Mixed-integer Programming for Control

Arthur Richards
Department of Aerospace Engineering
University of Bristol
Bristol, BS8 1TR, UK
arthur.richards@bristol.ac.uk

Jonathan How
Aerospace Control Laboratory
Massachusetts Institute of Technology
Cambridge MA 02139
jhow@mit.edu

Abstract— **This tutorial session describes how Mixed-integer Programming (MIP) can be employed for feedback control. MIP can be used to find optimal trajectories subject to integer constraints, which can encode discrete decisions or nonconvexity, for example. This optimization can be performed online within Model Predictive Control (MPC) to implement a feedback control law. The tutorial discusses how to model systems using MIP, the implementation as a MPC, and techniques for fast solutions of the optimizations to make them suitable for real-time use.**

I. Introduction

Mixed-integer programming (MIP) is a very general framework for capturing problems with both discrete decisions and continuous variables. This includes:

- Assignment problems [2]
- Control of hybrid systems [12]
- Piecewise-affine (PWA) systems (including approximations of nonlinear systems) [11]
- Problems with non-convex constraints (e.g., collision avoidance) [7]

MIP methods naturally handle these types of problems because the integer decision variables can be used to encode discrete decisions, e.g. assignment decisions or "left" vs "right" for collision avoidance [4], [5]. Having cast the problem in MIP form, techniques from Model Predictive Control (MPC) can be used to build a feedback control law using the optimization online.

This tutorial will focus on the two major challenges associated with using MIP/MPC for online control. The first is to encode the problem as a MIP optimization that can be embedded within MPC (see Figure 1). The second, often harder, challenge is to develop modifications to the problem statement to execute these optimizations in real-time. This challenge arises because integer programming is in the class of $\mathcal{NP}$-complete problems and can require extensive computation for problems with many integer variables. Further pressure on computation time arises from the desire to apply MIP/MPC to problems with fast dynamics. MPC has been widely and successfully employed in the process industry for problems with relatively long time scales, typically on the order of hours. However, some forthcoming applications for MIP/MPC, such as target assignment and path-planning for Unmanned Aerial Vehicles [26], have time scales on the order of minutes or seconds. Thus a successful deployment often requires judicious choice of approximation and solution techniques. While the final installation details are usually problem-specific, there are general approaches and guidelines for suitable approximations. The first part of the tutorial will focus on the problem formulation and general implementation techniques. The following talks will provide more specific implementation details and solution techniques that have recently been developed:

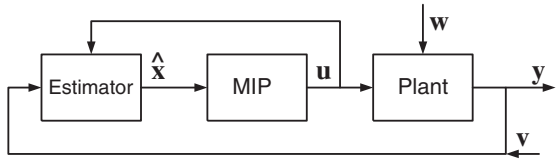

Fig. 1. MPC with Output Feedback

1. Topics of the first talk include modeling systems using MIP (assignment problems, non-convex constraints and PWA systems); using optimization for real-time control (stability requirements, robustness); and general approaches for online solutions (approximations, constraint relaxation, cost-to-go functions) by J. P. How (MIT) and A. Richards (Bristol).
2. Projected Variable Metric Algorithm for Mixed Integer Optimization Problem by Ali Ahmadzadeh (Univ. of Pennsylvania), Bijan Sayyarrodsari (Pavilion Tech. Inc), and Abdollah Homaifar (North Carolina A&T State Univ.)
3. MILP Assignment Problems for Multi-Vehicle Systems by Matthew Earl and Raffaello D'Andrea (Cornell Univ.)
4. Real-Time MILP Path-Planning for Tactical UAV Applications by Cedric Ma (Northrop Grumman Corp.) and Robert H. Miller, (Univ. of Michigan)
5. Receding Horizon Implementation of MILP for Vehicle Guidance by Yoshiaki Kuwata and Jonathan P. How (Massachusetts Inst. of Tech.)

II. Definition of MILP

A *Mixed-Integer Linear Program* (MILP) is a special case of a Linear Program (LP) in which some of the decision variables are constrained to take only integer values. Given matrices $\mathbf{A}_1, \mathbf{A}_2$ and vectors $\mathbf{f}_1, \mathbf{f}_2, \mathbf{b}$, the general MILP is

given by

$$\begin{aligned} \min_{\mathbf{x},\mathbf{z}} \quad & \mathbf{f}_1^T\mathbf{x} + \mathbf{f}_2^T\mathbf{z} \\ \text{subject to} \quad & \mathbf{A}_1\mathbf{x} + \mathbf{A}_2\mathbf{z} \leq \mathbf{b} \\ & \mathbf{z} \text{ integer} \end{aligned}$$

This problem is inherently non-convex: if for some problem $z_1 = 0$ and $z_1 = 1$ are both solutions, points in between $z_1 \in (0, 1)$ violate the integer requirement on z_1 and are therefore infeasible. The problem is also in the class of $\mathcal{NP}$-complete problems [3], [4]. This means that there is no algorithm that can guarantee solving any MILP problem in a time that is a polynomial function of the problem size, *i.e.,* the number of decision variables and constraints[1]. However, with good software and modeling, many useful MILP problems can be solved quickly enough to be of practical use, even though the worst-case guaranteed solution time is far longer.

III. Modeling using MILP

This section briefly illustrates by examples the capabilities of MILP for modeling complex problems. The common feature throughout is the encoding of discrete decisions using the integer variables.

A. Resource Allocation

Many resource allocation problems are inherently discrete. For example, the manufacture of items commonly concerns only integer quantities of items. Also, the assignment of tasks to agents is discrete.

Consider a simple task assignment problem in which N tasks are to be assigned to N agents. All the tasks must be assigned, and no agent can perform more than one task, hence every agent must be assigned a task. The tasks are discrete: they cannot be shared between agents. The cost of assigning agent i to task j is c_{ij}. To solve this problem by optimization, define the decision variable $z_{ij} = 1$ if task i is assigned to agent j and 0 otherwise. Therefore the problem can be written as

$$\begin{aligned} \min_{\mathbf{z}} \quad & \sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij} z_{ij} \\ \text{subject to} \quad & \sum_{i=1}^{N} z_{ij} = 1 \forall j \in \{1 \dots N\} \\ & \sum_{j=1}^{N} z_{ij} = 1 \forall i \in \{1 \dots N\} \\ & z_{ij} \in \{0, 1\} \forall i, j \end{aligned}$$

It turns out that the problem above is a special case and can be solved using only LP [6]. If the integer constraints $z_{ij} \in \{0, 1\}$ are replaced with simple bounds $0 \leq z_{ij} \leq 1$, then the vertices of the constraint set are all at integer values of z_{ij}, and since the LP solutions are at the vertices, these solutions will solve the integer problem too. However, suppose there is some resource required to perform each task, and define r_{ij} as the amount of resource consumed by agent j in performing task i. If the total resource available is R, the following constraint must be added

$$\sum_{i=1}^{N}\sum_{j=1}^{N} r_{ij} z_{ij} \leq R$$

The LP solution can no longer be used, and the problem must be posed and solved as a MILP.

Many other common resource allocation problems can be expressed in MILP form, including job shop scheduling [15], the traveling salesman problem [13], and the vehicle routing problem [14].

B. Piecewise Affine Functions

Any continuous nonlinear function can be arbitrarily well approximated using a Piecewise Affine (PWA) function [4]. In turn, a piecewise affine function can be encoded using MILP [12], [11]. The advantage of the MILP approach over a nonlinear optimization method lies in the handling of non-convexity. Some MILP solution methods guarantee globally-optimal solutions (see Section IV), whereas most nonlinear methods have no such guarantee, and may become "trapped" by local minima.

Consider a continuous PWA function $f(y)$ defined on the interval $Y_1 \leq y \leq Y_N$ with specified values at intermediate points $f(Y_i) = F_i$ and a linear interpolation between those points [3]. The minimization of $f(y)$ can be written in MILP form as

$$\begin{aligned} \min_{\mathbf{x}} \quad & \sum_{i=1}^{N} x_i F_i \\ \text{subject to} \quad & \sum_{i=1}^{N} x_i = 1 \\ & x_i \geq 0 \ \forall i \in \{1 \dots N\} \\ & x_1 \leq z_1 \\ & x_i \leq z_{i-1} + z_i \ \forall i \in \{2 \dots N-1\} \\ & x_N \leq z_{N-1} \\ & \sum_{i=1}^{N-1} z_i = 1 \,,\ z_i \in \{0, 1\} \end{aligned}$$

where the binary variable $z_i = 1$ if $Y_i \leq y \leq Y_{i+1}$, effectively "choosing" which interval the solution lies in, and then constraining the cost function to be a linear combination, using multipliers x_i and x_{i+1}, of the function values at the ends of that interval, *i.e.,* $f(y) = x_i F_i + x_{i+1} F_{i+1}$ corresponding to $y = x_i Y_i + x_{i+1} Y_{i+1}$.

C. Disjunctions and Non-convex Sets

Consider a problem in which one of two constraints must be satisfied, for example, *either* $\mathbf{a}_1^T\mathbf{x} \leq b_1$ *or* $\mathbf{a}_2^T\mathbf{x} \leq b_2$. While each of these is a linear constraint, there is no way of

[1]This statement is unproven, although backed by much evidence and experience. The formal study of complexity is a large subject in its own right and beyond the scope of this tutorial.

writing "or" in a standard LP [5]. However, the constraints can be expressed in MILP form as

$$\begin{aligned} \mathbf{a}_1^T\mathbf{x} &\leq b_1 + Mz_1 && \text{(1a)} \\ \mathbf{a}_2^T\mathbf{x} &\leq b_2 + Mz_2 && \text{(1b)} \\ z_1 + z_2 &\leq 1 && \text{(1c)} \end{aligned}$$

where z_1 and z_2 are additional binary variables and M is a very large positive number. If $z_1 = 1$, then the right hand side of (1a) is a very large positive number, and if M was chosen sufficiently large, this constraint is effectively relaxed. Then, since $z_1 = 1$, the logical constraint (1c) requires $z_2 = 0$, and the second constraint (1b) must be enforced in its original form, *i.e.* no M term. Similarly, if $z_2 = 1$, the constraint (1b) is relaxed, but (1c) requires $z_1 = 0$, hence (1a) is applied without the M term. This can be viewed as another discrete decision: the binary variables encode a choice of which constraint to apply, and the logical constraint ensures that at least one of them is applied.

The formulation in (1) is an example of what is commonly referred to as a "big-M" form (some authors use "big-D", "big-R" or others but the principle is the same). The binary variables act as "switches" on the continuous constraints, appearing with a large weighting, and are also subject to logical constraints. This is a very powerful method that can capture many useful logical constructs [9], [10]. However, the large weighting M can cause conditioning problems and makes the MILP solution process harder. Many useful problems can be solved using big M methods, but it is always worth spending some time looking for a big-M-free representation: if it exists, it could be more efficient.

The disjunction method can be extended to consider an arbitrary non-convex set constraint by approximating the set as a union of convex polytopes

$$\begin{aligned} \mathcal{S} &:= \bigcup_{i=1}^{N} \mathcal{P}_i \\ \mathcal{P}_i &:= \{\mathbf{x} \,:\, \mathbf{A}_i\mathbf{x} \leq \mathbf{b}_i\} \end{aligned}$$

Then the constraint $\mathbf{x} \in \mathcal{S}$ can be thought of as a disjunction: *either* $\mathbf{x} \in \mathcal{P}_1$, $\mathbf{x} \in \mathcal{P}_2$ ... *or* $\mathbf{x} \in \mathcal{P}_N$. Using the same approach as in (1), this can be written in MILP form

$$\begin{aligned} \mathbf{A}_i\mathbf{x} &\leq \mathbf{b}_i + \mathbf{1}Mz_i \quad \forall i \in \{1 \dots N\} \\ \sum_{i=1}^{N} z_i &\leq N - 1 \end{aligned}$$

D. Avoidance

A particularly useful variation of the disjunction constraint can be applied to collision avoidance for vehicle path-planning [7]. Suppose a point on a trajectory is required to be *outside* a polytope $\mathbf{x} \notin \mathcal{P}$ where $\mathcal{P} := \{\mathbf{x} \,:\, \mathbf{A}\mathbf{x} < \mathbf{b}\}$. This exclusion constraint can be written as a disjunction

$$\begin{aligned} \text{either } \mathbf{a}_1^T\mathbf{x} &\geq b_1 \\ \text{or } \mathbf{a}_2^T\mathbf{x} &\geq b_2 \\ &\vdots \\ \text{or } \mathbf{a}_N^T\mathbf{x} &\geq b_N \end{aligned}$$

where $\mathbf{a}_i$ is the i^{th} row of $\mathbf{A}$. Then using the big-M approach gives the following MILP constraints

$$\begin{aligned} \mathbf{a}_i^T\mathbf{x} &\geq b_1 - Mz_i \quad \forall i \in \{1 \dots N\} \\ \sum_{i=1}^{N} z_i &\leq N - 1 \end{aligned}$$

This method has been applied to obstacle avoidance problems in two and three dimensions for spacecraft and aircraft, and can be extended to prevent inter-vehicle collisions and impingement from thruster plumes [8].

IV. Solving MILP Problems

It is not necessary to have a deep understanding of MILP solution algorithms to successfully apply MILP for problem solving. A variety of solvers are readily available [1]. For large problems, commercial packages such as CPLEX [16] include efficient heuristics and solve many problems rapidly. Noncommercial software for solving MILP's are discussed in detail in Ref. [19]. Mathematical programming languages like AMPL [17] can simplify the implementation of complicated formulations [2].

The remainder of this section is devoted to a brief discussion of algorithms for solving general MILP problems. Note that there are other methods for discrete optimization that are applicable to more specific problem classes. For example, the knapsack problem [3] is a type of resource allocation problem that can be posed and solved as a MILP, but it can also be efficiently solved using dynamic programming (DP) [18]. While DP can be applied to many dynamics problems that can also be expressed in MILP form, it is not a solution algorithm for general MILP problems, so it is not discussed here. General MILP algorithms fall into two classes: heuristic and branching. Heuristic methods are only briefly discussed here. Branching methods are more attractive for many applications because of the guaranteed global optimality of solutions.

A. Branch and Bound

The branch-and-bound algorithm is the most popular choice for solving MILP problems [4], [3], [27]. CPLEX is based on this algorithm. The great advantage of the branch-and-bound method is that, when it terminates, the solution is known to be globally optimal. This is the great benefit of the MILP approach: it can achieve globally optimal solutions for non-convex problems.

The branch-and-bound algorithm begins by solving a relaxed form of the problem, replacing integrality constraints with simple bounds, and then "branching" on a chosen

[2]Example codes available at `http://hohmann.mit.edu/milp/`

variable: that variable is fixed at various integer settings, each generating a new relaxed sub-problem and a better bound on the optimal solution. This procedure continues, searching a "tree" with different integer settings for each branch. If the result of a relaxed subproblem satisfies the integrality constraints, that branch does not need to be searched any further, and its solution is a feasible solution to the original MILP. If the solution to an relaxed subproblem is not integral, but has a cost worse than the best MILP solution found already, that branch can be terminated, or "fathomed," as further branching will only increase the cost. The search terminates when all branches have been searched.

In general, the branch-and-bound method cannot be *guaranteed* to terminate without searching the entire tree, giving a computation time limit that is exponential in the number of variables. This is consistent with the $\mathcal{NP}$-completeness of the problem. However, when implemented with good heuristics to decide the branching and searching strategy, globally-optimal solutions can be found to many large, difficult problems in times small enough to be useful.

B. Heuristic Methods

Heuristic methods are typically based on some random search procedure. This is attractive for non-convex problems, as the randomness reduces the susceptibility of the methods to getting "stuck" in local minima. Several methods dominate: simulated annealing, genetic algorithms, and Tabu search.

Simulated annealing mimics the cooling of atoms in a metal. At an initial high "temperature," the solution can make large "jumps" at random across the search space. As the metal "cools," the probability of jumping reduces, and the method becomes more like a random local search.

Genetic algorithms mimic evolution [21] in that a "population" of solutions is chosen and evolved through "generations," each involving combining the features of pairs of individual solutions and introducing random variations. In both simulated annealing and genetic algorithms, because movements in the search are randomized, they can easily be chosen to use only integer solutions.

Tabu search has also proven to be an effective heuristic for solving combinatorial optimization problems such as scheduling, telecommunications, transportation and network problems [20]. The Tabu method searches a neighborhood of a given solution for a better feasible solution. The *neighborhood of a solution* is defined to be all solutions that can be reached in a *single move*, where the definition of a move is problem specific, but typically includes changing one "bit" in the solution from 1 to 0, or swapping the position of two elements in the solution vector. The problem with most neighborhood methods of this type is that they can get trapped in a local minimum. To prevent this looping between the same solutions, Tabu search uses the concept of *memory* and a Tabu list.

V. Using MILP for Control

This section describes how MILP can be employed online for feedback control.

A. Model Predictive Control

Model Predictive Control (MPC) [22], [23], also known as receding-horizon control, can be summarized in the following algorithm.

1. Using numerical optimization, design a control sequence for a finite time ahead (the "horizon") beginning from the current state
2. Implement an initial portion of that optimized[3] control sequence
3. Go to 1

This is a feedback control system: because the optimization in Step 1 uses the current state as its initial condition, the optimized control sequence is a function of the state and therefore the control is a function of the state, implying feedback action. It also involves feedforward, as the prediction within the optimization anticipates future behavior.

The major benefit of MPC is its natural ability to handle constraints. The constraints can be applied within the optimization in Step 1, and numerical optimization subject to constraints is readily done. MPC with constrained optimization results in a controller that is explicitly designed with both performance and constraints in mind. This is favorable compared to many other approaches combining an optimized controller, ignoring constraints, with additional de-tuning or supervisory control to account for the constraints.

B. Properties of Model Predictive Control

When designing a feedback controller, it is always necessary to check that the resulting closed-loop system is stable. Other properties of interest include robustness and convergence [24]. For MPC with constraints, feasibility is also a key concern: since the control law is based on the solution to an optimization, the system should have some guarantee that a solution can be found. This section discusses how to establish these useful properties for MPC.

In the case of MPC, all of these properties are established using a common approach, based on a recursion. It begins by assuming that a feasible solution is known at some time t_0. Then that solution is used to construct a candidate solution for the subsequent planning problem at time $t_0+\Delta$. That candidate solution is then proven to be feasible, *i.e.* to satisfy the constraints of the planning problem at time $t_0+\Delta$, for all initial conditions and times. In the case of robustness properties, the recursion must also be proven for all realizations of uncertainty between times t_0 and $t_0+\Delta$, *e.g.* all realizations of the plant or all disturbances. The recursion can also be expressed in set form: if the state is

[3]This may be the optimal solution to the optimization, but that does not mean that it is the optimal feedback control law, hence we use "optimized" here instead of "optimal".

in some state at time t_0, it is in that set at time $t_0 + \Delta$. These two forms are equivalent.

Once proven, the recursion implies feasibility of all the optimizations to be solved. If feasibility at time t_0 implies that a particular solution exists at $t_0 + \Delta$, then the planning problem at $t_0 + \Delta$ must be feasible, hence feasibility is maintained at all times. By some definitions, this can imply stability, as feasibility of a constrained optimization can imply invariance of the feasible set.

If a stronger form of stability is required, the cost of the optimization can be used as a Lyapunov function. If a particular solution is known to be feasible at time $t_0 + \Delta$, its cost is an upper bound on the optimal cost at that time. Hence the recursion can be used to establish $V(t_0 + \Delta) < V(t_0)$ where $V(t)$ is the optimal cost of the plan at time t. If V also meets the requirements for a Lyapunov function, which is common as optimizations typically have positive definite cost functions, then stability is proven [24]. Note that properties based on the monotonicity of the cost function typically require optimal solutions to be found at each step, whereas some other properties dependent only on constraints require only feasible solutions.

The nominal stability of MILP-based MPC has been proven for systems involving both integer and continuous states and inputs [9]. Where MILP has been employed for problems with nonconvex constraints, robust feasibility and convergence has been proven [26]. Stability has also been proven for a MILP-based control with an approximate terminal cost [25] (see Section VI-C for details).

C. Simple MPC Example

Consider the problem of controlling a linear time-invariant system

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k)$$

subject to state and control constraints

$$\mathbf{x}(k) \in \mathcal{X},\ \mathbf{u}(k) \in \mathcal{U}$$

which may include integrality constraints, *i.e.* some states and controls are integers representing decisions) or non-convex constraints using techniques from Section III. The objective is to minimize a one-norm cost function

$$\min \sum_{k=0}^{\infty} \left(|\mathbf{u}(k)| + |\mathbf{x}(k)|\right)$$

A nominally stable MPC is achieved by using the following optimization in Step 1 of the algorithm in Section V-A

$$V(k) = \min_{\mathbf{u},\mathbf{x},\mathbf{y}} \sum_{j=0}^{N} \left(|\mathbf{u}(k+j|k)| + |\mathbf{x}(k+j|k)|\right) \qquad (2a)$$

$$\text{subject to } \forall j \in \{0 \dots N\}$$

$$\mathbf{x}(k+j+1|k) = \mathbf{A}\mathbf{x}(k+j|k) + \mathbf{B}\mathbf{u}(k+j|k) \qquad (2b)$$

$$\mathbf{x}(k|k) = \mathbf{x}(k) \qquad (2c)$$

$$\mathbf{x}(k+N+1|k) = \mathbf{0} \qquad (2d)$$

$$\mathbf{x}(k+j|k) \in \mathcal{X} \qquad (2e)$$

$$\mathbf{u}(k+j|k) \in \mathcal{U} \qquad (2f)$$

The remainder of this section discusses the features of this optimization, the design features included, and how it can be extended for better performance.

- The double index notation $(k+j|k)$ denotes a prediction of a value j steps ahead from time k.
- N is the planning horizon. The optimization (2) approximates the original problem by solving only over this finite horizon and effectively constraining the plan to zero beyond.
- (2b) is the system dynamics model used for the predictions.
- (2c) is the initial condition constraint, ensuring that the plan starts from the current state.
- (2d) is the terminal constraint, required for stability [24]. This enforces that there is a feasible plan continuing on beyond the horizon, in this case all zero.
- (2e) and (2f) are the constraints applied to the plan.

This is a very simple MPC formulation that is generally-applicable. The optimization is a MILP, using slack variables to implement the minimization of the one-norm cost [3]. The stability of this MPC is proven using the recursion described in Section V-B. If a solution $\{\mathbf{u}(k_0|k_0), \mathbf{u}(k_0+1|k_0), \dots, \mathbf{u}(k_0+N-1|k_0), \mathbf{u}(k_0+N|k_0)\}$ is feasible at time k_0, then the candidate solution $\{\mathbf{u}(k_0+1|k_0), \dots, \mathbf{u}(k_0+N-1|k_0), \mathbf{u}(k_0+N|k_0), \mathbf{0}\}$ must be feasible at time k_0+1. This implies feasibility of all optimizations and, because the stage cost includes a positive definite term in the state $|\mathbf{x}|$, stability [9].

There are two straightforward extensions to this formulation. The first is the use of a more general terminal constraint, replacing (2d) with $\mathbf{x}(k+N+1|k) \in \mathcal{X}_F$, where $\mathcal{X}_F$ is a control invariant set [24]. This is clearly more complicated than using the origin, but is is much less restrictive. Stability is retained, and the set of feasible initial states is enlarged. This may also be thought of as enlarging the domain of attraction of the controller.

The second extension is to account for uncertainty by employing constraint tightening [26], [36]. This approach replaces (2e) with a time-varying constraint $\mathbf{x}(k+j|k) \in \mathcal{X}(j)$ where $\mathcal{X}(j)$ is a sequence of sets that is monotonically non-increasing with j. A similar change is made to the control constraints (2e). These modifications have the effect of retaining a "margin" for the action of uncertainty and, under some assumptions, stability and constraint satisfaction can be guaranteed despite the action of disturbances.

VI. Solving MILP Online

Section III showed that MILP can be used to model many useful forms of optimization problem, especially planning problems with discrete decisions or nonconvex constraints. Section V showed that by employing these optimizations online within MPC, feedback control can be realized with useful, meaningful properties such as stability and robustness. This section considers the remaining challenge: how to solve the MILP sufficiently quickly to apply the result in real-time.

Section IV described some general ways of solving MILPs. This section discusses further enhancements that are specific to the employment of MILP for online control problems. The principle is to use knowledge of the control problem to simplify the MILP before starting the optimizer, *e.g.* CPLEX, which has no knowledge of the structure of the problem.

A. Approximation

This section describes additional approximations that can be used to simplify the problems given previously in order to reduce the solution time. The most useful approach is to use prior knowledge to identify redundant or inactive constraints. These can then be removed from the problem, which typically leads to a faster solution time. Several strategies are presented for identifying these removable constraints, each is well suited to a particular class of problems.

Removal of Constraints. In some cases, it is possible to predict that the optimal solution of a constrained optimization will have a particular form for which some constraints will be inactive and thus can be removed from the problem. For example, minimum fuel spacecraft maneuvering problems with plume and avoidance constraints are still typically of the "bang-off-bang" form, so the plume constraints will be inactive during the coast phase [8]. This prior knowledge can be exploited to omit some of the plume constraints during the anticipated coasting phase before solving the problem. The reduced number of binary variables and constraints typically leads to a faster solution time. However, it is then necessary to verify that no plume impingement occurred at the steps where the constraints were removed. This can be done quickly, and if no impingement is found, the result is also the optimal solution to the completely constrained problem. Similar work has been done for collision avoidance [29]. In general, an iterative scheme is employed: constraints are removed initially. Then, at each iteration, feasibility of the solution to the relaxed problem is checked, and constraints are re-applied for the next iteration. Under the right circumstances, solving a number of relaxed problems is faster than solving the complete problem once.

Time-Step Grouping. When binary variables are replicated at each time step, for example as binary states or in avoidance constraints, the computation can be simplified by "sharing" the variables across adjacent time steps. For example, in a collision avoidance problem, as vehicles and obstacles move past each other, the interactions typically last for several time-steps at least, due to the comparatively large avoidance regions. Consequently, the binary variable settings for those regions will typically be identical over a long sequence of steps. For example, if a vehicle is on one "side" of an avoidance region at a certain time-step, it is likely to be on the same side at adjacent time-steps. This prediction can be used by "sharing" binary variables across small groups of adjacent time-steps. This essentially equates the binary variables across groups, and these extra constraints add conservatism to the problem. However, the prior assumption of equal binary settings predicts that the solution to the original problem would also satisfy these new constraints, so the expected fuel penalty is small. Time-step grouping does significantly reduce the problem size [8].

Making the Problem Completely Well-Posed. In Ref. [10], Bemporad and Morari discuss the notion of a *well-posed* problem in which binary variables are completely determined by the settings of the corresponding continuous variables. This concept helps identify inefficient modelling formulations. If a problem is not completely well-posed, it could indicate that there are unnecessary binary variables in the model. Experience has suggested, however, that there are some cases where taking a problem that is not well-posed and constraining it further to make it well-posed can slow down the solution process.

B. Feasible Initialization

Section V-B discussed how the fundamental properties of MPC were established using a recursion. If a solution was found at time t_0, then a candidate solution was guaranteed to be feasible at time $t_0 + \Delta$. If this candidate solution can be constructed explicitly and quickly, as is commonly the case, then it can be used in two ways to assist real-time operation.

The first method is to simply apply a time limit to the optimization and use the best solution found in the time available. If no better solution was found, then the candidate solution can be employed. This allows MPC to be operated as an "anytime" algorithm: an arbitrarily short computation time limit can be applied.

The second method uses the candidate solution to initialize the branch-and-bound procedure. Recall from Section IV-A that the algorithm would stop searching a particular branch when it was clear that no solution would be found that improved upon the current best. By giving the algorithm an integer solution at initialization, the "current best" starts lower, and branches can be terminated earlier, leading to faster computation.

The level of benefit from both of these methods is strongly dependent on the quality of the candidate solution. If the candidate solution is very close to the global optimum, then there is little degradation in performance if the candidate is used when a time limit is hit, and the

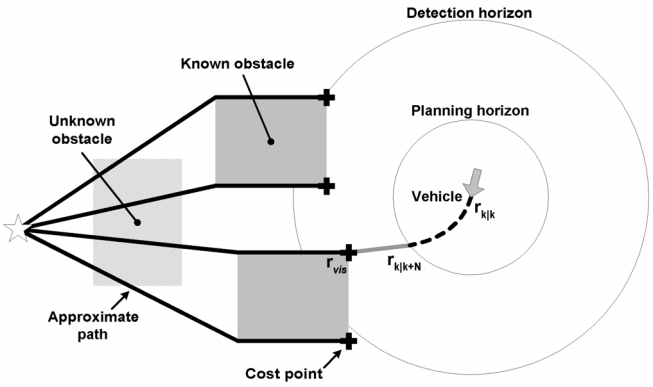

Fig. 2. Example of Cost-to-Go Method for

branch-and-bound method is accelerated a great deal by knowledge of a tight bound on the optimal cost. However, the candidate solution is by definition sub-optimal: if the optimal solution could be constructed, there would be no need for the MILP/MPC at all.

C. Approximate Cost-to-Go

Another way of simplifying the optimization is to shorten the planning horizon and employ an approximate cost-to-go to represent the plan beyond the horizon. If the cost-to-go were exactly known, then by the principle of optimality, receding horizon control would yield globally optimal solutions. However, the exact cost-to-go is typically too complex to calculate and in some cases uncertain. The cost-to-go method has the intuitive benefit of concentrating the detailed computation on areas where uncertainty is low, *i.e.* the near future, within the horizon, and reducing computation effort where uncertainty is high, beyond the horizon. The challenge then is to employ cost functions that are fast to compute and give good performance when applied in the uncertain environment.

Fig. 2 shows an example of this technique applied to a trajectory design problem with obstacle avoidance [28]. The controller uses a detailed trajectory model in the near term and an approximate path in the long term. This approach has been proven to guarantee the arrival at the target in bounded time. The approximation uses graph search techniques to generate a path composed of straight lines, ignoring dynamics constraints but cognisant of the presence of obstacles. This combination gives a good estimate of the cost-to-go and greatly reduces the computational effort required to design the complete trajectory. Discrepancies in the assumptions made in the two models are handled by ensuring that the planning horizon is sufficiently long [25].

VII. Examples

Recent examples of applications of MILP for real-time control include:

- UAV flight control [30]
- Aircraft scheduling problem [31]
- Helicopter Flight Control [33]
- Rover trajectory design [34], [32], [29]
- Spacecraft control (rendezvous and docking and formation flying) [8]
- Heat exchanger control [35]

The other talks in this tutorial session describe some of these applications in detail.

VIII. Conclusions

MIP has been shown to provide a modelling framework for systems involving both discrete and continuous decisions. Feedback control for such systems can be implemented by employing MIP for online control within MPC. With some additional constraints on the optimization model, MIP/MPC can be shown to be stable, feasible and robust. A key concern for implementing this type of controller is the computation time required to solve the optimization. Some general techniques have been described for achieving good solutions in real-time. The remaining talks in this tutorial give further details of specific MIP/MPC applications.

References

[1] Website, "Decision Tree for Optimization Software – Discrete," `plato.la.asu.edu/topics/problems/discrete.html`, Arizona State University, visited February 2005.

[2] M. Alighanbari, Y. Kuwata, and J. How, "Coordination and Control of Multiple UAVs with Timing Constraints and Loitering," *ACC*, June 2003.

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

[4] C. A. Floudas *Nonlinear and Mixed-Integer Programming – Fundamentals and Applications* Oxford University Press, 1995.

[5] Williams, H.,Brailsford, S., "Computational Logic and Integer Programming," in *Advances in Linear and Integer Programming,* Editor J. E. Beasley, Clarendon Press, Oxford, 1996, pp. 249–281.

[6] Burkard, R., Çela, E., "Linear Assignment Problems and Extensions," In Z. Du and P. Pardalos, editors, *Handbook of Combinatorial Optimization,* pp. 75-149. Kluwer Academic Publishers, 1999.

[7] Schouwenaars, T., DeMoor, B., Feron, E., How, J., "Mixed integer programming for safe multi-vehicle cooperative path planning," presented at the 2001 *European Control Conference*.

[8] A. Richards, T. Schouwenaars, J. P. How, and E. Feron, "Spacecraft Trajectory Planning With Collision and Plume Avoidance Using Mixed-Integer Linear Programming," AIAA *Journal of Guidance, Control, and Dynamics,* Vol. 25, No. 4, pp. 755–765, Aug. 2002.

[9] A. Bemporad and M. Morari, "Control of Systems Integrating Logic, Dynamics, and Constraints," in *Automatica,* Pergamon / Elsevier Science, New York NY, Vol. 35, 1999, pp. 407–427.

[10] A Bemporad and M Morari, "Robust Model Predictive Control: A Survey," In Robustness in Identification and Control, A Garulli, A Tesi and A Vechino (Eds.), Springer Verlag, 1999, p.207-226.

[11] A. Bemporad, G. Ferrari-Trecate, and M. Morari. "Observability and controllability of piecewise affine and hybrid systems," IEEE *Trans. Autom. Contr.,* 45(10):1864–1876, 2000.

[12] E.D. Sontag. "Interconnected automata and linear systems: A theoretical frameworkin discrete-time," In R. Alur, T. A. Henzinger, and E. D. Sontag, (eds.), *Hybrid Systems III - Verification and Control* number 1066 in Lecture Notes in Computer Science, pages 436 448. Springer-Verlag, 1996.

[13] D.S. Johnson and L.A. McGeoch, The Traveling Salesman Problem: A Case Study in Local Optimization, in *Local Search in Combinatorial Optimization*, E.H.L. Aarts and J.K. Lenstra (eds.), John Wiley & Sons, New York, NY, 1997, 215–310.

[14] P. Toth and D. Vigo (eds.), *The Vehicle Routing Problem*, published by the Society for Industrial & Applied Mathematics, Dec. 2001.

[15] , C. C. Pantelides, M. J. Realff, N. Shah, "Short-Term Scheduling of Pipeless Batch Plants." *Chemical Engineering Research and Design* 73 (A4), 1995, pp. 431–444.
[16] *ILOG CPLEX User's guide*. ILOG, 1999.
[17] R. Fourer, D. M. Gay, and B. W. Kernighar. *AMPL, A modeling language for mathematical programming*. The Scientific Press, 1993.
[18] D. P. Bertsekas, *Dynamic Programming and Optimal Control,* Athena Scientific, Belmont, Massachusetts, 2000.
[19] J. Linderoth and T.Ralphs, "Noncommercial Software for Mixed-Integer Linear Programming", Technical Report 04T-023, Department of Industrial and Systems Eng., Lehigh Univ., Dec 2004. `www.lehigh.edu/~jtl3/teaching/ie418/papers/MILP04.pdf`
[20] F. Glover and M. Laguna, *Tabu Search*, Kluwer 00362: Acad. Publ., 1997.
[21] I. Harjunkoski, P. Kabore, M. Fahl, "An Efficient MILP-Heuristic Approach for Solving Large-Scale Multi-Stage Scheduling Problems", *AIChE Annual Meeting,* Austin (TX), USA, November 7-12, 2004
[22] J.M. Maciejowski, *Predictive Control with Constraints*, Prentice Hall, England, 2002.
[23] J. A. Rossiter, "Model-based Predictive Control – A Practical Approach," CRC Press, Florida, 2003.
[24] D. Q. Mayne, J. B. Rawlings, C. V. Rao, P. O. M. Scokaert, "Constrained Model Predictive Control: Stability and Optimality," *Automatica*, 36(2000), Pergamon Press, UK, pp. 789–814.
[25] Y. Kuwata and J. How, "Stable Trajectory Design for Highly Constrained Environments using Receding Horizon Control," IEEE ACC 2004.
[26] A. Richards and J. P. How, "Model Predictive Control of Vehicle Maneuvers with Guaranteed Completion Time and Robust Feasibility," IEEE *ACC,* June 2003.
[27] J. Linderoth and M. Savelsbergh, "A Computational Study of Branch and Bound Search Strategies for Mixed Integer Programming," *INFORMS Journal on Computing,* 11 (1999) pp. 173–187.
[28] J. Bellingham, A. Richards, and J. How, "Receding Horizon Control of Autonomous Aerial Vehicles," in *Proceedings of the American Control Conference*, May 2002.
[29] M. G. Earl and R. d'Andrea, "Iterative MILP Methods for Vehicle Control Problems", IEEE *CDC* 2004.
[30] Valenti, M., Schouwenaars, T., Kuwata, Y., Feron, E., How, J., and Paunicka, J., "Implementation of a Manned Vehicle-UAV Mission System," AIAA *Guidance, Navigation, and Control Conference and Exhibit,* Providence, RI, August 2004.
[31] A. Bayen, J. Zhang, C. Tomlin and Y. Ye, "MILP formulation and polynomial time algorithm for an aircraft scheduling problem," *Conference on Decision and Control,* 2003.
[32] M. Mukai, T. Azuma and M. Fujita, "A Collision Avoidance Control for Multi-Vehicle Using PWA/MLD Hybrid System Representation," *IEEE Conference on Control Applications* pp. 872-877, Taipei, Taiwan, Sep. 2004.
[33] T. Schouwenaars, B. Mettler, E. Feron, and J. How, "Hybrid Model for Trajectory Planning of Agile Autonomous Vehicles," AIAA *Journal of Aerospace Computing, Information, and Communication* Vol. 1, Dec. 2004, pp. 629–651
[34] A. Richards, Y. Kuwata, and J.P. How, "Experimental Demonstrations of Real-time MILP Control," Proceeding of the AIAA *Guidance, Navigation, and Control Conference,* August 2003. AIAA-2003-5802
[35] A. Bemporad, F. Borrelli, and M. Morari, "Optimal Controllers for Hybrid Systems: Stability and Piecewise Linear Explicit Form," IEEE *Conference on Decision and Control,* Sydney, Australia , December 2000
[36] J. R. Gossner, B. Kouvaritakis and J. A. Rossiter, "Stable Generalized Predictive Control with Constraints and Bounded Disturbances," *Automatica,* Vol 33 No 4, Pergamon Press, UK, 1997, p.551.