Receding Horizon Implementation of MILP for Vehicle Guidance

Yoshiaki Kuwata and Jonathan How

I. MILP Trajectory Optimization

A. Problem Statement

The problem statement is to optimize the vehicle trajectory by minimizing time of arrival at the goal, given the initial position \( x_0 \), the goal location \( x_{\text{goal}} \), and the no-fly zones \( X_{\text{obst}} \). The resultant trajectory must be consistent with the vehicle dynamics such as turn rate and speed constraints and avoid no-fly zones and obstacles. The vehicle is modeled as a simple point mass with position and velocity as state variables \( x \) and acceleration as control inputs \( u \) [1].

\[
\min_{\Delta t} \phi_1(b_{\text{arrival}}) = \min \sum_{i=1}^{n_p} b_{\text{arrival},i} \Delta t
\]

Experience has shown that the computational effort required to solve this optimization can grow rapidly with the length of the trajectory to be planned [1], [2], but a receding horizon approach can be used to design large-scale trajectories.

B. Fixed Horizon Minimum Time Controller

One approach to design a minimum arrival time controller is to make a plan over a fixed planning horizon [1]. At time step \( k \), a series of control inputs \( \{u(k+i), i = 0, 1, \ldots, n_p-1\} \) are chosen that give the trajectory \( \{x(k+i), i = 1, 2, \ldots, n_p\} \). Constraints are added to specify that one of the \( n_p \) trajectory points must visit the goal. The MILP optimization minimizes the time along this trajectory at which the goal is reached, using \( n_p \) binary decision variables \( b_{\text{arrival}} \in \{0, 1\} \) as

\[
\min_{u(\cdot), b_{\text{arrival}}} \phi_1(b_{\text{arrival}}) = \min \sum_{i=1}^{n_p} b_{\text{arrival},i} \Delta t
\]

Experience has shown that the computational effort required to solve this optimization can grow rapidly with the length of the trajectory to be planned [1], [2], but a receding horizon approach can be used to design large-scale trajectories.

II. Receding Horizon Formulation

The receding horizon control strategy is comprised of a cost estimation phase and a trajectory design phase [3]. The cost estimation phase computes a compact “cost map” of the approximate minimum time-to-go from a limited set of points to the goal. The cost estimation phase is performed once for a given obstacle field and position of the goal, and would be repeated if the environment changes. The trajectory designer uses this cost map information in the terminal penalties of the receding horizon optimization to design a series of short trajectory segments that are followed until the goal is reached. An example of a result that would be expected from the trajectory design phase is shown schematically in Fig. 1. A trajectory consistent with discretized aircraft dynamics is designed from \( x(k) \) over a fine resolution planning horizon with length \( n_p \) steps. The trajectory is optimized using MILP to minimize the cost function assessed at the plan’s terminal point \( x(k+n_p) \). The optimization chooses a visible point \( x_{\text{vis}} \) from a list of candidate cost points \( x_{\text{cost}} \) whose cost-to-go was previously estimated, plus the cost-to-go estimate \( C_{\text{vis}} \) for \( x_{\text{vis}} \). This \( C_{\text{vis}} \) is estimated using a coarser model of the aircraft dynamics that can be evaluated very quickly.

\[
\min_{u(\cdot), b_{\text{vis}}} \phi_2 = \min_{u(\cdot), b_{\text{vis}}} \frac{L_2(x_{\text{vis}} - x(k+n_p))}{v_{\text{max}}} + C_{\text{vis}}
\]

Only the first \( n_c \) steps are executed before a new plan is formed starting from \( x(k+n_c) \). Ref. [4] discusses the stability of this trajectory design approach.

III. Code Implementation Example

To show the ease of implementation, consider the AMPL and CPLEX code for the cost point selection. The values of the position \( x_{\text{vis}} \) and cost \( C_{\text{vis}} \) are evaluated using binary variables \( b_{\text{vis}} \in \{0, 1\} \) and the \( n \) points on the cost map as

\[
x_{\text{vis}} = \sum_{j=1}^{n} b_{\text{vis},j} x_{\text{cost},j}
\]

\[
C_{\text{vis}} = \sum_{j=1}^{n} b_{\text{vis},j} C_j
\]

\[
\sum_{j=1}^{n} b_{\text{vis},j} = 1
\]

The associated AMPL source code is:

```AMPL
param costPosns {PTS, DIMS};
param costVals {PTS};
var bVisPt {PTS} binary;
var rVisPt {DIMS};
var visPtCost;
subject to visiblePtEqn {d in DIMS}:
  rVisPt[d] = sum(c in PTS) bVisPt[c]*costPosn[c,d];
subject to visiblePointCostEqn:
  visPtCost = sum(c in PTS) bVisPt[c]*costVal[c];
subject to chooseOneVisPt:
  sum (c in PTS) bVisPt[c] = 1;
```

Y. Kuwata, Dept. of Aeronautics and Astronautics, MIT, Cambridge, MA 02139, USA kuwata@mit.edu
J. How, Associate Professor, Dept. of Aeronautics and Astronautics, MIT, Cambridge, MA 02139, USA jhow@mit.edu
that RH-MILP always has a feasible solution and that the vehicle always reaches the goal in finite time [6]. Modified Dijkstra’s algorithm is developed that rejects infeasible node sequences to form a modified cost map. In order to maintain feasibility of the trajectory optimization, the planning horizon must be extended beyond the execution horizon, and a lower bound of the planning margin required was derived analytically.

C. Three Dimensional Environment

In a three dimensional environment the paths are chosen to stay as close to the terrain as possible to avoid exposure to threats, but the vehicle can choose to pop-up over the obstacles if necessary. Candidate cost points are placed on the edges of the obstacles as opposed to just the obstacle corners. The extended RH-MILP algorithm is shown to be computationally tractable in complicated environment [7].

D. Initial Guess

In order to shorten the solution time of the MILP, an initial feasible solution can be provided with the solver [7], [8]. RH-MILP executes only the first step of the $n_p$ step plan, and re-optimizes from the state that will be reached. The decisions (e.g. visible point selection, obstacle avoidance) made in the previous solution could be used when constructing an initial guess.

V. CONCLUSIONS

This paper presents MILP based techniques for trajectory optimization of unmanned aerial vehicles. To reduce the computation load associated with MILP optimization, a receding horizon controller has been developed. The combination of coarse cost map and detailed short trajectory significantly reduces the size of the MILP optimization. Several extensions are also presented that further reduce computation time and expand the capabilities of RH-MILP.

ACKNOWLEDGMENTS

Research funded by AFOSR Grant # F49620-01-1-0453.

REFERENCES