_y - M b_{pqi4} \\ \sum_{k=1}^{4} b_{pqik} &\leq 3 \\ b_{pqik} &= 0 \text{ or } 1 \end{aligned} \tag{24}$$

5 Software

The total optimization problem from the previous section can be readily solved by a mixed integer program solver implemented in the CPLEX software package. CPLEX requires the standard MPS-format as input [8], which can be generated by running a specific m-script in Matlab. However, writing such an MPS-file generating script is an error prone task and such scripts are usually not very flexible in adapting the cost function or the type of constraints. We have therefore used the AMPL-language [9] to formulate the path planning problem.

Implementing the constraints in AMPL is straightforward, requiring minimum translation from the form shown in Section 4. The forms of the problem and the constraints are defined in a model file, while the parameter values are in a separate data file. Therefore, changes to the problem may be made without rebuilding the constraints expressions. The data files can be edited directly or generated by a simple Matlab script. AMPL combines the model and data to an MPS-file, which is then solved by CPLEX. Moreover, AMPL can handle piece-wise linear variables and thus absolute values: it hence generates the necessary slack variables and extra constraints automatically.

We thus specify the system matrices, the obstacle coordinates and other optional dynamical parameters in Matlab, and use a specific script to generate an AMPL input file. An AMPL script selects the appropriate model and data file, calls the solver and writes the resulting solution to an output file readable by Matlab. Other Matlab-scripts then plot the path and visualize the state and input sequence.

6 Planning Strategies

This section presents two planning strategies: a receding horizon strategy and a fixed arrival time approach. The latter can also be used to calculate the shortest time path.

6.1 Receding horizon

When applying a receding horizon strategy the path of the vehicles is composed of a sequence of locally optimal segments. At a certain time step the MILP from Section 4 is solved for the N future time steps, which provides the input commands for these N future time steps. However, only a subset of these N input commands are actually implemented. Instead, the process is repeated periodically and a new set of commands is developed for each time window. Usually the applied subset is restricted to the first control input. Hence a new set of input commands is calculated at each time step.

6.2 Fixed arrival time

An alternative approach is to use a *fixed arrival time*, which means that the arrival time is specified and the mathematical program is solved once over the entire time range. The final state (at time step N) can be constrained in this approach ($\mathbf{s}_N = \mathbf{s}_f$) instead of including it in the cost function. The cost function for a single vehicle then reduces to

$$J_T = \sum_{i=0}^{N-1} \sum_{j=1}^{n_u} r_j \, |u_{ij}| \tag{25}$$

where u_{ij} denotes the j^{th} component of the input vector at time step i. Thus the calculated path is designed to minimize the fuel used for that specific time range and weight vector $\mathbf{r}$.

The extension to the multiple vehicle case is straightforward. In case the arrival times of all K vehicles are the same, the cost function becomes

$$J = \sum_{p=1}^{K} \sum_{i=0}^{N-1} \sum_{j=1}^{n_u} r_{pj} \, |u_{pij}| \tag{26}$$

where u_{pij} denotes the j^{th} component of the input vector of the p^{th} vehicle at time step i. Note that it is also possible to specify different arrival times for each vehicle.

The fixed arrival time approach can also be used to calculate the minimum-time path between two points. This can be done by solving a feasibility problem corresponding to the constraints of the MILP, while decreasing the horizon length. The shortest time path can then be computed for the minimum horizon length for which the mathematical program is still feasible.

6.3 Optimality and computation time

The major difference between the fixed arrival time and the receding horizon approach is the difference in the optimization criteria. By specifying the arrival time and accounting for the final state in the constraints, the cost function (6) is reduced to only contain the input variables. Therefore there is no trade-off between input and state minimization and the computed input sequence is thus the fuel-optimal one for that specific horizon length and those specific weight factors, taking into account the dynamic and collision avoidance constraints. When using a receding horizon (RH), however, the solution is locally optimal on every segment. Thus the total RH-path probably requires more fuel than the vehicle would consume if the path was calculated over the entire time range.

Consider an autonomous ground vehicle that has to manoeuvre between a set of rectangular obstacles. The vehicle is modeled as a simple 2-D discrete double integrator with discretization step $\Delta t = 0.1s$. Figure 1 shows the trajectory for a fixed arrival time $T_{arr} = 8s$ or 80 time steps. The initial position is $(15, 8)$; the destination is $(-0.5, -5)$. The vehicle mostly moves around the obstacles, which consumes the least amount of fuel. Although the vehicle could arrive earlier, it uses the complete time range to arrive at its destination. The velocity and acceleration thus remain as low as possible, which corresponds to the minimum fuel use.

In order to compare the different strategies the total sum E_{tot} of the absolute values of the input components is considered as a measure for fuel consumption:

$$E_{tot} = \sum_{i=0}^{N_{arr}} (|u_{ix}| + |u_{iy}|) \tag{27}$$

N_{arr} is the last but one time step, corresponding to the last input. In the fixed arrival time example $N_{arr} = 79$ and the vehicle consumes $E_{tot} = 21.2$ of fuel.

In Figure 2 the fixed arrival time approach is used to compute the shortest time path between the two points as described in

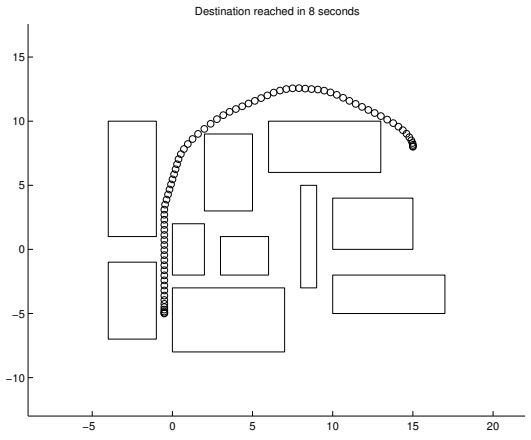

Figure 1: Fuel-optimal path of unmanned vehicle for a fixed arrival time $T_{arr} = 8s$

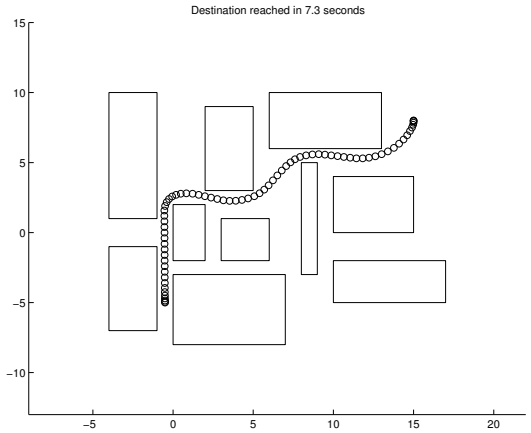

Figure 2: Fuel-optimal shortest time path of unmanned vehicle: $T_{min} = 7.3s$

Section 6.2. This shortest time is $T_{min} = 7.3s$. To reach the destination within $7.3s$, the vehicle now has to manoeuvre between the obstacles, requiring $E_{tot} = 34.3$ of fuel.

Figure 3 shows the receding horizon path for a horizon $T_{hor} = 3s$ or 30 time steps. The fuel consumption is now $E_{tot} = 35.7$ and the vehicle reaches the destination after $T_{arr} = 9.2s$ or 92 iterations. The fuel consumption is thus 59.3% greater than in the fixed arrival time case with $T_{arr} = 8s$. As now the "distance" to the final location is included in the cost function and is thus minimized, the vehicle starts manoeuvring between the obstacles, which explains the increased fuel consumption.

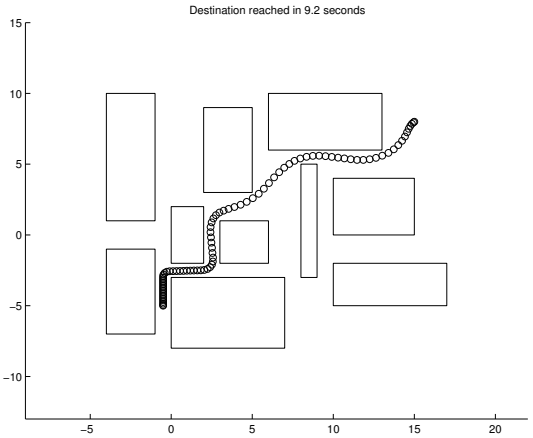

Figure 3: Trajectory of unmanned vehicle under receding horizon control with $T_{hor} = 3s$

Of course, the key benefit of the receding horizon approach is that the solution can be calculated much faster than the fixed-time problem. This follows from the fact that the number of variables and constraints increases polynomially with the horizon length, but the worst-case solution time increases exponentially [5]. The trade-off between optimality and computation time is important for a real-time implementation of these path

planning algorithms. This is shown in Table 1, where the arrival time T_{arr}, the total fuel consumption E_{tot}, the total computation time T_{comp} and the average computation time per iteration T_{it} are shown for different horizon lengths T_{hor}. Although the vehicle arrives earlier, in this scenario, the fuel consumption decreases with the horizon length. For comparison, Table 2 gives the fuel consumption E_{tot} and computation time T_{comp} for a fixed arrival time planning for the arrival times T_{arr} in Table 1. With a fixed time approach the vehicle clearly consumes less fuel, but the computation time increases strongly. Future research will explore these observations in more detail.

$T_{hor}(s)$	$T_{arr}(s)$	E_{tot}	$T_{comp}(s)$	$T_{it}(s)$
3.0	9.2	35.7	129	1.40
3.5	9.0	34.1	267	2.97
4.0	8.4	31.6	441	5.25
4.5	8.2	30.4	728	8.88
5.0	8.2	30.4	1213	14.79

Table 1: Arrival times, fuel consumption and computation times for different horizon lengths.

$T_{arr}(s)$	E_{tot}	$T_{comp}(s)$
9.2	17.4	613
9.0	17.8	442
8.4	19.2	193
8.2	20.0	189
8	21.2	162

Table 2: Arrival times, fuel consumption and computation times for fixed arrival time planning

6.4 Safety

Although the solution to the MILP yields a collision free path, in a receding horizon setting this still does not guarantee that no collisions will occur. Indeed, in a RH-setting a single vehicle or a set of vehicles can be led to a critical situation for which the MILP becomes unfeasible in the next iteration. This can happen if a vehicle has moved too close to an obstacle or if multiple vehicles have moved too close to each other. If the remaining distance is too short to stop or to avoid the obstacle or other vehicles, a collision will occur. Such critical situations typically occur when the horizon is too short to "spot" obstacles or other vehicles in time.

One thus has to guarantee that a vehicle can always stop relative to the surrounding environment or has enough time to carry out a collision avoidance manoeuvre. More generally, one needs policies that guarantee *a priori* that no inter-vehicle collisions nor collision with obstacles will occur. Thus strategies are needed in which a vehicle can always return to a safe state whenever it faces a critical situation. However, general "safe" guidance policies are not readily available, which is a topic of further research.

7 Conclusion

This paper presented a new approach to trajectory planning of multiple vehicles that directly incorporates collision avoidance as mixed integer/linear constraints. A mathematical programming formulation has been proposed including binary constraints for stationary and moving obstacle avoidance and multi-vehicle path planning. Several strategies with corresponding trade-offs were discussed and examples were given to show the potential of our approach. It was shown that receding horizon strategies, while computationally more attractive than strategies aimed at computing complete trajectories *a priori*, can lead the system to unsafe conditions. Future work will therefore concentrate on procedures ensuring that a single vehicle or a set of vehicles can always return to a "safe" state from wherever it stands.

Acknowledgments

The authors would like to thank Emilio Frazzoli, Sommer Gentry, Mao Zhi-Hong, Arthur Richards (MIT) and Jeroen Buijs (KU Leuven) for their support during this work.

References

[1] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[2] J. Barraquand, L.E. Kavraki, J.C. Latombe, T.Y. Li, R.Motwani, and P.Raghavan. A random sampling scheme for path planning. *International Journal of Robotics Research*, 16(6):759–774, 1997.

[3] T. Lozano-Perez. *Spatial planning with polyhedral models*. June 1980. MIT Ph.D. Thesis (E.E.).

[4] Michael Athans and Peter L. Falb. *Optimal Control, An Introduction to the Theory and Its Applications*. McGraw-Hill, 1966.

[5] Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.

[6] Hamdy A. Taha. *Operations Research, An Introduction*. Macmillan Publishing Company, New York, 4th edition, 1987.

[7] Andrew Robertson, Gokhan Inalhan, and Jonathan P. How. Spacecraft formation flying control design for the orion mission. *Proceedings of AIAA/GNC*, August 1999.

[8] *ILOG CPLEX User's guide*. ILOG, 1999.

[9] R. Fourer, D. M. Gay, and B. W. Kernighar. *AMPL, A modeling language for mathematical programming*. The Scientific Press, 1993.