Logic-Based Solution Methods for Optimal Control of Hybrid Systems

Alberto Bemporad, *Member, IEEE*, and Nicolò Giorgetti, *Student Member, IEEE*

***Abstract*—Combinatorial optimization over continuous and integer variables is a useful tool for solving complex optimal control problems of hybrid dynamical systems formulated in discrete-time. Current approaches are based on mixed-integer linear (or quadratic) programming (MIP), which provides the solution after solving a sequence of relaxed linear (or quadratic) programs. MIP formulations require the translation of the discrete/logic part of the hybrid problem into mixed-integer inequalities. Although this operation can be done automatically, most of the original symbolic structure of the problem (e.g., transition functions of finite state machines, logic constraints, symbolic variables, etc.) is lost during the conversion, with a consequent loss of computational performance. In this paper, we attempt to overcome such a difficulty by combining *numerical* techniques for solving convex programming problems with *symbolic* techniques for solving constraint satisfaction problems (CSP). The resulting "hybrid" solver proposed here takes advantage of CSP solvers for dealing with satisfiability of logic constraints very efficiently. We propose a suitable model of the hybrid dynamics and a class of optimal control problems that embrace both symbolic and continuous variables/functions, and that are tailored to the use of the new hybrid solver. The superiority in terms of computational performance with respect to commercial MIP solvers is shown on a centralized supply chain management problem with uncertain forecast demand.**

***Index Terms*—Constraint satisfaction, hybrid systems, optimal control, optimization.**

I. Introduction

OVER the last few years, we have witnessed a growing interest in the study of dynamical processes of a mixed continuous and discrete nature, denoted as hybrid systems, both in academia and in industry. Hybrid systems are characterized by the interaction of continuous models, describing continuous variables governed by differential or difference equations, and of discrete models, describing symbolic variables governed by logic rules, switching mechanisms, and other discrete behaviors. Hybrid systems can switch between many operating modes where each mode is governed by its own characteristic continuous dynamical laws. Mode transitions may be endogenous (variables crossing specific thresholds), or exogenous (discrete commands directly given to the system). The interest in hybrid systems is mainly motivated by the large variety of practical situations where physical processes interact with digital controllers, as for instance in embedded control systems.

Despite the fact that one of the first papers on hybrid systems appeared in the 1960s [1], only in recent years several modelling frameworks for hybrid systems were proposed; we refer the interested reader to [2], [3], and references therein. Several authors focused on the problem of solving optimal control problems for hybrid systems. For continuous-time hybrid systems, most of the literature either studied necessary conditions for a trajectory to be optimal, or focused on the computation of optimal/suboptimal solutions by means of dynamic programming or the maximum principle [4]–[6] (see also the survey paper [7]).

The hybrid optimal control problem becomes less complex when the dynamics is expressed in discrete time, as the main source of complexity becomes the combinatorial (yet finite) number of possible switching sequences. In particular for discrete-time hybrid systems, in [8]–[10] the authors have solved optimal control problems by transforming the hybrid model into a set of linear equalities and inequalities involving both real and (0–1) variables, so that the optimal control problem can be solved by a mixed-integer programming (MIP) solver [11]. Examples of applications where hybrid modeling and optimal control based on MIP were employed are reported in [12]–[16].

An MIP solver provides the solution after solving a sequence of relaxed standard linear (or quadratic) programs (LP, QP). A potential drawback of MIP is 1) the need for converting the discrete/logic part of the hybrid problem into mixed-integer inequalities, therefore, losing most of the original discrete structure, and 2) the fact that its efficiency mainly relies upon the tightness of the continuous LP/QP relaxations.

Such drawbacks are not suffered by techniques for solving CSP, i.e., the problem of determining whether a set of constraints over discrete variables can be satisfied. Under the class of CSP solvers we mention constraint logic programming (CLP) [17] and SAT solvers [18], the latter specialized for the satisfiability of Boolean formulas.

While CSP methods are superior to MIP approaches for determining if a given problem has a feasible (discrete-valued) solution, the main drawback is their inefficiency for solving optimization, as they do not have the ability of MIP approaches to solve continuous relaxations (e.g., linear programming relaxations) of the problem in order to get upper and lower bounds to the optimum value. For this reason, it is extremely interesting to integrate the two approaches into one single cooperative solver. Some efforts have been done in this direction

Manuscript received Noember 4, 2004; revised September 5, 2005. Recommended by Guest Editor M. Egerstedt. This work was supported by the European Community through Project "Computation and control" IST-2001-33520, MIUR through Project PRIN'03 "Models for optimization, control, and coordination of distributed production systems", and by the University of Siena through Project PAR'02 "Optimization methods for hybrid systems."

A. Bemporad is with the Dipartimento di Ingegneria dell'Informazione, University of Siena, Siena 53100, Italy (e-mail: bemporad@dii.unisi.it).

N. Giorgetti is with General Motors Powertrain Europe, Turin, Italy.

Digital Object Identifier 10.1109/TAC.2006.876949

[19]–[23], showing that such "hybrid" solution methods have a tremendous performance in solving mathematical programs with continuous (quantitative) and discrete (logical/symbolic) components, compared to MIP or CSP individually. Such successful results have stimulated also the industrial interest of worldwide leaders in commercial software for combinatorial optimization: ILOG Inc. is currently distributing optimization programming language (OPL), a modeling and programming language which allows the formulation and solution of optimization problems, using both MIP and CSP techniques, combining to some extent the advantages of both approaches; Dash Optimization is releasing Xpress together with Xpress-CP, an interface to the constraint engine COSYTEC SA, which allows the user to combine different solvers. The study on cooperative solvers was also stimulated in the context of supply chain management; we quote here the main motivations which lead to the funding of the European project Large Scale Integrated Supply Chain Optimization Software (LISCOS): "Formulating production planning and other similar problems in terms of mixed integer programming (MIP) has the advantages of simplicity of modelling and the use of powerful existing solvers for the canonical form (typically a matrix representation) of the resulting numerical instance of the model. However, a straightforward conversion of a MIP formulation into a canonical representation leads to a loss of the semantics of the problem, which is often required for effective solution even when using sophisticated cutting plane identification and clarification." (source: http://www.liscos.fc.ul.pt/approach.html).

In light of the aforementioned promising attempts at combining numerical and symbolic techniques, done in the interdisciplinary field of artificial intelligence and operations research and in a few application domains, in this paper, we propose an original cooperative approach of MIP and CSP techniques that combines convex programming (e.g., linear, quadratic, etc. [24]) for optimization over real variables, and of SAT solvers or CLP techniques for determining the satisfiability of Boolean formulas (or logic constraints), and apply it in the context of optimal control of hybrid dynamical systems.

Based on the benefits and drawbacks of the previous work in [8]–[10] for solving control and stability/safety analysis problems for hybrid systems using MIP techniques, here we build up a new modeling approach directly tailored to the use of a "hybrid" MIP+CSP solver and show its computational advantages over pure MIP methods. Preliminary results along this direction appeared in the conference papers [25]–[27].

The paper is organized as follows. Discrete-time hybrid models are introduced in Section II. In Section III, the optimal control problem is formulated and, in Section IV, it is reformulated in a suitable way for the combined MIP-CSP approach. Section V introduces the new solution algorithms. Section VI extends the optimal control setting to the case of uncertainty in the references and in the constraint bounds. An application example showing the benefits of this technique, compared to pure MIP approaches [8], [10] is shown in Section VII.

II. Discrete-Time Hybrid Automata

Several class of hybrid dynamical system models were proposed in the literature, each usually tailored to solve a particular

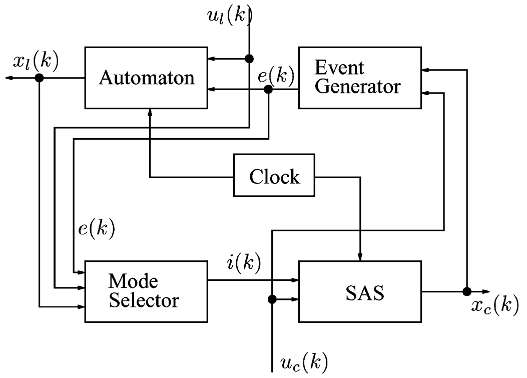


Fig. 1. Discrete-time hybrid automaton.

problem. Timed automata [28] and linear hybrid automata [29] were proved successful for formal verification, whereas mixed logical dynamical (MLD) models [8] or piecewise affine models (PWA) [30] were proved suitable for control and analysis purposes. Here, we focus on the class of discrete hybrid automata (DHA) introduced in [10]. A DHA is the interconnection of an automaton (AUT), which models the discrete dynamics of the hybrid system, with a discrete-time switched affine system (SAS), which models the continuous dynamics of the hybrid system, through an event generator (EG) and a mode selector (MS) (see Fig. 1). For convenience, we briefly recall the DHA formalism in the following and refer the reader to [10] for details.

A. Automaton

The discrete dynamics of a hybrid system is modeled as an automaton (or finite state machine). We refer here to "synchronous automata," as we assume transitions are clocked and synchronous with the sampling time of the continuous dynamics. The automaton evolves according to the following logic state update function:

$$x_l(k+1) = f_l\left(x_l(k), u_l(k), e(k)\right) \tag{1}$$

where $k \in \mathbb{Z}^+ = \{0, 1, \ldots\}$ is the time index, $x_l \in \mathcal{X}_l \subseteq \{0,1\}^{n_l}$ is the logic state, $u_l \in \mathcal{U}_l \subseteq \{0,1\}^{m_l}$ is the exogenous logic input, $e \in \mathcal{E} \subseteq \{0,1\}^{n_e}$ is the endogenous input coming from the EG defined in Section II-C, and $f_l : \mathcal{X}_l \times \mathcal{U}_l \times \mathcal{E} \to \mathcal{X}_l$ is a deterministic Boolean function. An automaton can be represented as a directed graph. In the sequel, with a slight abuse of notation, we will refer to the codomain of Boolean functions both as {0,1} and as $\{\texttt{FALSE}, \texttt{TRUE}\}$. In the context of Boolean functions and formulas, the equal sign $(=)$ should be interpreted as an if-and-only-if condition $(\longleftrightarrow)$.

B. Switched Affine System

The continuous dynamics is modeled by a discrete-time switched affine system (SAS). A SAS is a collection of affine systems

$$x_c(k+1) = A_{i(k)}x_c(k) + B_{i(k)}u_c(k) + f_{i(k)} \tag{2}$$

where $x_c \in \mathcal{X}_c \subseteq \mathbb{R}^{n_c}$ is the continuous state vector, $u_c \in \mathcal{U}_c \subseteq \mathbb{R}^{m_c}$ is the exogenous continuous input vector, $i(k) \in \mathcal{I} \triangleq \{[1\,0\ldots0]^T, [0\,1\ldots0]^T, \ldots, [0\ldots0\,1]^T\} \subset \{0,1\}^s$ is the "mode" in which the SAS is operating, $|\mathcal{I}| = s$ is the number of elements of $\mathcal{I}$, and $\{A_i, B_i, f_i\}_{i\in\mathcal{I}}$ is a collection of matrices of opportune dimensions. The mode $i(k)$ is generated by the MS, as described in Section II-D. A SAS of the form (2) preserves the value of the state when a switch occurs. However, resets can be modeled in the present discrete-time setting as detailed in [10].

C. Event Generator

An event generator is a mathematical object that generates a binary signal $e_j(k) \in \{0,1\}$ according to the satisfaction of a linear affine constraint

$$[e_j(k) = 1] \longleftrightarrow \left[a_j^T x_c(k) + b_j^T u_c(k) \le c_j\right] \tag{3}$$

where the subscript j denotes the jth component of the vector, and $a_j \in \mathbb{R}^{n_c}$, $b_j \in \mathbb{R}^{m_c}$, and $c_j \in \mathbb{R}$ define a linear guard (i.e., a hyperplane) in the space of continuous states and inputs.

D. Mode Selector (MS)

The dynamic mode $i(k)$ of the SAS is selected through a *mode selector (MS)*

$$i(k) = f_{\mathrm{MS}}\left(x_l(k), u_l(k), e(k)\right) \tag{4}$$

where $f_{MS} : \mathcal{X}_l \times \mathcal{U}_l \times \mathcal{E} \to \mathcal{I}$ is a Boolean function of the logic state $x_l(k)$, of the logic input $u_l(k)$, and of the logic event signal $e(k)$. We say that a *mode switch* occurs at step k if $i(k) \neq i(k-1)$. Note that contrary to continuous time or discrete-event hybrid models, where switches can occur at any time, in our discrete-time setting a mode switch can only occur at sampling instants.

III. Optimal Control

A finite-time optimal control problem for the class of hybrid systems introduced in Section II can be formulated as follows:

$$\min_{\{x(k+1),u(k)\}_{k=0}^{T-1}} \sum_{k=0}^{T-1} \ell_k\left(x(k+1) - r_x(k+1), u(k) - r_u(k)\right) \tag{5a}$$

s.t.

$$\text{dynamics } (1), (2), (3), (4) \tag{5b}$$

$$h_D^j\left(x(0), \{x(k+1), u(k), e(k), i(k)\}_0^{T-1}\right) = \texttt{TRUE} \tag{5c}$$

$$h_A^t\left(x(0), \{x(k+1), u(k), e(k), i(k)\}_0^{T-1}\right) = \texttt{TRUE},\quad j = 1,\ldots,p_A, \qquad t = 1,\ldots,p_B \tag{5d}$$

where T is the control horizon, $\ell_k : \mathbb{R}^{n+m} \to \mathbb{R}$ are nonnegative convex functions ($k = 0,\ldots,T-1$), $n = n_c + n_l$, $m = m_c + m_l$, $r_x(k+1) \in \mathbb{R}^n$, $r_u(k) \in \mathbb{R}^m$, ($k = 0,\ldots,T-1$) are given reference values to be tracked by the state and input vectors, respectively, $h_D : \mathbb{R}^{(n+m)T} \times \{0,1\}^{(n_e+s)T} \to \{\texttt{TRUE}, \texttt{FALSE}\}^{p_A}$, and $h_A : \mathbb{R}^{(n+m)T} \times \{0,1\}^{(n_e+s)T} \to \{\texttt{TRUE}, \texttt{FALSE}\}^{p_B}$.

The constraints in (5) are classified in three different categories.

- Dynamical constraints (5b).
  They represent the discrete-time dynamics of the hybrid automaton. They may also include other constraints such as saturation of continuous input variables that are embodied in the variable domain $\mathcal{U}_c$.
- Design constraints (5c).
  These are artificial constraints imposed by the designer to fulfill the required specifications. Examples of such constraints may be state limits

  $$[x_{min}(k) \le x_c(k) \le x_{max}(k)] = \texttt{TRUE}$$

  where $x_{min}(k)$ and $x_{max}(k)$ are bounds that the designer wants to impose on continuous states $\forall k = 1,\ldots,T$.
- Ancillary constraints (5d).
  These constraints provide an *a priori* additional and auxiliary information for determining the optimal solution. They do not change the solution itself, rather help the solver by restricting the set of feasible combinations, and, therefore, the size of the decision tree in a branch a bound strategy. For example, as detailed in Section IV-C, one may precompute all possible mode transitions of the SAS dynamics using reachability analysis, and impose *reachability constraints* whenever a transition from the hth mode to the jth mode is not possible.

IV. Problem Reformulation

Problem (5) can be solved via MILP when the costs ℓ_k are convex piecewise linear functions, for instance

$$\ell_k(x,u) = \|Q_x x\|_\infty + \|Q_u u\|_\infty \tag{6}$$

where Q_x and Q_u are full-rank matrices and $\|\cdot\|_\infty$ denotes the infinity-norm ($\|Qx\|_\infty = \max_{j=1,\ldots,n} |Q^j x|$, where Q^j is the jth row of Q), or via mixed integer quadratic programming (MIQP) when

$$\ell_k(x,u) = x^T Q_x x + u^T Q_u u$$

where Q_x and Q_u are positive–semidefinite matrices [11].

As described in [8] and [10], by adopting the so-called "big-M" technique the event generator (3) can be equivalently expressed as

$$\left(a_j^T x_c(k) + b_j^T u_c(k) - c_j\right) \le M_j\left(1 - e_j(k)\right) \tag{7a}$$

$$\left(a_j^T x_c(k) + b_j^T u_c(k) - c_j\right) > m_j e_j(k) \tag{7b}$$

where $j = 1,\ldots,n_e, M_j, m_j$ are known upper and lower bounds, respectively, on $a_j^T x_c(k) + b_j^T u_c(k) - c_j$, and

$e_j(k) \in \{0,1\}$. From a computational viewpoint, it is convenient to have a set of inequalities without strict inequalities, so we follow the common practice [31] of replacing the strict inequality (7) as

$$\left(a_j^T x_c(k) + b_j^T u_c(k) - c_j\right) \geq \epsilon + (m_j - \epsilon) e_j(k) \tag{7c}$$

where ϵ is a small positive scalar, e.g., the machine precision, although the equivalence does not hold for $0 < (a_j^T x_c(k) + b_j^T u_c(k) - c_j) < \epsilon$ [i.e., for the numbers in the interval $(0,\epsilon)$ that cannot be represented in the machine]. The continuous state update equation of the SAS dynamics (2) can be equivalently written as the combination of linear terms and *if-then-else* rules

$$w_i(k) = \begin{cases} A_i x_c(k) + B_i u_c(k) + f_i, & \text{if } (\delta_i = 1) \\ 0, & \text{otherwise} \end{cases} \tag{8a}$$

$$x_c(k+1) = \sum_{i=1}^{s} w_i(k) \tag{8b}$$

where $w_i(k) \in \mathbb{R}^{n_c}$, $i = 1, \ldots, s$. The SAS representation (8) can be translated into a set of constraints by also using the big-M technique [31]

$$-M_i^j \delta_i(k) + w_i(k) \leq 0 \tag{9a}$$

$$m_i^j \delta_i(k) - w_i(k) \leq 0 \tag{9b}$$

$$m_i^j (1 - \delta_i(k)) + w_i(k) \leq A_i^j x_c(k) + B_i^j u_c(k) + f_i^j \tag{9c}$$

$$-M_i^j (1 - \delta_i(k)) - w_i(k) \leq -A_i^j x_c(k) - B_i^j u_c(k) - f_i^j \tag{9d}$$

where M_i^j and m_i^j are known upper and lower bounds on $A_i^j x_c(k) + B_i^j u_c(k) + f_i^j$, $\delta_i(k) \in \{0,1\}$, $w_i(k) \in \mathbb{R}^{n_c}$, $x_c \in \mathbb{R}_c^n$, and $u \in \mathbb{R}_c^m$, the superscript ^{j} denotes the jth component or row $j = 1, \ldots, n_c$, $i = 1, \ldots, s$, and k is the time index. Note that the binary mode vector $i(k) = [\delta_1(k) \ldots \delta_s(k)]^T \in \{0,1\}^s$ is subject to the logical "exclusive or" condition

$$\delta_1(k) \oplus \delta_2(k) \oplus \ldots \oplus \delta_s(k) = \texttt{TRUE}. \tag{10}$$

In this paper, we wish to solve problem (5) by using MIP and CSP techniques in a combined approach, taking advantage of CSP algorithms for dealing with the purely logic part of the problem. Contrary to the MLD approach of [8] and [10], where also the automaton and MS parts of the hybrid system are converted to (mixed) integer linear inequalities, here we keep their symbolic description as a set of Boolean functions that can be either directly managed by a CLP algorithm [32], or translated into conjunctive normal form (CNF) in case SAT algorithms are employed [33].

A. Alternative Relaxations of DHA

DHA is a mathematical representation of other equivalent computation-oriented and domain-specific frameworks such as *mixed logical dynamical (MLD)* systems [8], *piecewise affine (PWA)* systems [30], and other frameworks [34]. In this section, we propose a new alternative approach to obtain tighter relaxations than the big-M relaxations (7), (9) by using disjunctive programming ideas [35], and the fact that DHA can be converted to PWA models [36].

Consider piecewise affine (PWA) systems described in discrete-time by the relations

$$x(k+1) = A_{i(k)} x(k) + B_{i(k)} u(k) + f_{i(k)} \tag{11}$$

for $\begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \in P_{i(k)}$, where

$$P_i \triangleq \left\{ \begin{bmatrix} x \\ u \end{bmatrix} : H_i^x x + H_i^u u \leq K_i \right\} \tag{12}$$

$i = 1, \ldots, s$ are convex polytopes in the input+state space. A_i and B_i are matrices of appropriate dimensions and f_i are real vectors for all $i = 1, \ldots, s$.

Standard disjunctive programming techniques were used in [37] to model piecewise affine systems (11). We take a different route here and assume that polyhedral cells are given in V-representation (that is, as the convex hull of vertices and positive combination of extreme rays), rather than in H-representation (intersection of half spaces defined by hyperplanes)

$$\begin{aligned} P_i &= \operatorname{conv}\left\{v_i^1, \ldots, v_i^{n_i}\right\} + \operatorname{cone}\left\{w_i^1 \ldots, w_i^{m_i}\right\} \\ &= \Bigg\{ z = \begin{bmatrix} x \\ u \end{bmatrix} \in \mathbb{R}^{n+m} : z = \sum_{j=1}^{n_i} \lambda_i^j v_i^j + \sum_{h=1}^{m_i} \mu_i^h w_i^h, \\ &\qquad \lambda_i^j, \mu_i^h \geq 0, \sum_{j=1}^{n_i} \lambda_i^j = 1 \Bigg\}, \qquad i = 1, \ldots, s \end{aligned} \tag{13}$$

The problem of transforming a V- to a H-representation (or vice versa) is known in computational geometry as the problem of *vertex enumeration*, and several tools are available (see e.g., [38]).

Lemma 1: Given a collection $\mathcal{P} = \{P_1 \ldots, P_s\}$ of (possibly unbounded) polyhedra in V-representation (13) and an arbitrarily large box $B = \{z \in \mathbb{R}^{n+m} : \|z\|_\infty \leq M\}$ for some $M > 0$, introduce s binary variables $\delta_1, \ldots, \delta_s \in \{0,1\}$ and define the set

$$S \triangleq \{ z \in \mathbb{R}^{n+m} :$$

$$z = \sum_{i=1}^{s} \left(\sum_{j=1}^{n_i} \lambda_i^j v_i^j + \sum_{h=1}^{m_i} \mu_i^h w_i^h \right) \tag{14a}$$

$$\begin{aligned} &\lambda_i^j \geq 0 \qquad \forall j = 1, \ldots, n_i \qquad \forall i = 1, \ldots, s \\ &\mu_i^h \geq 0 \qquad \forall h = 1, \ldots, m_i \qquad \forall i = 1, \ldots, s \end{aligned} \tag{14b}$$

$$\sum_{j=1}^{n_i} \lambda_i^j = \delta_i \qquad \forall i = 1, \ldots, s \tag{14c}$$

$$\mu_i^h \leq M_i^h \delta_i \qquad \forall i = 1, \ldots, s \tag{14d}$$

$$\sum_{i=1}^{s} \delta_i = 1 \qquad \forall i = 1, \ldots, s \tag{14e}$$

$$\delta_i \in \{0,1\} \qquad \forall i = 1, \ldots, s\} \tag{14f}$$

where each M_i^h is defined as the solution of the following linear program:

$$M_i^h \triangleq \max_{\substack{\{\lambda_i^j\},j=1,\ldots,n_i,\\ \{\mu_i^h\},h=1,\ldots,m_i}} \mu_i^h \tag{15a}$$

s.t.

$$\left\|\sum_{j=1}^{n_i}\lambda_i^j v_i^j + \sum_{h=1}^{m_i}\mu_i^h w_i^h\right\|_\infty \le M \tag{15b}$$

$$\lambda_i^j, \mu_i^h \ge 0 \quad \sum_{j=1}^{n_i}\lambda_i^j = 1. \tag{15c}$$

It holds that

i) $(B \cap \bigcup_{i=1}^s P_i) \subseteq S \subseteq (\bigcup_{i=1}^s P_i)$;

ii) if all polyhedra P_i are bounded, then $\bigcup_{i=1}^s P_i = S$;

iii) let $\hat{S}$ be defined as in (14) without constraints (14d) and (14f); then, $\hat{S} = \mathrm{conv}(P_1,\ldots,P_s)$.

Proof: i) Take any $z \in B \cap \bigcup_{i=1}^s P_i$. Then, $z \in P_t$ for some $t \in \{1,\ldots,s\}$, with $z = \sum_{j=1}^{n_t}\lambda_t^j v_t^j + \sum_{h=1}^{m_t}\mu_t^h w_t^h$ for some combination of λ_t^j and μ_t^j, with $\lambda_t^j \ge 0$, $\mu_t^j \ge 0$, and $\sum_{j=1}^{n_t}\lambda_t^j = 1$. Since $x \in B$, then by (15) each $\mu_t^h \le M_t^h$. Let $\delta_t = 1$ and $\delta_i = 0$ for all $i \ne t$, and let $\lambda_i^j = \mu_i^h = 0$ for all $i \ne t$, and for all j and h. Then, it immediately follows that $z \in S$. ii) If all polyhedra are bounded, then $m_i = 0$, $\forall i \in \{1,\ldots,s\}$ (no extreme rays). Take any $z \in \bigcup_{i=1}^s P_i$. By following the same reasoning above, it follows that $z \in S$. Vice versa, take any z in S. Then, for some $t \in \{1,\ldots,s\}$, the corresponding $\delta_t = 1$ and, hence, $\delta_i = 0$, $\forall i \ne t$. By (14c), this implies that $\sum_{j=1}^{n_t}\lambda_t^j = 1$ and that $\lambda_i^j = 0$, $\forall i \ne t$ and $\forall j$. Then, since $\lambda_t^j \ge 0$ ($\forall j = 1,\ldots,n_t$), by (14a) it follows that $z = \sum_{j=1}^{n_t}\lambda_t^j v_t^j$, which in turn implies that $z \in \bigcup_{i=1}^s P_i$. iii) By (14c) and (14e), it follows that $\sum_{i=1}^s\sum_{j=1}^{n_i}\lambda_i^j = 1$. Then, by (14a), $z \in \mathrm{conv}\{v_1^1,\ldots,v_s^{n_s}\} + \mathrm{cone}\{w_1^1,\ldots,w_s^{m_s}\} = \mathrm{conv}(P_1,\ldots,P_s)$. □

Remark 1: The PWA form obtained by translating a DHA system with the procedure described in [36] is always bounded, as the MLD representation used in an intermediate step is always bounded. For this reason, we do not need to define μ variables in (16), and we will restrict our attention to polytopes $P_i = \mathrm{conv}\{v_i^1 \ldots, v_i^{n_i}\}$ under the implicit assumption that the domain of validity of the PWA dynamics in the (x,u)-space is bounded.

Assume the DHA dynamics (1)–(4) is converted to the PWA form (11), (12), e.g., by using the procedure described in [36], and that the H-representation (12) is converted to the V-representation (13). Then, by exploiting Lemma 1, it is easy to prove the following proposition.

Proposition 1: System (11) and (13) can be equivalently described by the relations

$$x(k+1) = \sum_{i=1}^s\left(f_i\delta_i(k) + [A_iB_i]\sum_{j=1}^{n_i}\lambda_i^j(k)v_i^j\right) \tag{16a}$$

$$\sum_{i=1}^s\sum_{j=1}^{n_i}\lambda_i^j(k)v_i^j = \begin{bmatrix}x(k)\\u(k)\end{bmatrix} \tag{16b}$$

$$\sum_{j=1}^{n_i}\lambda_i^j(k) = \delta_i(k), \lambda_i^j(k) \ge 0$$
$$\forall j = 1,\ldots,n_i \qquad \forall i = 1,\ldots,s \tag{16c}$$

$$\sum_{i=1}^s \delta_i(k) = 1,$$
$$\delta_i(k) \in \{0,1\} \qquad \forall i = 1,\ldots,s \tag{16d}$$

where $[\delta_1(k),\ldots,\delta_s(k)]^T = i(k)$ is the current mode of the hybrid dynamics rewritten in PWA form.

Equation (16) is an alternative representation of the DHA dynamics (1)–(4). Accordingly, the optimal control problem (5) can be reformulated by changing (5b) into "dynamics (16)" and by adding the logic constraint (4) [relating the mode $i(k)$ to the logic variables $x_\ell(k)$, $u_\ell(k)$, and $e(k)$] and (1) [relating the logic state $x_\ell(k+1)$ to $x_\ell(k)$, $u_\ell(k)$, and $e(k)$] as ancillary constraints in (5d). As explained in the next section, such logical constraints are handled by the CSP solver and therefore exploited to drive more efficiently the search for the optimal solution.

Remark 2: In order to guarantee the uniqueness of $x(k+1)$, $\forall k \ge 0$, in (11) [or (16)], one must make sure that $P_i \cap P_j = \emptyset$, $\forall i \ne j$, $i,j \in \{1,\ldots,s\}$. To avoid that nonempty intersections occur because of overlapping boundaries, one can allow some of the inequalities in (12) to be strict: $H_i^x(t,:)x + H_i^u(t,:)u < K_i(t)$, where $H_i(t,:)$ represents the t-th row of matrix H_i, and $K_i(t)$ is the t-th element of vector K_i, and come back to closed polyhedra by replacing the strict inequality as $H_i^x(t,:)x + H_i^u(t,:)u \le K_i - \epsilon$ where ϵ is a small positive scalar, e.g. the machine precision, as observed earlier.

B. Logic-Based Form of the Optimal Control Problem

By using the transformations into mixed integer inequalities based on the big-M relaxations (7), (9), problem (5) can be cast as the mixed-integer convex program

$$\min_{\substack{\{x(k+1),u(k),\\ w(k),\delta(k),e(k)\}\\ k=0,\ldots,T-1}} \sum_{k=0}^{T-1}\ell_k\left(x(k+1)-r_x(k+1), u(k)-r_u(k)\right) \tag{17a}$$

s.t.

$$Ax_c(k) \le b,\ x_c(k+1) = \sum_{i=1}^s w_i(k) \tag{17b}$$

$$M_1x_c(k)+M_2u_c(k)+M_3w(k) \le M_4e(k)+M_5\delta(k)+M_6 \tag{17c}$$

$$g\left(x_l(k+1), x_l(k), u_l(k), e(k), \delta(k)\right) = \mathtt{TRUE}$$
$$w(k) = [w_1(k)\ldots w_s(k)]^T,$$
$$w_i(k) \in \mathbb{R}^{n_c}, \qquad \delta(k) \in \{0,1\}^s \tag{17d}$$

where $\{x_c(k+1), u_c(k), w(k)\}_{k=0}^{T-1}$ are the continuous optimization variables, $\{x_l(k+1), u_l(k), \delta(k), e(k)\}_{k=0}^{T-1}$ are the binary optimization variables, $x_c(0)$, $x_l(0)$ are a given initial state, constraints (17b) and (17c) represent the EG and SAS parts (7a), (7c), (8b), and (9), and the purely continuous or mixed constraints from (5c) and (5d), while (17d) collects the automaton

(1), the MS (4), possible purely Boolean constraints from (5c) and (5d), as well as the exclusive or condition (10). Matrices M_i, $i = 1 \dots 6$, are obtained by the big-M representations (7) and (9).

The alternative formulation using the disjunctive programming relaxations (16) admits an optimal control formulation similar to (17) with $\lambda_i^j(k)$ replacing $w_i(k)$.

Problem (17) belongs to the following general class of *mixed logical/convex* problems:

$$\min_{z,\nu,\mu} f(z,\mu) \tag{18a}$$

$$\begin{aligned}
&\text{s.t.}\\
&g_c(x_c(0),z) \le 0,\ h_c(x_c(0),z) = 0\\
&\text{(Continuous constraints)} && \text{(18b)}\\
&g_m(x_c(0),x_l(0),z,\mu) \le 0,\\
&h_m(x_c(0),x_l(0),z,\mu) = 0 \ \text{(Mixed constraints)} && \text{(18c)}\\
&g_L(x_l(0),\nu,\mu) = \mathtt{TRUE}^{n_{CP}} \text{(Logic constraints)},\\
&z \in \mathbb{R}^{n_z}, \qquad \nu \in \{0,1\}^{n_\nu}, \qquad \mu \in \{0,1\}^{n_\mu} && \text{(18d)}
\end{aligned}$$

where $g_c : \mathbb{R}^{n_z} \to \mathbb{R}^{q_{gc}}$ and $g_m : \mathbb{R}^{n_z+n_\mu} \to \mathbb{R}^{q_{gm}}$ are convex functions, $h_c : \mathbb{R}^{n_z} \to \mathbb{R}^{q_{hc}}$ and $h_m : \mathbb{R}^{n_z+n_\mu} \to \mathbb{R}^{q_{hm}}$ are affine functions, and $g_L : \{0,1\}^{n_\nu \times n_\mu} \to \{0,1\}^{n_{CP}}$ is a Boolean function or its CNF representation. In (18), z collects all the continuous variables ($x_c(k+1)$, $u_c(k)$, $k = 0,\dots,T-1$), the auxiliary variables $w_i(k)$ (or $\lambda_i^j(k)$) needed for expressing the SAS dynamics, possibly slack variables for upper bounding the cost function in (17a) [9], while μ collects the integer variables that appear in mixed constraints ($e(k)$, $\delta_i(k)$, $k = 0,\dots,T-1$, $i = 1,\dots,s$), and ν collects the integer variables such as $x_l(k)$ and $u_l(k)$ that only appear in logic constraints.

C. Reachability Constraints

The formulation of the control optimization problem (17) [or equivalently (18)] can be improved by considering a PWA representation of the hybrid model, and by adding ancillary constraints related to reachability analysis.

Proposition 2: For any $i,h \in \{1,\dots,s\}$, let $P_i = \{\left[\begin{smallmatrix}x\\u\end{smallmatrix}\right] : H_i^x x + H_i^u u \le K_i\}$ and $P_h = \{\left[\begin{smallmatrix}x\\u\end{smallmatrix}\right] : H_h^x x + H_h^u u \le K_h\}$ be polyhedral cells of the PWA dynamics (11). If the following set of linear inequalities:

$$\begin{cases} H_i^x x_0 + H_i^u u_0 \le K_i \\ H_h^x(A^i x_0 + B^i u_0 + f^i) + H_h^u u_1 \le K_h \end{cases} \tag{19}$$

is infeasible, then

$$\begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \in P_i \to \begin{bmatrix} x(k+1) \\ u(k+1) \end{bmatrix} \notin P_h \tag{20}$$

or, equivalently

$$[\delta_i(k) = 1] \to [\delta_h(k+1) = 0] \tag{21}$$

for all $k \in \mathbb{Z}^+$.

Remark 3: Constraints of the form (21) involve only integer variables, and can be, therefore, directly handled in (18d) by the CSP part of the hybrid solver described in Section V.

Remark 4: In principle, one could augment the number of possible reachability constraints (21) by subdividing the cells P_i into smaller subcells. In fact, the larger the number s of cells, the smaller the volume of the cells, and, therefore, the more likely is that transitions between cells are infeasible, i.e., the larger is the number of reachability constraints (20) that can be included in the optimization problem. However, one must keep in mind that the number of integer variables is also s, so there is a tradeoff between the advantage of having more reachability constraints (20) restricting the domain of feasible choices, and the advantage of having a smaller number of binary variables δ_i to determine.

V. Logic-Based Branch&Bound

A. Constraint Satisfaction and Optimization

CSP and optimization are similar enough to make their combination possible, and yet different enough to make it profitable. The two fields evolved more or less independently until a few years ago. However, they have complementary strengths, and the last few years have seen growing efforts to combine them [19]–[21], [39], [40]. CSP, for example, offers a more flexible modeling framework than mathematical programming. It not only permits more succinct models, but the models allow one to exploit the structure of the problem itself to direct the search. CSP relies on logic-based methods such as domain reduction and (Boolean) constraint propagation to accelerate the search for the feasible solution. Conversely, optimization exploits a number of specialized techniques for highly structured problem classes, such as linear programming problems, and converges to optimality by using a wide range of relaxations. These may be based on polyhedral analysis, in which case they take the form of cutting planes, or on the solution of Lagrangean dual methods.

The recent interaction between CSP and optimization promises to affect both fields. In the following sections, we illustrate two approaches for merging them into a single problem-solving technology; in particular, in the first algorithm we will combine convex optimization and satisfiability of Boolean formulas (SAT) and in the second, linear programming with CLP.

1) SAT Problems: An instance of a satisfiability (SAT) problem is a Boolean formula that has three components

- a set of n variables: $x_1, x_2, \dots, x_n$;
- a set of literals; a literal is a variable $(Q = x)$ or a negation of a variable $(Q = \neg x)$;
- a set of m distinct clauses: $C_1, C_2, \dots, C_m$; each clause consists of only literals combined by just logical "*or*" $(\vee)$ connectives.

The goal of the satisfiability problem is to determine whether there exists an assignment of truth values to variables that makes the following conjunctive normal form (CNF) formula satisfiable:

$$C_1 \wedge C_2 \wedge \dots \wedge C_m$$

TABLE I
zCHAFF VERSUS CPLEX ON 3CNF BENCHMARKS. THE REPORTED MEASURED TIMES ARE IN SECONDS

		Satisfiable instances		Unsatisfiable instances	
N. Vars	N. Cons	zCHAFF	CPLEX	zCHAFF	CPLEX
20	91	0	0.036	-	-
50	218	0	0.343	0	0.453
75	325	0	0.203	0	3.671
100	430	0	23.328	0	33.921
125	538	0.016	15.171	0.031	209.766
150	645	0.031	20.625	0.281	4949.58
175	753	0.031	> 1500	0.891	> 5000

where $\wedge$ is the logical "*and*" connective. For a survey on SAT problems and related solvers the reader is referred to [18].

SAT solvers are much more efficient than MIP solvers for solving satisfiability problems. To support this statement, we compared the time spent for solving a feasibility problem with a SAT solver and with an MIP solver on the equivalent mixed-integer formulation, obtained by translating the CNF formula into a set of linear inequalities [10, Sec. IV-A]. In Table I, the state-of-the-art SAT solver zCHAFF [33] is compared with the state-of-the-art commercial mixed-integer linear programming solver CPLEX [41] on a set of uniform random 3CNF benchmarks (taken from http://www.satlib.org), where the 3SAT instances are all in the phase transition region.[1]

2) CLP Problems: An instance of a CLP problem consists of the following elements:

- a set of n integer variables $a_1, a_2, \dots, a_n$ and their corresponding finite domains $D_1, \dots, D_n$;
- a set of linear (in)equalities;
- a set of logic constraints defined by the logic operators *and* ($\wedge$), *or* ($\vee$), *not* ($\neg$), *implication* ($\rightarrow$), and *equivalence* ($\longleftrightarrow$) connecting valid propositions, e.g., $a_1 = 3$ and $a_2 \leq 4$.

The goal of the CLP problem is to find a feasible assignment to variables that satisfies the set of constraints. For a survey on CLP problems and related solvers the reader is referred to [42]. The efficiency of CLP is based on *constraint propagation* and *choice points*. Constraint propagation is an inference rule for finite domain problems that progressively narrows the domains $D_1, \dots, D_n$ of the variables. However, constraint propagation is not a complete solution method. A problem is split into complementary cases, by defining *choice points*, once constraint propagation cannot advance further. By iterating between propagation and choice points the solutions of the problem are finally determined. For more details, we refer the reader to the book [17].

B. A SAT-Based Hybrid Algorithm

The basic ingredients for an integrated approach are 1) a solver for convex problems obtained from relaxations over continuous variables of mixed integer convex programming problems of the form (18a)–(18c), and 2) a SAT solver for testing the satisfiability of Boolean formulas of the form (18d). The relaxed model is used to obtain a solution that satisfies the constraint sets (18b) and (18c) and optimizes the objective function (18a). The optimal solution of the relaxation may fix some of the (0–1) variables to either 0 or 1. If all the (0–1) variables in the relaxed problem have been assigned (0–1) values, the solution of the relaxation is also a feasible solution of the mixed integer problem. More often, however, some of the (0–1) variables have fractional parts, so that further "branching" and solution of further relaxations is necessary. To accelerate the search of feasible solutions one may use the fixed (0–1) variables to "infer" new information on the other (0–1) variables by solving a SAT problem obtained by constraint (18d). In particular, when an integer solution of μ is found from convex programming, an SAT problem then verifies whether this solution can be completed with an assignment of ν that satisfies (18d).

The basic branch&bound (B&B) strategy for solving mixed integer problems can be extended to the present "hybrid" setting where both convex optimization and SAT solvers are used. In a B&B algorithm, the current best integer solution is updated whenever an integer solution with an even better value of the objective function is found. In the hybrid algorithm, an additional SAT problem is solved to ensure that the integer solution obtained for the relaxed problem is feasible for the constraints (18d) and to find an assignment for the other logic variables ν that appear in (18d). It is only in this case that the current best integer solution is updated.

The B&B method requires the solution of a series of convex subproblems obtained by branching on integer variables. Here, the noninteger variable to branch on is chosen by selecting the variable with the largest fractional part (i.e., the one closest to 0.5), and two new convex subproblems are formed with that variable fixed at 0 and at 1, respectively. When an integer feasible solution of the relaxed problem is obtained, a satisfiability problem is solved to complete the solution. The value of the objective function for an integer feasible solution of the whole problem is an upper bound (UB) of the objective function, which may be used to rule out branches where the optimum value attained by the relaxation is larger than the current upper bound.

Let P denote the set of convex and SAT subproblems to be solved. The proposed SAT-based B&B method can be summarized as follows.

1) **Initialization**. $\mathrm{UB} = \infty$ and $P = \{(p^0, \mathrm{SAT}^0)\}$. The convex subproblem p^0 is generated by using (18a)–(18c) along with the relaxation $\mu \in [0,1]^{n_\mu}$, and the SAT subproblem SAT^0 is generated by using (18d).
2) **Node selection**. If $P = \emptyset$ then go to 7); otherwise, select and remove a (p, SAT) problem from the set P; The criterion for selecting a problem is called *node selection rule*.
3) **Logic inference**. Solve problem SAT. If it is infeasible go to step 2).
4) **Convex reasoning**. Solve the convex problem p, and
   a) if the problem is infeasible or the optimal value of the objective function is greater than UB, then go to step 2);
   b) if the solution is not integer feasible, then go to step 6).
5) **Bounding**. Let $\mu^* \in \{0,1\}^{n_\mu}$ be the integer part of the optimal solution found at step 4); to extend this partial

[1]All simulations have been done on a P4 2.8-GHz machine with 1-GB of RAM running CPLEX 9.0 and zCHAFF 2003.12.04, using the Matlab interfaces Cplexmex and Satmex available at http://www.dii.unisi.it/hybrid/tools.html.

solution, solve the SAT problem finding ν such that $g_L(x(0), \nu, \mu^*) = \texttt{TRUE}$. If the SAT problem is feasible then update UB; otherwise, add to the LP problems of the set P the "no-good" cut [19]

$$\sum_{i\in T^*} \mu_i - \sum_{j\in F^*} \mu_j \le B^* - 1$$

where $T^* = \{i|\mu_i^* = 1\}$, $F^* = \{j|\mu_j^* = 0\}$, and $B^* = |T^*|$. Go to step 2).

6) **Branching**. Among all variables that have fractional values, select the one closest to 0.5. Let μ_i be the selected noninteger variable, and generate two subproblems $(p \cup \{\mu_i = 0\}, \text{SAT}\&\{\neg\mu\})$ and $(p \cup \{\mu_i = 1\}, \text{SAT}\&\{\mu\})$, and add them to set P; go to step 2).

7) **Termination**. If $\text{UB} = \infty$, then the problem is infeasible. Otherwise, the optimal solution is the current value UB.

Remark 5: At each node of the search tree, the algorithm executes a three-step procedure: Logic inference, solution of the convex relaxation, and branching. The first step and the attempted completion of the solution do not occur in MIP approaches but they are introduced here by the distinction of mixed (0–1) variables μ and pure (0–1) variables ν. The logic inference and the attempted completion steps do not change the correctness and the termination of the algorithm but they improve the performance of the algorithm because of the efficiency of the SAT solver in finding a feasible integer solution.

Remark 6: The class of problems (18) is similar to the mixed logical/linear programming (MLLP) framework introduced by Hooker in [43]

$$\min\ c^T x \tag{22a}$$
$$\text{s.t.}$$
$$p_j(y,h) \to \left[A^j(x) \ge a^j\right], \qquad j \in J \tag{22b}$$
$$q_i(y,h), \qquad i \in I \tag{22c}$$

where $x \in \mathbb{R}^{n_x}$, $y \in \{0,1\}^{n_y}$, $h \in \mathcal{D} \subset \mathbb{Z}^{n_h}$, (22b) is the continuous part, and (22c) is the logic part. If we consider y as the only discrete variables and a liner cost function, constraints (18b) and (18c) represent the linearization of (22b), and constraints (18d) are equivalent to (22c).

There are, however, a few differences between frameworks (18) and (22). First, the relaxation problem of (18) is the same for each node in the search tree, while in (22) the relaxation depends on which left-hand side of (22b) is true. Second, in the class of problems (18) constraints of type

$$[\mu = 1] \longleftrightarrow [z_1 + z_2 \ge \alpha]$$

cannot be introduced and they have to be converted into inequalities, becoming part of constraints (18c). Inference is done only in the logic part, by the SAT solver, and no information is derived by the continuous part. In the MLLP framework, instead, inferences are made in both ways.

Remark 7: Due to the finite number of (0–1) variables in (18) the number of possible combinations is finite. Moreover, in the B&B strategy, no (0–1) variable is assigned twice in a same path of the tree, so that at each assignment (branch) the number of possible combinations will reduce. Therefore, the algorithm will always terminate in finite time with a solution, if one exists, or reporting infeasibility.

C. A CLP-Based Hybrid Algorithm

The CLP-based hybrid algorithm is very similar to the SAT-based algorithm. In this case, we assume the cost function (18a) is linear [this is for instance the case when the cost terms (6) are used in the optimal control problem formulation, see (23)]. The basic ingredients for an integrated approach of MIP and CLP are 1) a linear program (LP) obtained by relaxing a mixed integer linear programming (MILP) problem and 2) a CLP feasibility problem. The relaxed MILP model is used to obtain a solution that satisfies the constraint sets (18b) and (18c) and optimizes the objective function (18a). The optimal solution of the relaxation may fix some (0–1) variables to an integer value. If all the (0–1) variables in the relaxed problem have been assigned the solution of the relaxation is also a feasible solution for the MILP problem. More often, however, some (0–1) variables are not assigned and, in these cases, further "branching" and solution of further relaxations is necessary. To accelerate the search of feasible solutions, one may use the fixed (0–1) variables to "infer" new information on the other (0–1) variables by applying a constraint propagation phase on constraints (18d): When an integer solution of μ is found, a CLP then verifies whether this solution can be used to find an assignment of ν that satisfies (18d).

In this CLP-based algorithm, an additional CLP problem is solved to ensure that the integer solution obtained for the relaxed MILP problem is feasible for the constraints (18d) and to find an assignment for the other logic variables ν that appear in (18d). It is only in this case that the current best integer solution for the relaxed MILP problem is updated. If it cannot be extended, then the best current integer solution is not updated.

The B&B method requires the solution of a series of LP subproblems obtained by branching on integer variables. The noninteger variable to branch on is chosen by defining a choice point, after assigning integer values which are consistent with the current solution of the MILP. The difference in the various LP subproblems is only in the upper and lower bounds for all integer variables updated by the constraint propagation phase. The value of the objective function for any feasible solution of the problem is a UB of the objective function. After an integer feasible solution of the MILP relaxed problem is obtained, a feasibility problem is solved via CLP to complete the solution.

Let P denote the set of LP subproblems to be solved. The logic-based B&B method can be summarized as follows.

1) **Initialization**. $\text{UB} = \infty$ and $P = \{p^0\}$. The LP subproblem p^0 is generated by using (18a)–(18c) and possibly removing some of the (0–1) variables after a constraint propagation phase.

2) **Select a node**. Select and remove an LP problem p from the set P; if $P = \emptyset$, then go to 5). The criterion for selecting an LP is called *node selection rule*.

3) **Linear reasoning**. Solve the LP problem p, and
   - if the LP is infeasible or the optimal value of the objective function is greater than UB, then go to step 2);
   - if the solution is not integer feasible, then go to step 4);
   - if the solution has integral values for all the integer variables, then we solve the following CLP problem to extend this partial solution: Find ν s.t. $g_L(x(0), \nu, \mu)$. If the CLP problem is feasible, then update UB; otherwise, go to step 2).
4) **Branch on a variable**. Among all variables that have been assigned nonintegral values, select one according to some specified *branching variable selection rule* (e.g., the variable with the largest fractional part). Let μ_i be the selected noninteger variable, and by using choice points define the constraint $\mu_i = 0$. Generate two LP subproblems $p_i^1 = p \cup \{\mu_i = 0\}$ and $p_i^2 = p \cup \{\mu_i = 1\}$, apply constraint propagation to attempt to fix the other (0–1) variables, and add p_i^1 and p_i^2 to set P. Go to step 2).
5) **Termination**. If $\mathrm{UB} = \infty$, then the problem is infeasible. Otherwise, the optimal solution corresponds to the current value UB.

Remark 8: In the above algorithm, there is no cutting strategy since at each node of the search tree infeasible solutions are removed by the constraint propagation phase. Therefore, CLP helps to find infeasible paths in the search tree (paths with the terminal node infeasible) faster than a classical B&B procedure.

Remark 9: The role of constraint propagation is obviously to reduce as much as possible the domain sets of the μ variables that appear in the constraints managed by the CLP solver. In this way, the constraint propagation can reduce the search space removing some branches in the search tree that cannot have feasible solutions. The SAT solver behaves in a similar way to the CLP solver. SAT inference is a feasibility check. If a partial assignment of the μ variables is infeasible for the set of constraints (18d), SAT is able to find the infeasibility easier and more quickly than a CLP solver when dealing with purely Boolean constraints. SAT solvers are also more efficient for finding a feasible assignment for the ν variables with respect to CLP solvers. However, the efficiency of SAT solvers relies upon the representation of the logic part of the problem: While CLP can be used both with logic formulas and linear constraints, as well as global constraints, SAT turns out to be useful only with Boolean constraints.

Remark 10: The same considerations of Remark 7 can be repeated here for showing the termination of the CLP-based algorithm.

VI. Optimal Control Under Reference Uncertainty

In this section, we extend the optimal control approach developed in the previous sections to handle the case of uncertain references. We restrict the analysis to the case of optimal control problems that lead to linear cost functions. To this end, assume the cost function (5a) is expressed as in (6) as a weighted sum of the infinity norms of the state and input tracking errors. Then, as shown in [9] and [11], the optimal control problem can be transformed into an optimization problem with the linear cost function

$$\sum_{k=0}^{N-1} \varepsilon_k^u + \varepsilon_k^x \tag{23a}$$

under the additional constraints

$$\begin{aligned} \mathbf{1}_m \varepsilon_k^u &\geq \pm Q_u\left(u(k) - r_u(k)\right) \\ \mathbf{1}_n \varepsilon_k^x &\geq \pm Q_x\left(x(k+1) - r_x(k+1)\right) \end{aligned} \tag{23b}$$

for all $k = 0, 1, \ldots, N-1$, where $\mathbf{1}_h$ is a column vector of one of length h, and (23b) must be considered in a component-wise sense.

It is a quite typical situation that the references $r_x(k+1)$ and $r_u(k)$ in (5a) are not known exactly, as they may derive from forecasts or extrapolations of past references. We assume here that, while the hybrid dynamics and design/ancillary constraints have no uncertainty, we only know that the references $r_x^j(k+1) \in [\bar{r}_x^j(k+1) - \hat{r}_x^j(k+1), \bar{r}_x^j(k+1) + \hat{r}_x^j(k+1)]$, for $j = 1, \ldots, n$, and $r_u^j(k) \in [\bar{r}_u^j(k) - \hat{r}_u^j(k), \bar{r}_u^j(k) + \hat{r}_u^j(k)]$, for $j = 1, \ldots, m$, and for all $k = 0, \ldots, T-1$. The objective is to minimize the cost function

$$\sum_{k=0}^{T-1} \max_{r_x(k+1)} \left\|Q_x\left(x(k+1) - r_x(k+1)\right)\right\|_\infty + \max_{r_u(k)} \left\|Q_u\left(u(k) - r_u(k)\right)\right\|_\infty \tag{24}$$

with the idea in mind of maintaining the tracking error as small as possible whatever the reference is within the given range.

A. Bertsimas–Sym's Approach to Robust Linear Programming

Consider the following mixed-integer linear optimization problem:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b \\ & l \leq x \leq u \end{aligned} \tag{25}$$

where $x \in \mathbb{R}^{q_r} \times \{0,1\}^{q_l}$, $q = q_r + q_l$, $A \in \mathbb{R}^{t \times q}$, $c \in \mathbb{R}^q$, and $b \in \mathbb{R}^t$. Without loss of generality we assume that data uncertainty only affects the elements of A and c but not of b.[2]

Assume we have reasonable estimates for the mean value of the coefficients a_{ij} and their range $\hat{a}_{ij}$, without a statistical information about the distribution of these coefficients. Similarly, assume we have estimates for the cost coefficients c_j and an estimate of their range d_j. Specifically, the model U of data uncertainty we consider is as follows.

Uncertainty for matrix A.

[2]In fact, in case b it is uncertain [as it would be in linear constraints generated from (23b)]; it is possible to introduce a new variable x_{q+1}, and write $Ax - bx_{q+1} \leq 0, l \leq x \leq u$, and $1 \leq x_{q+1} \leq 1$.

Let $Q = \{1, 2, \ldots, q\}$. Each entry a_{ij}, $j \in Q$ is modeled as independent, symmetric, and bounded random variable (but with unknown distribution) $\tilde{a}_{ij}$, $j \in Q$ that takes values in $[a_{ij} - \hat{a}_{ij}, a_{ij} + \hat{a}_{ij}]$.

Uncertainty for cost vector c.

Each entry c_j, $j \in Q$ takes values in $[c_j, c_j + d_j]$, where d_j represents the deviation from the nominal cost coefficient c_j.

Note that we allow the possibility that $\hat{a}_{ij} = 0$ or $d_j = 0$. Note also that the only assumption that we place on the distribution of the coefficients a_{ij} is that it is symmetric.

For robustness purposes, for every i, we introduce a number $\Gamma_i, i = 0, 1, \ldots, t$ that takes values in the interval $[0, |J_i|]$, where $J_i = \{j | \hat{a}_{ij} > 0\}$, for $i = 1, \ldots, t$, or $J_0 = \{j | d_j > 0\}$, for $i = 0$. Γ_0 is assumed to be integer, while Γ_i, $i = 1, \ldots, t$ is not necessarily an integer. The role of parameter Γ_i in the constraints is to adjust the robustness of the solution against its level of conservatism. Γ_i is called the protection level of the constraint i. The parameter Γ_0 controls the level of robustness in the objective.

Problem (25) can be reformulated as follows [44]:

$$\begin{aligned}
\min \quad & c^T x + \max_{\{S_0 | S_0 \subseteq J_0, |S_0| \leq \Gamma_0\}} \left\{ \sum_{j \in S_0} d_j |x_j| \right\} \\
\text{s.t.} \quad & \sum_j a_{ij} x_j + \max_{\{S_i \cup \{t_i\} | S_i \subseteq J_i, |S_i| \leq \lfloor \Gamma_i \rfloor, t_i \in J_i \setminus S_i\}} \\
& \left\{ \sum_{j \in S_i} \hat{a}_{ij} |x_j| + (\Gamma_i - \lfloor \Gamma_i \rfloor) \hat{a}_{i|t_i|} |x_{t_i}| \right\} \leq b_i \qquad \forall i \\
& l \leq x \leq u
\end{aligned} \tag{26}$$

for which the following theorem holds.

Theorem 1 ([44]): Problem (26) has the equivalent MILP formulation

$$\begin{aligned}
\min \quad & c^T x + z_0 \Gamma_0 + \sum_{j \in J_0} p_0 j \\
\text{s.t.} \quad & \sum_j a_{ij} x_j + z_i \Gamma_i + \sum_{j \in J_i} p_{ij} \leq b_i \qquad \forall i \\
& z_0 + p_0 j \geq d_j y_j \qquad \forall j \in J_0 \\
& z_i + p_{ij} \hat{a}_{ij} y_j \qquad \forall i \neq 0, j \in J_i \\
& p_{ij} \geq 0 \qquad \forall i, j \in J_i \\
& y_j \geq 0 \qquad \forall j \\
& z_i \geq 0 \qquad \forall i \\
& -y_j \leq x_j \leq y_j \qquad \forall j \\
& l_j \leq x_j \leq u_j \qquad \forall j \\
& x_i \in \{0, 1\}, \qquad i = 1, \ldots, q_l.
\end{aligned} \tag{27}$$

While the original problem (25) involves n variables and m constraints, its robust counterpart (26) has $2q + t + l$ variables, where $l = \sum_{i=0}^{m} |J_i|$ is the number of uncertain coefficients, and $2q + t + l$ constraints. All the new variables introduced in (27) are continuous variables.

Theorem 2 ([45]): Let $x^\star$ be an optimal solution of Problem (27). Suppose that the data in matrix A are subject to the model of uncertainty U, the probability that the ith constraint is violated satisfies:

$$\begin{aligned}
& \Pr \left(\sum_j \tilde{a}_{ij} x_j^\star > b_i \right) \\
& \leq B(n, \Gamma_i) \\
& = \frac{1}{2^n} \left\{ (1 - \mu) \sum_{l = \lfloor \nu \rfloor}^{n} \binom{n}{l} + \mu \sum_{l = \lfloor \nu \rfloor + 1}^{n} \binom{n}{l} \right\}
\end{aligned} \tag{28}$$

where $n = |J_i|$, $\nu = \Gamma_i + n/2$, and $\mu = \nu - \lfloor \nu \rfloor$. Moreover, the bound is tight.

Remark 11: The bound in (28) is independent of $x^\star$.

Coming back to the hybrid optimal control problem under uncertain references, it is apparent that the formulation in (27) is easily embedded in the logic-based problem (18). The role of the parameters Γ_i is to adjust the level of conservatism of the optimal solution, as something that has a probability to be the worst case is minimized, rather than the worst-case itself.

The setup of this section can be easily extended to the case of uncertainty in the design constraints imposed on the model.

VII. Application Example

In this section, we test the ideas developed in the previous sections on a finite-horizon optimal management of a supply chain. Supply chain management (SCM) is the planning and execution of supply chain activities, ensuring a coordinated flow within the enterprise and among integrated companies [46], [47]. These activities include the sourcing of raw materials and parts, manufacturing and assembly, warehousing and inventory tracking, order entry and order management, distribution across all channels, and, ultimately, delivery to the customer. The primary objectives of SCM are to reduce supply costs, improve product margins, increase manufacturing throughput, and improve return on investment [48], [49].

A. Problem Description

- There are F factories; P products can be produced in each factory; there are R retailers in which products are sold (assumption: $R > P$).
- $\text{delay}_{r,f}$ represents the delay time for delivering to retailer r from factory f. If no connection exists, then $\text{delay}_{r,f} = \infty$ (in computations, ∞ is replaced by a finite big number). Since we are working in discrete time, we assume $\text{delay}_{r,f}$ to be integer number.
- $q_{p,r}(0)$ is the set of initial levels of product p in the retailer r.
- $\text{out}_{p,f}$ is the quantity of product p the factory f is able to produce at each instant time (we assume it is not a time varying quantity).
- $\text{sell}_p(t)$ is the sell forecast of product p at time t.
- d_p is the deviation from the nominal coefficient $\text{sell}_p(t)$; if $d_p = 0$ there is no deviation.

The following quantities represent the variables of the problem:

- $s_{r,p}(t) \in \{0, 1\}$: A retailer r sells a product p at time t iff $s_{r,p}(t) = 1$ (0, otherwise).

- $\text{prod}_{p,f}(t) \in \{0,1\}$: A product p is produced in the factory f at time t iff $\text{prod}_{p,f}(t) = 1$ (0, otherwise).
- $\text{conn}_{r,f}(t) \in \{0,1\}$: A retailer r is connected to the factory r at time t iff $\text{conn}_{r,f}(t) = 1$ (0, otherwise).
- $q_{p,r}(t)$: The quantity of product p stored in retailer r at time t.
- $sl_{p,r}(t)$: The quantity of product p that must be sold from retailer r at time t.

The following constraints are included in the supply chain problem.

1) Mixed Integer Constraints:

- Discrete-time continuous dynamics of the retailers

$$q_{p,r}(t+1) = q_{p,r}(t) + P_{p,r}(t) - \text{sell}_{p,r}(t) \tag{29}$$

$\forall p = 1, \dots, P$, $\forall r = 1, \dots, R$, and $\forall t = 0, \dots, T-1$, where

$$P_{p,r}(t) = \sum_{f=1}^{F} \text{conn}_{r,f}(t) \cdot \text{prod}_{p,f}(t - \text{delay}_{r,f}) \cdot \text{out}_{p,f}. \tag{30}$$

$P_{p,r}(t)$ represents the amount of product p produced at time t by all the factories committed to the production of p and delivered to retailer r. Obviously, $P_{p,r}(t)$ depends on the delays of the delivers. Equation (30) is nonlinear and we can translate it into a linear one by introducing a new binary variable $\text{channel}_{p,r}(t) = 1$, if there exists a connection between product p and retailer r at time t

$$\text{channel}_{p,r}(t) \longleftrightarrow \text{conn}_{r,f}(t) \wedge \text{prod}_{p,f}(t - \text{delay}_{r,f}) \quad \forall f = 1, \dots, F \tag{31a}$$

$$P_{p,r}(t) = \sum_{f=1}^{F} \text{channel}_{p,r}(t) \cdot \text{out}_{p,f}. \tag{31b}$$

- Each retailer cannot store more than a Q bound of products

$$\sum_{p=1}^{P} q_{p,r}(t) \le Q \tag{32a}$$

$\forall r = 1, \dots, R$ and $\forall t = 0, \dots, T-1$. There is also the natural lower bound

$$q_{p,r}(t) \ge 0 \tag{32b}$$

$\forall p = 1, \dots, P$, $\forall r = 1, \dots, R$, and $\forall t = 0, \dots, T-1$.

- Every factory can produce at most one product at time

$$\sum_{p=1}^{P} \text{prod}_{p,f}(t) \le 1 \tag{33}$$

$\forall f = 1, \dots, F$ and $\forall t = 0, \dots, T-1$.

TABLE II
PROBLEM DATA FOR THE SUPPLY CHAIN PROBLEM

Delays	
1	∞
∞	2
1	2
∞	3

Initial levels			
0	10	0	5
20	0	0	10
0	2	0	0

Production rates		
10	10	10
20	60	90
30	40	20

NEAR table			
1	0	1	0
0	1	0	0
0	0	1	1
0	0	0	1

Black list	
0	0
0	1
1	0
1	1

APP		
1	0	0

Deviations d_p		
10	20	0

Sells $sell_p$		
100	100	100

- At most, Sp products can be sold by a retailer:

$$\sum_{p=1}^{P} s_{r,p}(t) \le Sp \tag{34}$$

$\forall r = 1, \dots, R$ and $\forall t = 0, \dots, T-1$.

- A factory can serve at most $\text{MAX}r$ retailers:

$$\sum_{r=1}^{R} \text{conn}_{r,f}(t) \le \text{MAX}r \tag{35}$$

$\forall f = 1 \dots, F$ and $\forall t = 0, \dots, T-1$.

- A retailer must be served by at least $\text{MIN}f$ factories

$$\sum_{f=1}^{F} \text{conn}_{r,f}(t) \ge \text{MIN}f \tag{36}$$

$\forall r = 1, \dots, R$ and $\forall t = 0, \dots, T-1$.

2) Logic Constraints:

Near retailers.

Near retailers are defined through the table NEAR, an upper triangular matrix such that $\text{NEAR}_{i,j} = 1$ if retailer i is near to retailer j ($\text{NEAR}_{i,i} = 1$ by definition). Near retailers cannot have the same products (this is to spread the products and to avoid competition among retailers):

$$s_{p,i} \wedge \text{NEAR}_{i,j} \to \neg s_{p,j} \tag{37}$$

$\forall p = 1, \dots, P$, $\forall i \ne j$, $j > i$, and $i, j = 1, \dots, R$.

Black list retailers.

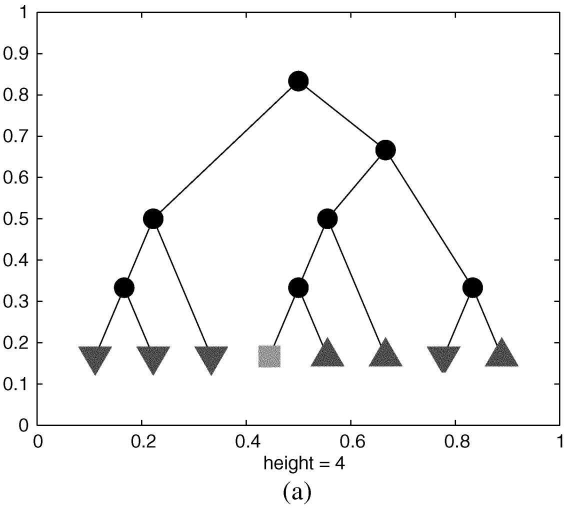
A retailer i is in the black list of factory j if $BL_{i,j} = 1$. A retailer r is in a black list of a factory f if the retailer is not able to sell the products produced in two instant times. In this case the factory stops the connection with the retailer for two instant times:

$$\text{conn}_{r,f}(t) \wedge BL_{f,r} \to (\neg\text{conn}_{r,f}(t+1) \wedge \neg\text{conn}_{r,f}(t+2)). \tag{38}$$

More appreciated products on the market.

TABLE III
OPTIMAL CONTROL SOLUTION: COMPARISON AMONG SAT-BASED B&B, CLP-BASED B&B, CPLEX 9.0 MILP, AND A NAIVE MILP IMPLEMENTATION

T	Bool.	SATbB&B			CLPbB&B		CPLEX		Naive MILP	
	Vars	(s)	LPs	SATs	(s)	LPs	(s)	LPs	(s)	LPs
5	210	0.401	8	10	0.201	5	0.61	74	3.45	193
10	420	1.430	10	16	1.349	8	1.802	216	8.91	312
20	840	7.825	33	40	7.612	17	10.301	632	18.160	748
30	1260	11.510	78	98	13.67	104	23.081	692	114.53	1021
40	1680	79.318	264	304	104.67	579	125.930	934	813.23	1404
100	4200	207.840	2306	2409	253.273	3201	403.020	3657	> 1200	–



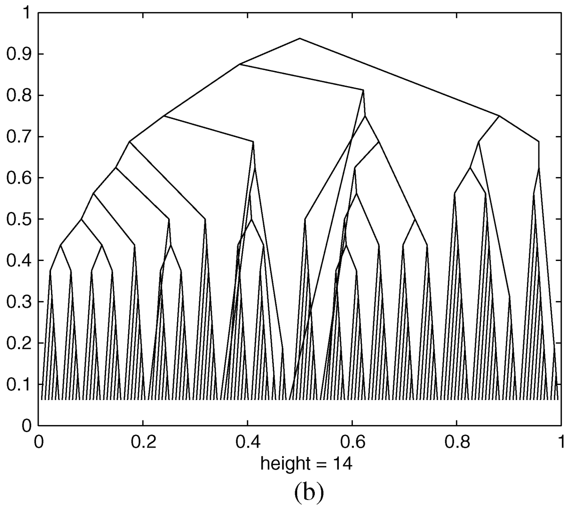


Fig. 2. Comparison of the trees generated by the SAT-based and naive MILP algorithms when solving the problem with $T = 10$. (a) SAT-based algorithm (circle: feasible, square: integer feasible, triangle (down): SAT cut, triangle (up); bound phase). (b) Naive MILP algorithm.

p is an appreciated product from the market if $\mathrm{APP}_p = 1$ (0, otherwise). If product p is an appreciated product by the market, it has to be produced by at least a factory

$$\mathrm{APPR}_p \rightarrow \bigvee_{f=1}^{F} \mathrm{prod}_{p,f}(t). \tag{39}$$

B. Cost Function

We want to track as much as possible the demand forecast on a T-step horizon:

$$\min \sum_{t=0}^{T-1} \sum_{p=1}^{P} \left| \mathrm{sell}_p(t) - \sum_{r=1}^{R} sl_{r,p}(t) \right|. \tag{40}$$

We assume uncertainty on the sell forecasts; namely $\mathrm{sell}_p(t)$ takes value in $[\mathrm{sell}_p(t), \mathrm{sell}_p(t) + d_p]$, where d_p represents the deviation from the nominal coefficient $\mathrm{sell}_p(t)$.

C. Numerical Results

We consider an instance of the problem with two factories, three products, and four retailers and a variable planning horizon T. Problem data are reported in Table II, where time invariant sells are considered ($\mathrm{sell}_p(t) = \mathrm{sell}_p, \forall t = 0, \ldots, T-1$). Each retailer cannot store more than $Q = 200$ unit of products; at most $Sp = 2$ products can be sold by a retailer; a factory can serve at most $\mathrm{MAX}r = 3$ retailers; and a retailer has to be served by at least $\mathrm{MIN}f = 1$ factories.

The supply chain problem can be viewed as a hybrid system in DHA form: (29), (31b) represent the (discrete-time) continuous dynamics of the retailers which are selected at each instant time by (31a) (the MS); the automaton part is described by the logic constraints (37), (38), and (39). All the previous constraints represent the dynamical constraints (5b) in the finite-time optimal control problem (5), whereas constraints (32), (33), (34), (35), and (36) represent the design constraints (5c), and (40) is the cost function (5a).

The SCM problem at hand can be solved by using the logic-based approaches described in Section V. Each part of the supply chain problem is managed by either the SAT/CLP solver or the LP solver: The cost function (40), the constraints (29), (31b), (32), (33), (34), (35), and (36), the additional constraints coming form the robust reformulation are managed by the LP solver, and the logic part (31a), (37), (38), and (39) is managed by the SAT/CLP solver.[3] In all our simulations we adopted depth-first search as the *node selection rule*, to reduce the amount of memory used during the search.

For the initial condition reported in Table II and for different optimization horizons, we solved the optimal control problem stated above. The results are summarized in Table III (the optimal solution is clearly the same both using SAT-based B&B, CLP-based B&B, and MILP).

[3]Our simulations were carried out on a Pentium Mobile 1.2 GHz with 640-Mb RAM, by describing and solving the problem within the Matlab 7.0 environment and by calling zCHAFF 2004.5.13 [33] for SAT, Solver 6.0 [32] for CLP, and CPLEX 9.01 [41] for LP through MEX interfaces.

The performances of SAT-based B&B and CLP-based B&B are always better than the one of the commercial MILP solver of CPLEX. As the logic-based B&B strategy is implemented in Matlab interpreted code, in Table III, we also compare the performance of a "naive MILP" solver implemented in Matlab, that is obtained from the SAT-based B&B code by simply disabling SAT inference. The increase of performance introduced by SAT inference is evident. The main reason is that the SAT/CLP B&B algorithms solve a much smaller number of LPs than the MILP solver. In the SAT-based solver the "cuts" performed, i.e., the infeasible SAT problems, obtained at step 3) of the algorithm are very useful to exclude subtrees containing no integer feasible solution, see Fig. 2. Moreover, the time spent for solving the integer feasibility problem at the root node of the search tree described as SAT problem is much smaller than solving a pure integer feasibility problem (see Table I). In the CLP-based solver the efficiency is obtained through the constraint propagation phase which is applied at each node of the search tree [step 4) of the CLP algorithm]. Constraint propagation removes at each node infeasible values (that is, values which violate some constraints in the CLP problem) and this is equivalent to introduce cuts. Constraint propagation can be considered as a cutting generation procedure and this explains the reduced number of LPs solved by the algorithm. A very interesting comparison is between the SAT-based and CLP-based algorithms. For short horizons, the CLP algorithm results faster than the SAT-based as constraint propagation is more effective on a small number of constraints (as explained by the smaller number of LPs solved with respect to the SAT-based). However, for larger horizons, the SAT algorithm results more effective to remove infeasible solutions. This does not mean that the CLP approach is worst for longer horizons. The main reason is that constraint propagation is more effective on set of linear integer inequalities than of pure logic constraints. In the problem at hand, we do not exploit the information coming from constraints (33)–(36) in the CLP part. This should improve the performance of the CLP-based algorithm but it would require also a specific tuning for the problem in order to understand which constraints make the constraint propagation phase effective.

VIII. Conclusion

In this paper, we have provided a new framework for formulating and solving problems of optimal control of hybrid dynamical systems that takes advantage of the combined use of numerical and symbolic solution techniques. Through computational experiments on a nontrivial optimal supply chain management problem, we have shown the superiority of the approach in terms of computational performance with respect to previous techniques based on mixed-integer optimization. We believe that the proposed approach opens up the use of hybrid system modeling and control methodologies to a large variety of complex application problems where both dynamical/continuous and purely discrete building blocks need to be taken into account in automatic decision making.

References

[1] H. Witsenhausen, "A class of hybrid-state continuous-time dynamic systems," *IEEE Trans. Autom. Control*, vol. 11, no. 2, pp. 161–167, Feb. 1966.

[2] P. Antsaklis, "A brief introduction to the theory and applications of hybrid systems," *Proc. IEEE, Special Issue on Hybrid Systems: Theory and Applications*, vol. 88, no. 7, pp. 879–886, Jul. 2000.

[3] C. Cassandras, D. Pepyne, and Y. Wardi, "Optimal control of a class of hybrid systems," *IEEE Trans. Autom. Control*, vol. 46, no. 3, p. 3981, Mar. 2001, 415.

[4] X. Xu and P. Antsaklis, "An approach to switched systems optimal control based on parameterization of the switching instants," in *Proc. IFAC World Congr.*, Barcelona, Spain, 2002.

[5] B. Lincoln and A. Rantzer, "Optimizing linear system switching," in *Proc. 40th IEEE Conf. Decision and Control*, 2001, pp. 2063–2068.

[6] F. Borrelli, M. Baotic, A. Bemporad, and M. Morari, "Dynamic programming for constrained optimal control of discrete-time linear hybrid systems," *Automatica*, vol. 41, no. 10, pp. 1709–1721, Oct. 2005.

[7] X. Xu and P. Antsaklis, "Results and perspectives on computational methods for optimal control of switched systems," in *Hybrid Systems: Computation and Control*, O. Maler and A. Pnueli, Eds. New York: Springer-Verlag, 2003, pp. 540–555, no. 2623, ser. Lecture Notes in Computer Science.

[8] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, Mar. 1999.

[9] A. Bemporad, F. Borrelli, and M. Morari, "Piecewise linear optimal controllers for hybrid systems," in *Proc. Amer. Control Conf.*, Chicago, IL, Jun. 2000, pp. 1190–1194.

[10] F. Torrisi and A. Bemporad, "HYSDEL—a tool for generating computational hybrid models," *IEEE Trans. Contr. Syst. Technology*, vol. 12, no. 2, pp. 235–249, Mar. 2004.

[11] A. Bemporad, Hybrid Toolbox—User's Guide Dec. 2003 [Online]. Available: http://www.dii.unisi.it/hybrid/toolbox

[12] F. Borrelli, A. Bemporad, M. Fodor, and D. Hrovat, "A hybrid approach to traction control," in *Hybrid Systems: Computation and Control*. New York: Springer-Verlag, 2001, pp. 162–174, ser. Lecture Notes in Computer Science.

[13] A. Bemporad, N. Giorgetti, I. Kolmanovsky, and D. Hrovat, "A hybrid system approach to modeling and optimal control of DISC engines," in *Proc. 41th IEEE Conf. Decision and Control*, 2002, pp. 1582–1587.

[14] R. Möbus, M. Baotic, and M. Morari, , O. Maler and A. Pnueli, Eds., "Multi-objective adaptive cruise control," in *Hybrid Systems: Computation and Control*. New York: Springer-Verlag, 2003, pp. 359–374, ser. Lecture Notes in Computer Science.

[15] G. Ferrari-Trecate, E. Gallestey, P. Letizia, M. Spedicato, M. Morari, and M. Antoine, "Modeling and control of co-generation power plants: a hybrid system approach," in *Hybrid Systems: Computation and Control*, C. J. Tomlin and M. R. Greenstreet, Eds. New York: Springer-Verlag, 2002, vol. 2289, pp. 209–224, ser. Lecture Notes in Computer Science.

[16] T. Geyer, M. Larsson, and M. Morari, "Hybrid emergency voltage control in power systems," in *Proc. European Control Conf.*, Cambridge, U.K., 2003.

[17] K. Marriot and P. Stuckey, *Programming With Constraints: An Introduction*. Cambridge, MA: MIT Press, 1998.

[18] J. Gu, P. Purdom, J. Franco, and B. Wah, "Algorithms for the satisfiability (SAT) problem: a survey," in *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. Providence, RI: AMS, 1997, vol. 35, pp. 19–151.

[19] J. Hooker, *Logic-Based Methods for Optimization*. New York: Wiley, 2000.

[20] A. Bockmayr and T. Kasper, "Branch and infer: a unifying framework for integer and finite domain constraint programming," *INFORMS J. Comput.*, vol. 10, no. 3, pp. 287–300, Summer 1998.

[21] R. Rodosek, M. Wallace, and M. Hajian, "A new approach to integrating mixed integer programming and constraint logic programming," *Ann. Oper. Res.*, vol. 86, pp. 63–87, 1997.

[22] F. Focacci, A. Lodi, and M. Milano, "Cost-based domain filtering," in *Principle and Practice of Constraint Programming*, J. Jaffar, Ed. New York: Springer-Verlag, 2001, vol. 1713, pp. 189–203, ser. Lecture Notes in Computer Science.

[23] I. Harjunkoski, V. Jain, and I. Grossmann, "Hybrid mixedinteger/constraint logic programming strategies for solving scheduling and combinatorial optimization problems," *Comp. Chem. Eng.*, vol. 24, pp. 337–343, 2000.

[24] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[25] A. Bemporad and N. Giorgetti, "A logic-based hybrid solver for optimal control of hybrid systems," in *Proc. 42nd IEEE Conf. Decision and Control*, Maui, HI, Dec. 2003, pp. 640–645.

[26] ——, "A SAT-based hybrid solver for optimal control of hybrid systems," in *Hybrid Systems: Computation and Control*R. Alur and G. Pappas, Eds. New York: Springer-Verlag, 2004, pp. 126–141, no. 2993, ser. Lecture Notes in Computer Science.

[27] ——, "SAT-based branch & bound and optimal control of hybrid dynamical systems," in *Int. Conf. Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimisation Problems (CP-AI-OR)*, J.-C. Regin and M. Rueher, Eds., Nice, France, Apr. 2004, no. 3011, pp. 96–111, ser. Lecture Notes in Computer Science.

[28] E. Asarin, M. Bozga, A. Kerbrat, O. Maler, A. Pnueli, and A. Rasse, "Data-structures for the verification of timed automata," in *Hybrid and RealTime Systems*, O. Maler, Ed. New York: Springer-Verlag, 1997, vol. 1201, pp. 346–360.

[29] R. Alur, C. Courcoubetis, T. Henzinger, and P. Ho, "Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems," in *Hybrid Systems*, R. Grossmann, A. Nerode, and A. Ravn, Eds. New York: Springer-Verlag, 1993, vol. 736, pp. 209–229, ser. Lecture Notes in Computer Science.

[30] E. Sontag, "Nonlinear regulation: the piecewise linear approach," *IEEE Trans. Autom. Control*, vol. 26, no. 2, pp. 346–358, Apr. 1981.

[31] H. Williams, *Model Building in Mathematical Programming*, 3rd ed. New York: Wiley, 1993.

[32] *"Solver 9.0 User Manual,"* ILOG, Inc., Gentilly Cedex, France, 2004.

[33] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: engineering an efficient sat solver," in *39th Design Automation Conf.*, June 2001 [Online]. Available: http://www.ee.princeton.edu/"chaff/zchaff.php

[34] W. Heemels, B. D. Schutter, and A. Bemporad, "Equivalence of hybrid dynamical models," *Automatica*, vol. 37, no. 7, pp. 1085–1091, July 2001.

[35] E. Balas, "Disjunctive programming and a hierarchy of relaxations for discrete optimization problems," *SIAM J. Alg. Disc. Meth.*, vol. 6, pp. 466–486, 1985.

[36] A. Bemporad, "Efficient conversion of mixed logical dynamical systems into an equivalent piecewise affine form," *IEEE Trans. Autom. Control*, vol. 49, no. 5, pp. 832–838, May 2004.

[37] O. Stursberg and S. Panek, "Control of switched hybrid systems based on disjunctive formulations," in *Hybrid Systems: Computation and Control*, C. Tomlin and E. M. R. Greenstreet, Eds. New York: Springer-Verlag, 2002, vol. 2289, pp. 421–435, ser. Lecture Notes in Computer Science.

[38] K. Fukuda*, cdd/cdd+ Reference Manual*. Zurich, Switzerland: Institute for Operations Research, ETH-Zentrum, 0.61 (cdd) 0.75 (cdd+) ed.

[39] G. Ottosson, "Integration of Constraint Programming and Integer Programming for Combinatorial Optimization," Ph.D. dissertation, Comp. Sci. Dept., Inf. Tech., Uppsala Univ., Uppsala, Sweden, 2000.

[40] E. Tsang*, Foundations of Constraint Satisfaction*, . San Diego, CA: Academic, 1993.

[41] *"CPLEX 9.0 User Manual,"* ILOG, Inc., Gentilly Cedex, France, 2004.

[42] J. Jaffar and M. Maher, "Constraint logic programming: a survey," *J. Log. Program.*, vol. 19/20, pp. 503–581, 1994.

[43] J. Hooker and M. Osorio, "Mixed logical/linear programming," *Discrete Appl. Math.*, vol. 96–97, pp. 395–442, 1999.

[44] D. Bertsimas and M. Sym*, Mathematical Programming*. Berlin, Germany: Springer-Verlag, 2003, vol. 98, no. 1, pp. 49–71.

[45] ——, "The price of robustness," *Operation Res.*, vol. 52, no. 1, pp. 35–53, Jan.–Feb. 2004.

[46] H. Min and G. Zhou, "Supply chain modeling: past, present and future," *Comput. Chem. Eng.*, no. 43, pp. 231–249, 2002.

[47] E. Perea, I. Grossmann, E. Ydstie, and T. Tahmanebi, "Dynamic modeling and classical control theory for supply chain management," *Comput. Chem. Eng.*, no. 24, pp. 1143–1149, 2000.

[48] R. Lancioni, "New developments in supply chain management for the millennium," *Ind. Marketing Manage.*, no. 29, pp. 1–6, 2000.

[49] M. W. Braun, D. Rivera, W. Carlyle, and K. Kempf, "Application of model predictive control to robust management of multiechelon demand networks in semiconductor manufacturing," *Simulation*, vol. 79, no. 3, pp. 139–156, Mar. 2003.

Alberto Bemporad (S'95–M'99) received the M.S. degree in electrical engineering, in 1993, and the Ph.D. degree in control engineering, in 1997, from the University of Florence, Florence, Italy.

He spent the academic year 1996–1997 at the Center for Robotics and Automation, Department of Systems Science and Mathematics, Washington University, St. Louis, as a Visiting Researcher. In 1997–1999, he held a Postdoctoral position at the Automatic Control Lab, ETH, Zurich, Switzerland, where he collaborated as a Senior Researcher, in 2000–2002. Since 1999, he has been with the Faculty of Engineering of the University of Siena, Siena, Italy, where he is currently an Associate Professor. He has published several papers in the area of hybrid systems, model predictive control, multiparametric optimization, computational geometry, robotics, and automotive control. He is coauthor of the *Model Predictive Control Toolbox* (The Mathworks, Inc.) and author of the *Hybrid Toolbox for Matlab*.

Dr. Bemporad was an Associate Editor of the IEEE TRANSACTIONS ON AUTOMATIC CONTROL during 2001–2004. He has been the Chair of the Technical Committee on Hybrid Systems of the IEEE Control Systems Society, since 2002.

Nicolò Giorgetti (S'02) was born in Florence, Italy, in 1977. He received the Laurea degree in computer engineering from the University of Florence, Florence, Italy, in 2002. He received the Ph.D. degree in control engineering from the Dipartimento di Ingegneria dell'Informazione, University of Siena, Siena, Italy.

From September 2004 to April 2005, he held a visiting position at the General Robotics, Automation, Sensing and Perception (GRASP) Lab, School of Engineering and Applied Science, University of Pennsylvania, Philadelphia, PA. He is currently with General Motors Powertrain Europe, Turin, Italy. His current research interests are in hybrid dynamical systems, model predictive control (MPC), constraint satisfaction (SAT/CLP) and optimization. His interests include also the application of the hybrid systems theory to automotive case studies.