Performance Driven Reachability Analysis for Optimal Scheduling and Control of Hybrid Systems

A. Bemporad[†*], L. Giovanardi[‡], F.D. Torrisi[†]

[†]Automatic Control Laboratory, Swiss Federal Institute of Technology - bemporad,torrisi@aut.ee.ethz.ch

[‡]Dipartimento di Sistemi e Informatica, Università di Firenze, Italy - giovanardi@dsi.unifi.it

[*]Dipartimento di Ingegneria dell'Informazione, Università di Siena, Italy

Abstract

In this paper we tackle the optimal control problem for piecewise linear and hybrid systems by using a computational approach based on performance-driven reachability analysis. The idea consists of coupling a reach-set exploration algorithm, essentially based on repetitive use of linear programming, to a quadratic programming solver which selectively drives the exploration. In particular, an upper bound on the optimal cost is continually updated during the procedure, and used as a criterion to discern non-optimal evolutions and to prevent their exploration. The result is an efficient strategy of branch-and-bound nature, which is especially attractive for solving long-horizon hybrid optimal control and scheduling problems.

1 Introduction

Hybrid models describe processes whose evolution is guided by both dynamics and logic rules. Apart from being theoretically challenging, hybrid systems are ubiquitous in real-world and industrial applications. A unified framework for their study is now undergoing a fast development, thanks to the interaction between the computer science area and the systems and control engineering community. The main interest has been initially in the field of verification and safety analysis, for which many results and techniques are now available [1]. In contrast, comparatively few contributions propose control synthesis methods.

In the context of continuous-time dynamical models, a general optimal control problem for a broad class of hybrid systems is formulated in [2]. Optimal quadratic control of piecewise linear and hybrid systems is also addressed in [3, 4], where the authors derive bounds on the solution to the associated Hamilton-Jacobi-Bellmann inequalities, which are computable by solving convex optimization problems (linear matrix inequalities [3] or finite-dimensional linear programming [4]).

The inherently hybrid nature of many optimization problems encountered in industrial applications is also revealed by the complementary point of view of *discrete event systems* [5]. In the area of intelligent manufacturing and queuing systems, for example, one frequently faces *scheduling* problems. The goal of scheduling is to accomplish a given set of tasks (also identified as *jobs*) so as to optimize a meaningful performance criterion. Since the jobs to be scheduled usually involve some dynamics, the problem is hybrid. The dynamics taken into account in scheduling problems is generally very simple (often just of integral type, corresponding to timed events [6, 7]). Optimization of hybrid processes through dynamic simulation is also proposed in [8]. Here, the authors use mixed-integer linear programming (MILP) to obtain a candidate switching sequence. A standard scheduling problem is then solved for the fixed sequence. Other approaches based on mixed-integer linear programming (MILP) have been proposed in the chemical process control literature [9], for a survey the reader is referred to [10].

Model predictive control (MPC) of hybrid systems is proposed in [11], where the standard MPC optimization problem is modified to include also discrete components in the state-update and output equations as well as in the constraints. The problem is then solved on-line via mixed-integer quadratic programming (MIQP), and only the first sample of the optimal input sequence is applied to the plant, a new optimization being repeated at each time step. The fundamental limitations of this approach reside in the on-line computational burden, that requires the sampling time T_s to be sufficiently large, and in the related complexity determined by the number of involved 0-1 variables which grows linearly with the time horizon T, and thus practically limits T to small values.

The present paper tackles the optimal control problem for hybrid systems by using reachability analysis, for which an algorithm was proposed in [12]. This algorithm abstracts all possible controlled behaviors of the system into a high-level structure. We show here how the procedure can be suitably adapted for optimization purposes. Notably, if an optimization stage is performed in parallel with reach-set computation, the latter can be selectively carried out according to a convenient strategy that discards evolutions which are recognized not to be optimal, and finally determines the desired optimal input sequence. The procedure exploits linear programming for reach-set computation, and quadratic programming for the optimization stage.

Compared with the approach of [11], where complexity of the involved MIQP exponentially depends on the prediction horizon T, the method proposed in this paper appears particularly attractive for solving hybrid optimal/receding horizon control problems where the prediction horizon T is large and switching is not frequent. In fact, complexity here is mainly related to the number of switches. Therefore, the larger the number of sampling steps between switches (e.g., because of a small sampling time T_s), the more efficient the algorithm is with respect to the use of MIQP solvers as in

[11]. An additional feature of this solution is that it embeds a practical reachability test for the system to be controlled, which is generally a prerequisite.

The paper is organized as follows. In Sect. 2 we introduce the considered class of hybrid systems and formulate the associated optimal control problem, discussing its applications and inherent complexity. Sect. 3 describes the performance-driven reachability analysis strategy, for which a complete algorithm is detailed. Sect. 4 is devoted to an application example. Some concluding comments end the paper in Sect. 5.

2 Problem Formulation

Consider the *piecewise affine* (PWA) discrete-time system described by the equations

$$x(t+1) = A_i x(t) + B_i u(t) + f_i, \quad \text{for } \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \in \mathcal{X}_i \qquad (1)$$

where $\{\mathcal{X}_i\}_{i=0}^{s-1}$ is a partition of the state+input set,

$$\mathcal{X}_i \triangleq \left\{ \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} : \; K_i x + L_i u \leq M_i \right\}$$

and f_i are suitable constant vectors. Each subsystem defined by (A_i,B_i,f_i), $i \in \{0,1,\ldots,s-1\}$ is termed a *component* of the PWA system (1). If the vectors f_i are null, system (1) is referred to as piecewise linear.

In [13] PWA systems are shown to be equivalent to the class of *mixed logical dynamical* (MLD) systems described by the relations

$$x(t+1) = Fx(t) + G_1 u(t) + G_2 \delta(t) + G_3 z(t) \qquad (2a)$$
$$E_2\delta(t) + E_3 z(t) \leq E_1 u(t) + E_4 x(t) + E_5 \qquad (2b)$$

where in general $x \in \mathbb{R}^{n_c} \times \{0,1\}^{n_\ell}$ is a vector of continuous and binary states, $u \in \mathbb{R}^{m_c} \times \{0,1\}^{m_\ell}$ are the inputs, $\delta \in \{0,1\}^{r_\ell}$, $z \in \mathbb{R}^{r_c}$ represent auxiliary binary and continuous variables respectively, which are introduced when transforming logic relations into mixed-integer linear inequalities [11], and Φ, $G_{1\ldots3}$, $E_{1\ldots5}$ are matrices of suitable dimensions. For the moment we suppose $n_\ell = 0$, $m_\ell = 0$. As discussed later, both hypotheses can be removed. In particular, the former is merely introduced to simplify the exposition, while the issue of removing the latter will be discussed in Remark 5.

MLD systems are capable to model a broad class of systems arising in many applications: linear hybrid dynamical systems, hybrid automata, nonlinear dynamic systems where the nonlinearity can be approximated by a piecewise linear function, some classes of discrete event systems, linear systems with constraints, etc. Examples of real-world applications that can be naturally modeled within the MLD framework are reported in [11], and a language called HYSDEL (HYbrid Systems DEscription Language) to translate high-level hybrid descriptions into MLD models has been developed at ETH. Clearly, in view of the equivalence between PWA and MLD models, all the expressiveness of the MLD modeling framework is inherited by the PWA paradigm.

In this paper we are interested in considering the following optimal control problem

$$J_{\text{opt}} = \min_{U_0^{T-1}} \left\{ \|x(T) - x_f\|_P^2 + \sum_{t=0}^{T-1} \|u(t)\|_R^2 + \|x(t) - x_f\|_Q^2 \right\} \qquad (3a)$$

subj. to

$$\begin{cases} x(t+1) = A_{i(t)}x(t) + B_{i(t)}u(t) + f_{i(t)} \;\; \text{for } \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \in \mathcal{X}_{i(t)} \\ u_{\min} \leq u(t) \leq u_{\max}, \; t = 0,1,\ldots,T-1 \\ x_{\min} \leq x(t) \leq x_{\max}, \; t = 1,\ldots,T \\ x(0) = x_0 \\ i(t) \in \{0,\ldots,s-1\} \\ x(T) \in \mathcal{X}_{\text{fin}} \end{cases} \qquad (3b)$$

where T is the prediction horizon, P, Q and R are positive definite weighting matrices, $x(t)$ is the state evolved at time t by applying the input sequence $U_0^{t-1} \triangleq \{u(0), \ldots, u(t-1)\}$ to (1) from the initial state $x(0) = x_0$, $i(t) \in \{0,\ldots,s-1\}$ is the index such that $K_{i(t)}x(t) + L_{i(t)}u(t) \leq M_{i(t)}$ is satisfied, x_f and $\mathcal{X}_{\text{fin}}$ are a reference state and a polyhedral final target set, respectively (for instance, $\mathcal{X}_{\text{fin}}$ can be a satisfactory range around the equilibrium x_f), $u_{\min}$, $u_{\max}$ and $x_{\min}$, $x_{\max}$ are hard bounds on inputs and states, respectively[1]. The sets $\mathcal{U} \triangleq \{u : \; u_{\min} \leq u \leq u_{\max}\}$ and $\mathcal{X}_{\text{safe}} \triangleq \{x : \; x_{\min} \leq x \leq x_{\max}\}$ will be used in the sequel for compactness of notation.

Remark 1 For ease of notation, we have supposed that the weighting matrices are constant with respect to i (=space) and t (=time). Nonetheless, the proposed framework can easily handle the case of region-dependent and/or time-varying weights $R_{i(t)}(t)$, $Q_{i(t)}(t)$. This feature is helpful for instance when the index i reflects different plant operation modes. The same extension can be done for input and state limits $u_{\min}$, $u_{\max}$, $x_{\min}$, $x_{\max}$.

Remark 2 Existing heuristic information about the expected optimal schedule can be easily embedded into (3). In fact, when heuristics are given as space/time "landmarks" to be hit, such requirements can be expressed in (3) as additional constraints of the form $x(\bar{t}) \in \mathcal{X}_{\text{heur}}(\bar{t})$, where $\mathcal{X}_{\text{heur}}(\bar{t})$ represents the landmark area in the state-space to be reached at time $\bar{t}$.

By taking into account the equivalence between PWA and MLD systems (2) mentioned above, the optimization problem (3) can be reformulated as the mixed-integer quadratic programming (MIQP) problem

$$J_{\text{opt}} = \min_{U_0^{T-1}} \left\{ \|x(T) - x_f\|_P^2 + \sum_{t=0}^{T-1} \|u(t)\|_R^2 + \|x(t) - x_f\|_Q^2 \right\} \qquad (4a)$$

subj. to

$$\begin{cases} x(t+1) = Fx(t) + G_1 u(t) + G_2 \delta(t) + G_3 z(t) \\ E_2\delta(t) + E_3 z(t) \leq E_1 u(t) + E_4 x(t) + E_5 \\ u_{\min} \leq u(t) \leq u_{\max}, \; t = 0,1,\ldots,T-1 \\ x_{\min} \leq x(t) \leq x_{\max}, \; t = 1,\ldots,T \\ x(0) = x_0 \\ \delta(t) \in \{0,1\}^{r_\ell} \\ x(T) \in \mathcal{X}_{\text{fin}} \end{cases} \qquad (4b)$$

[1] More in general, we can allow only some components of the inputs or states to be constrained (e.g. $u_{\min}^i = -\infty$). In (3b), constraints relating to unconstrained input and state components are simply removed.

This is indeed the approach pursued in [11]. We stress once again that in the MLD+MIQP formulation (4) the binary variables $\delta(t)$ play the role of the index variables $i(t)$ in (3).

Practical applications of the above optimal control problems can be developed according to two different philosophies. In some cases, an open-loop solution to (3) is sought. This situation resembles a typical scenario for discrete event systems [5], in which optimal input sequences have to be planned. Sometimes a closed-loop implementation is required instead. In this case the proposed optimization strategy can be pursued on-line, combined with a receding horizon philosophy. The result is an optimal control law in feedback form enjoying nice stability properties, provided that suitable terminal state conditions are chosen. In particular, when (3)/(4) are solved on-line within an MPC scheme, under feasibility assumptions the choice of the terminal state constraint $\mathcal{X}_{\rm fin} = \{x_f\}$ guarantees stability of the controlled system, which is proven by using standard Lyapunov analysis techniques [11].

2.1 Complexity Issues

Problem (3) is $\mathcal{NP}$-hard, because of the presence of integer variables $i(t)$ associated with the different regions of the PWA representation, or, equivalently, of binary variables $\delta(t)$ representing logic literals in the MLD formulation (4). Solving the optimal control problem amounts to finding the switching sequence $I_0^{T-1} \triangleq \{i(0), \ldots, i(T-1)\}$ and the control profile U_0^{T-1} that lead $x(0)$ to $\mathcal{X}_{\rm fin}$ and minimize (3a), while fulfilling the constraints (3b). Even in the absence of discrete inputs ($m_\ell = 0$), the number of all possible switching sequences I_0^{T-1} is combinatorial with respect to T, namely s^T, and any enumeration method would be impractical.

A similar complexity issue arises in reachability analysis of hybrid systems. Determining if the target set $\mathcal{X}_{\rm fin}$ can be reached from the initial state $x(0)$ is, in general, an undecidable problem. For discrete-time hybrid systems, the problem becomes decidable when *practical reachability*, or reachability over a finite horizon of T steps, is considered. This problem is investigated in [12] for PWA and MLD systems, where the authors propose an algorithm that, given a set of initial conditions $\mathcal{X}(0)$, a collection of disjoint target sets $\mathcal{Z}_1$, $\mathcal{Z}_2$, ..., $\mathcal{Z}_L$, a bounded set of inputs $\mathcal{U}$, and a time horizon T, determines if $\mathcal{Z}_j$ is reachable from $\mathcal{X}(0)$ within T steps, and computes the input sequence driving any $x_1 \in \mathcal{X}(0)$ to $x_2 \in \mathcal{Z}_j$. The algorithm avoids brute force enumeration, relying heavily on the PWA description for computing set evolutions within the regions $\mathcal{X}_i$, while making a substantial use of MLD descriptions and of the associated branch-and-bound computational tools [14] for detection of guard-line crossing, that is, of switching between different regions.

3 Optimization via Reachability Analysis

In view of the relationship between the reachability analysis problem and the optimal control problem (3a)–(3b), in this section we exploit the ideas of the verification algorithm proposed in [12] in order to solve the optimal control problem stated in Sect. 2. We use reachability analysis to determine admissible switching profiles I_0^{T-1}. On the other hand, during the reachability analysis, set evolutions are selectively propagated in accordance with the value function J. More precisely, set evolutions having an intermediate cost which is greater than a current upper-bound on $J_{\rm opt}$ are not propagated. Here below, we detail the basic ingredients of the algorithm: computation of the reach sets, detection of guard-line crossing, construction of a tree which abstracts the evolution of the system, fathoming criteria for set evolutions. The idea leads to a solver to the optimal control problem which is of branch-and-bound nature, namely a *branch* occurs whenever a switching is detected, and a *bound* on the optimal cost allows instantaneous fathoming of entire non-optimal subtrees.

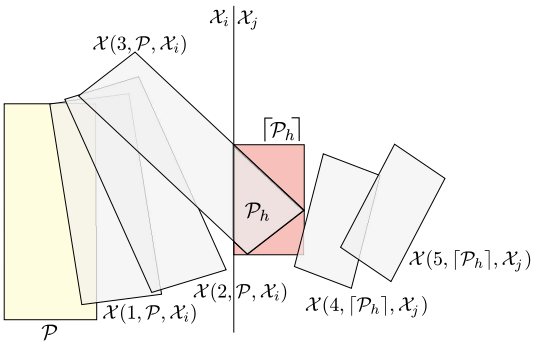

Figure 1: Reach set evolution, guard-line crossing, outer approximation of a new intersection

3.1 Reach Set Computation and Switching Detection

Let $\mathcal{P}$ be a generic set of initial conditions at time t_0 defined by the polyhedral representation $\mathcal{P} \triangleq \{x : S_0 x \leq T_0\}$, and consider the *reach set* $\mathcal{X}(t, \mathcal{P}, \mathcal{X}_i)$, defined as the set of states x which are reachable from $\mathcal{P}$ at time $t > t_0$ by following a path contained in $\mathcal{X}_i$. The subset $\mathcal{S}_i(t_0, t, \mathcal{P})$ of $\mathcal{P}$ whose evolution lies in $\mathcal{X}_i$ for $k \in [t_0, t]$ is a polyhedral set in the augmented space of tuples $(x, u(t_0), \ldots, u(t))$. Therefore, the reach set $\mathcal{X}(t, \mathcal{P}, \mathcal{X}_i)$ is a polyhedral set as well.

Switching detection amounts to finding all possible new regions $\mathcal{X}_h$ entered by the reach set at the next time step, i.e., all nonempty sets $\mathcal{P}_h \triangleq \mathcal{X}(t, \mathcal{P}, \mathcal{X}_i) \bigcap \mathcal{X}_h$, $h \neq i$, as exemplified in Fig. 1. Rather than enumerating and checking for nonemptiness for all $h = 0, \ldots, s-1$, $h \neq i$, we can here exploit the equivalence between PWA systems and MLD models (2), and solve the switching detection problem via branch-and-bound and linear programming. The reader is referred to [12] for details. In the average case, the MLD form requires a number of feasibility tests which is much smaller than enumerating and solving a feasibility test for all the possible regions of the PWA model.

3.2 Tree of Evolution

By exploiting reach-set computation and guard-line crossing detection, we are able to build up a tree (Fig. 2) which abstracts the system behavior over a T-step hori-

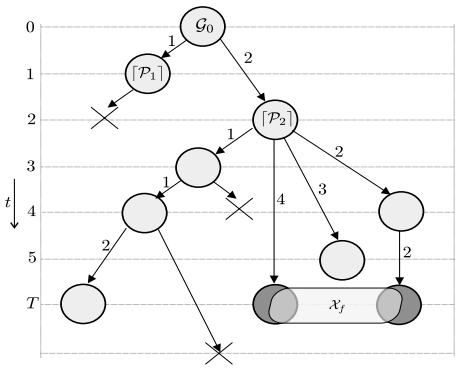

Figure 2: Tree of evolution

zon, for all possible inputs satisfying input saturation and state constraints. The nodes of the tree represent sets from which a reach set evolution is computed, and a branch connects two nodes if a transition exists between the two corresponding sets. Each branch b has an associated weight T_b which represents the time steps needed for the transition. The root node of the tree is the initial set[2] $\mathcal{G}_0 = \{x(0)\}$, from which the reach set evolution is computed according to Sect. 3.1. When a new set $\mathcal{P}_h$ crossing a guard-line is detected, it becomes a new node. The new node is connected by a weighted branch from $\mathcal{G}_0$, and inserted in a list of sets to be subsequently explored. Then, computation of the reach set proceeds in each region $\mathcal{X}_h$ from each new intersection $\mathcal{P}_h$. Each evolution is stopped, or *fathomed*, once one of the following conditions happens:

F1. $t \geq T$, the time horizon limit has been reached.

F2. $\mathcal{X}(t, \mathcal{P}_h, \mathcal{X}_i) \bigcap \mathcal{X}_i = \emptyset$. This means that the whole evolution has left region $\mathcal{X}_i$.

F3. $\mathcal{X}(t, \mathcal{P}_h, \mathcal{X}_i) \bigcap \mathcal{X}_{\text{safe}} = \emptyset$, i.e., all the possible evolutions from $\mathcal{P}_h$ have become unsafe.

As in principle the number of facets of $\mathcal{P}_h$ grows linearly with time (due to both switching and additional degrees of freedom introduced by new input samples), for a practical implementation we need to approximate $\mathcal{P}_h$ so that its complexity is bounded, and therefore reach set computation from $\mathcal{P}_h$ has a limited complexity with respect to the initial region. Hyper-rectangular outer approximations, denoted in Fig. 1 by the symbol $\lceil\mathcal{P}_h\rceil$, are the simplest candidates. Note that, because of the outer approximation of new intersections $\mathcal{P}_h$, not all switching sequences present in the tree are feasible. In [12] a refinement procedure based on simple linear programming feasibility tests is used to identify all feasible switching sequences I_0^{T-1}. In the proposed approach, no additional linear program is needed to accomplish the above mentioned task since feasibility checking is embedded in an intermediate cost computation stage, as will be clarified in the next subsection.

3.3 Partial Cost Computation

We note that each node $\mathcal{G}_k$ of the tree corresponds to a unique switching path from $\mathcal{G}_0$ to $\mathcal{G}_k$ itself. The switching path is associated with a switching sequence $I_0^{T_k-1} = \{i_k(0), \ldots, i_k(T_k-1)\}$ of length $T_k = \sum_b T_b$, where T_b are the time intervals associated with the arcs along the switching path. If we were to build up the whole tree as described before, the leaves at distance T from the initial set (those which were fathomed by condition **F1** in Sect. 3.2) and possessing a nonempty intersection with $\mathcal{X}_{\text{fin}}$ would certainly be the candidates for the solution to the optimal control problem. We could then enumerate all possible winning T-step switching sequences and then compute for each of them the solution J^* to problem (3) with I_0^{T-1} fixed, through standard quadratic programming. This can be merely considered as an extension of the needed refinement step discussed above (as already mentioned, refinement is explicitly embedded since infeasibility of the quadratic program means infeasibility of the corresponding switching sequence).

On the other hand, we are not interested in the full reachability analysis, so we do not need to build up the whole tree. The idea is to associate a cost J_k to each node $\mathcal{G}_k$ by computing the intermediate minimum cost from $\mathcal{G}_0$ to the corresponding region, given by the solution of the following standard quadratic program

$$J_k = \min_{U_0^{T_k-1}} J(U_0^{T_k-1}, 0, T_k) \tag{5a}$$

subj. to

$$\begin{cases} x(t+1) = A_{i_k(t)}x(t) + B_{i_k(t)}u(t) + f_{i_k(t)} \quad \text{for } \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \in \mathcal{X}_{i_k(t)} \\ u_{\min} \leq u(t) \leq u_{\max}, \ t = 0, 1, \ldots, T_k - 1 \\ x_{\min} \leq x(t) \leq x_{\max}, \ t = 1, \ldots, T_k \\ x(T_k) \in \mathcal{G}_k \\ x(0) = x_0 \end{cases} \tag{5b}$$

where $I_0^{T_k-1} = \{i_k(0), \ldots, i_k(T_k-1)\}$ is the corresponding switching sequence, and

$$J(U_{T_i}^{T_f-1}, T_i, T_f) \triangleq \left(\sum_{t=T_i}^{T_f-1} \|u(t)\|_R^2 + \|x(t) - x_f\|_Q^2 \right).$$

When no feasible solution to (5) exists, we conventionally set $J_k = +\infty$. The advantage provided by these additional calculations is that — once the target region is eventually reached in T steps and we can compute an upper bound J^* on the overall cost — we have an additional fathoming condition. More precisely, as soon as an evolution intersects $\mathcal{X}_{\text{fin}}$ at $t = T$, we compute J^* as described before and enforce the following new fathoming condition

F4. $J_k \geq J^*$, the intermediate cost exceeds or is equal to the current upper bound.

Whenever a new exploration reaches $\mathcal{X}_{\text{fin}}$ in T steps with a lower cost, the upper bound J^* is updated, along with the the correspoding minimizer $U^* = \arg\min J^*$. Note again that we are able to rule out possible infeasible switches introduced by the hyper-rectangular approximation $\lceil\mathcal{P}_h\rceil$ as soon as they are detected. In fact, in this case at least one of the intermediate optimal control problems associated to the child nodes will be infeasible.

[2] With a slight abuse of notation, $\mathcal{G}_k$ will be referred to as both a node of the tree and the associated set in the state space.

3.4 Node Selection Criterion

The last point to be addressed is the choice of an effective exploration strategy, that is, the ordering criterion according to which new nodes are taken from the list and explored. In order to reduce fruitless explorations, an adequate strategy should recognize the more promising paths to be searched. To this purpose, we suggest the following node selection criterion

NS. Select the node having the smallest associated *normalized cost* $\hat{J}_k \triangleq J_k/T_k$

where, for convenience, we set $J_0 = \hat{J}_0 = 0$. The normalization factor $1/T_k$ is instrumental to the proposed performance-driving mechanism. In fact, it penalizes, among sequences characterized by identical intermediate cost J_k, those sequences with a smaller minimum cumulated time T_k, which therefore have a larger time-to-go, and are more likely to give rise to a higher overall cost J^*. As a result, the exploration is guided by performance in the sense that the procedure aims at reaching the target set $\mathcal{X}_{\rm fin}$ through the most promising evolutions. This strategy leads to tighter upper bounds J^*, and thus to a more effective fathoming condition **F4**.

3.5 The Optimal Control Algorithm

The following is a complete algorithmic representation of the optimization procedure described in the previous section.

Algorithm 1

```
1   initialize TREE with root node G_0 = {x(0)}
2   initialize LIST with {G_0, J_0 = 0, T_0 = 0}
3   J* ← +∞, U* ← ∅
4   while LIST nonempty do
5       extract node G_k with minimum normalized cost Ĵ_k
        from LIST
6       if the associated cost J_k ≥ J* then go to 26
7       let i ∈ {0,...,s−1} such that G_k ⊆ X_i
8       t ← T_k
9       if X_fin ⋂ X(t, G_k, X_i) ≠ ∅ and t = T then
10          let J_fin ← minimum cost along the path
            G_0 ⇝ G_k ⇝ X_fin
11          if J_fin<J* then let J*←J_fin, U*←argminJ_fin
12      endif ;
13      if X(t, G_k, X_i) ⋂ X_safe = ∅ then go to 26
14      if t ≥ T then go to 26
15      t ← t + 1
16      X(t, G_k, G_i) = A_i X(t−1, G_k, X_i) ⊕ B_i U ⊕ {f_i}
17      for all h ≠ i s.t. P_h ≜ X_h ⋂ X(t, G_k, X_i) ≠ ∅ do
18          let J_h ← minimum cost along the path G_0 ⇝ P_h
19          if J_h < J* then
20              insert ⌈P_h⌉ in TREE and connect G_k to ⌈P_h⌉
                with weight (t−T_k)
21              insert {⌈P_h⌉, J_h, t} in LIST
22          endif
23      endfor
24      X(t, G_k, X_i) ← X(t, G_k, X_i) ⋂ X_i ⋂ X_s
25      if X(t, G_k) ≠ ∅ then go to 9
26  endwhile
27  if J* = +∞ then problem infeasible else let U_opt ← U*
28  end
```

When a feasible input sequence U^* is already known, in step 3 J^* can be initialized accordingly, in order to improve the fathoming condition **F4** already in the early stages of the algorithms. Reach set evolution is performed in step 16 (symbol $\oplus$ donotes convex sum of sets), while switching detection is performed in step 17 by exploiting the MLD formulation (2). As discussed before, in steps 20-21 the hyper-rectangular outer approximation $\lceil \mathcal{P}_h \rceil$ is used, rather than $\mathcal{P}_h$. In step 10, it is conventionally understood that when no feasible solution exists $J_{\rm fin} \triangleq +\infty$. The fathoming conditions **F1**–**F4** are invoked in step 14, 25, 13, 6-19, respectively.

Remark 3 Algorithm 1 is basically a branch and bound algorithm, where branching is associated with the switching of the system, and bounding is given by the fathoming conditions. In particular, conditions **F1**–**F3**) provide a bound for infeasibility, while **F4** a bound related to the cost function. Compared to a branch and bound MIQP solver [14], Algorithm 1 is neither a depth-first nor a breadth-first algorithm, but rather a *best-first* algorithm which exploits the structure of the control problem. The adjective "best-first" stems from the node selection criterion, that aims at exploring first the most promising nodes. Note that the structure of the control problem also determines the way Algorithm 1 computes the lower bounds. In fact, while a standard MIQP solver would obtain lower bounds by relaxing the integrality constraints, Algorithm 1 compute lower bounds by optimizing over reachable subpaths.

Remark 4 Many variations and enhancements to the algorithm are possible, especially with respect to the partial cost computation strategy and the node selection criterion. They are discussed in detail in [15].

Remark 5 When binary inputs $u_j(t) \in \{0,1\}$ are present ($m_\ell \neq 0$), they can be handled in a convenient way by relaxing them during the reachability analysis ($0 \leq u_j \leq 1$), and adding back the integrality constraints only later in the computation of the partial cost (which then becomes an MIQP rather than a QP).

Remark 6 In the particular case where the dynamics of the system is simply linear ($s = 1$), the algorithm executes just one single QP (as only one reach-set computation is performed), in accordance with non-hybrid, conventional finite-horizon linear quadratic solvers.

4 An Example

Even though the worst-case performance of Algorithm 1 is lower with respect to plain enumeration (as typical of branch-and-bound algorithms), the proposed method is considerably faster in the average, as shown by the following example. Consider the unstable system

$$x(t+1) = \begin{cases} \begin{bmatrix} 0.93 & 0.38 \\ -0.12 & 0.96 \end{bmatrix} x(t) + \begin{bmatrix} 0.19 \\ 0.98 \end{bmatrix} & \text{if} \quad x_1 x_2 \geq 0, \\ \begin{bmatrix} 0.97 & 0.09 \\ -0.24 & 0.94 \end{bmatrix} x(t) + \begin{bmatrix} 0.05 \\ 0.97 \end{bmatrix} & \text{if} \quad x_1 x_2 < 0 \end{cases} \tag{6}$$

whose open-loop evolution from the initial condition $x_0 = [1\ 1]'$ is depicted in Fig. 3(a). System (6) is a PWA system defined over the four quadrants, and belongs to the class of unstable PWA systems obtained

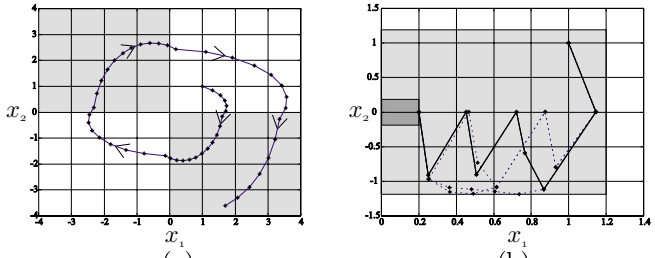

Figure 3: Open-loop and closed-loop trajectories

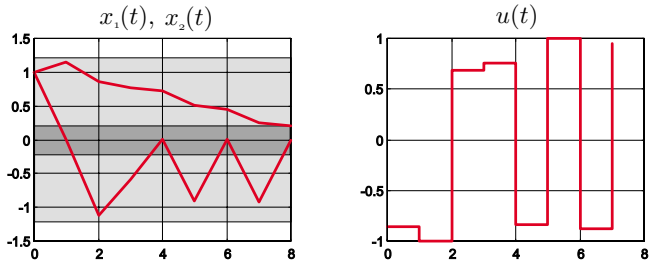

Figure 4: Closed loop state and input evolution

by interconnecting stable linear systems. The goal consists of transferring the state from the initial condition $x_0 = [1\ 1]'$ to $\mathcal{X}_{\text{fin}} = [-0.2, 0.2] \times [-0.2, 0.2]$ in accordance with the following performance objective

$$J_{\text{opt}} = \min_{U_0^7} \left\{ 0.1\|x(8)\|^2 + \sum_{t=0}^{7} |u(t)|^2 + 10\|x(t)\|^2 \right\}$$

$$\begin{cases} -1 \leq u(t) \leq 1 & t = 0, \ldots, 7 \\ -1.2 \leq x_i(t) \leq 1.2 & t = 1, \ldots, 7 \quad i = 1, 2 \\ -0.2 \leq x_i(8) \leq 0.2 & i = 1, 2. \end{cases}$$

If the optimal input sequence $U_{\text{opt}} = \{u(0), \ldots, u(7)\}$ were computed by enumerating all possible region switchings, this would require the solution of $4^8 = 65536$ QPs. By applying Algorithm 1, the optimal solution is found after solving 304 QPs. Fig. 3(b) shows the trajectories obtained by feeding the plant with the optimal input sequence U_{opt} (see Fig. 4). Algorithm 1 is executed in 39.21 s on a Pentium II 300 Mhz running Matlab 5.3.

The same optimal trajectories can be obtained by using the MIQP approach of [11], which is based on the equivalent MLD model of (6) and uses the formulation (4). The MIQP requires 767 QPs in order to determine the optimal solution, which is roughly twice the number of QPs needed by the method proposed in this paper.

5 Conclusions and Acknowledgements

We have presented a branch-and-bound strategy based on reachability analysis for dealing with an NP-hard hybrid optimal control problem. The spirit of the approach can be summarized as follows: "Optimize what is reachable, reach what is optimizing". Main advantages are expected for seldom switching systems (e.g. because of over-sampling). An application example has shown the potentials of the proposed method.

The authors thank the partners of the Esprit Project 26270 for stimulating discussions. This research has been supported by Swiss NSF and Italian MURST.

References

[1] R. Alur, T.A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Trans. on Software Engineering*, 22:181–201, 1996.

[2] M.S. Branicky, V.S. Borkar, and S.K. Mitter. A unified framework for hybrid control: model and optimal control theory. *IEEE Trans. Automatic Control*, 43(1):31–45, 1998.

[3] A. Rantzer and M. Johansson. Piecewise linear quadratic optimal control. In *Proc. American Contr. Conf.*, Albuquerque, 1997.

[4] S. Hedlund and A. Rantzer. Optimal control of hybrid systems. In *Proc. 38th IEEE Conf. on Decision and Control*, pages 3972–3976, Phoenix, AZ, December 1999.

[5] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.

[6] C. G. Cassandras, Q. Liu, and K. Gokbayrak. Optimal control of a two-stage hybrid manufacturing system model. In *Proc. 38th IEEE Conf. on Decision and Control*, pages 450–455, Phoenix, AZ, December 1999.

[7] F. Martinelli. A scheduling problem for n competing queues with finite capacity. In *Proc. 38th IEEE Conf. on Decision and Control*, pages 2276–2281, Phoenix, AZ, December 1999.

[8] C.C. Pantelides, M.P. Avraam, and N. Shah. Optimization of hybrid dynamic processes. In *Proc. American Contr. Conf.*, 2000. Available upon request from the authors.

[9] J.M. Pinto and I.E. Grossmann. A logic-based approach to scheduling problems with resource constraints. *Computers & Chemical Engineering*, 21(8):801–818, 1997.

[10] J.M. Pinto and I.E. Grossmann. Assignment and sequencing models for the scheduling of process systems. *Annals of Operations Research*, 81:433–466, 1998.

[11] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, March 1999.

[12] A. Bemporad, F.D. Torrisi, and M. Morari. Optimization-based verification and stability characterization of piecewise affine and hybrid systems. In B. Krogh and N. Lynch, editors, *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science. Springer Verlag, 2000.

[13] A. Bemporad, G. Ferrari-Trecate, and M. Morari. Observability and controllability of piecewise affine and hybrid systems. *IEEE Trans. Automatic Control*, to appear. `http://control.ethz.ch/`.

[14] C. A. Floudas. *Nonlinear and Mixed-Integer Optimization*. Oxford University Press, 1995.

[15] A. Bemporad, L. Giovanardi, and F.D. Torrisi. Performance driven reachability analysis for optimal scheduling and control of hybrid systems. Technical Report AUT00-15, Automatic Control Laboratory, ETH Zurich, Switzerland, September 2000.