OPTIMIZING HYBRID DYNAMIC PROCESSES BY EMBEDDING GENETIC ALGORITHMS INTO MPC

Thomas Tometzki [*] **Olaf Stursberg** [*,1]
Christian Sonntag [*] **Sebastian Engell** [*]

[*] *Process Control Laboratory (BCI-AST)*
University of Dortmund, 44221 Dortmund, Germany.

Abstract: Optimizing the operation of processes with hybrid dynamics is challenging since the discrete dynamics (i.e. abrupt changes of states or inputs) introduces discontinuities with which gradient-based solvers often cannot cope very well. This contribution suggests a scheme that combines model predictive control (MPC) with genetic algorithms and embedded simulation of the hybrid dynamics. As demonstrated for the example of a chemical reactor, the genetic algorithm provides good results even if the prediction horizon includes points of discontinuities of the continuous dynamics.

Keywords: Automata, genetic algorithms, hybrid systems, optimal control, predictive control.

1. INTRODUCTION

Automated industrial processes are suitably modeled by hybrid dynamic systems if the evolution of continuous quantities (like levels, temperatures, or concentrations) is superposed by switching behavior. The latter arises e.g. from discretely operated actuators or autonomous abrupt changes between qualitatively different dynamics. The consideration of hybrid dynamics is particularly important if the process performs transitions between significantly different operating points as they occur for shutdown, start-up, or product changeover. This contribution proposes a technique to algorithmically compute (near-) optimal control strategies to realize such transition procedures. The starting point is a hybrid model of the plant given as a hybrid automaton with nonlinear continuous dynamics and discrete as well as continuous inputs. An optimization problem is formulated to determine the optimal input trajectories to drive the hybrid automaton from an initial state into a state within a target region such that an appropriate cost criterion is minimized, the hybrid dynamics is considered as a constraint, and unsafe state sets are not reached during the transition. The cost criterion represents the combination of the transition time and costs formulated over the state and the input trajectories.

Our previous work on this task revealed the following problems: If the optimization constraints are approximated by algebraic discrete-time linear models and if the optimization problem is solved iteratively using a model predictive control (MPC) scheme and mixed-integer linear programming (Stursberg and Engell, 2002), the solution performance suffers from large numbers of auxiliary variables required to encode the switching dynamics (Till *et al.*, 2004). If alternatively a graph search algorithm with embedded nonlinear programming (NLP) is used (Stursberg, 2004*a*; Stursberg, 2004*b*), the non-smoothness arising from the switching can lead to a lack of convergence in

[1] Corresponding author: olaf.stursberg@uni-dortmund.de

the NLP step. This paper introduces a method that combines the advantages of the two previous approaches in the following sense: The search for (near-) optimal state and input trajectories is carried out using a moving horizon scheme to benefit from the advantage of linearly increasing complexity with the overall time period required for the transition (if a fixed prediction horizon is used). The optimization in any iteration of the MPC scheme is carried out using a genetic algorithm (GA) with embedded hybrid simulation. Since this approach does not employ the gradients of the cost function and of the constraints, the convergence problems mentioned above for NLP do not occur here.

Apart from the work referenced above, a number of alternative approaches to optimal control of hybrid systems were published in recent years: While e.g. (Branicky *et al.*, 1998; Sussmann, 1999; Shaikh and Caines, 2003) aim at extending the maximum principle and calculus of variations to discontinuities, the approaches in (among others) (Shah and Pantelides, 1996; Buss *et al.*, 2000; Zhang and Cassandras, 2001; Bemporad *et al.*, 2002; Lee and Barton, 2003; Stein *et al.*, 2004) address different issues for efficiently computing optimal controllers for certain subclasses of hybrid systems (mostly piecewise affine systems). To the knowledge of the authors of this paper, only the publications (Wegele *et al.*, 2002) and (Olaru *et al.*, 2004) consider the use of GA for hybrid system optimization. While the first does not propose a specific solution algorithm, the second describes a method that solves the optimization problem by GA within an MPC scheme. In contrast to the approach proposed in this paper (which considers continuous-time and nonlinear continuous dynamics), the method in (Olaru *et al.*, 2004) is restricted, however, to MLD-systems, i.e. discrete-time piecewise affine systems.

2. PREDICTIVE CONTROL OF HYBRID AUTOMATA

The model considered in this contribution is formulated as a *hybrid automaton* according to (Stursberg, 2004*a*). Such a model is suitable to represent the transition procedure mentioned above, as it includes continuous and discrete inputs, and it can express the state-dependent switching between different (possibly unstable) continuous dynamics:

Definition 1. Hybrid Automaton
A hybrid automaton with mixed inputs $HA = (X, U, V, Z, inv, \Theta, g, r, f)$ consists of the following elements:

- the *state vector* x defined on the continuous state space $X \subseteq \mathbb{R}^{n_x}$;
- the *continuous inputs* u defined on the continuous input space $U = [u_1^-, u_1^+] \times \ldots \times [u_{n_u}^-, u_{n_u}^+]$, $u_j^-, u_j^+ \in \mathbb{R}$;
- a finite number n_d of *discrete inputs* $v_j \in \mathbb{R}^{n_v}$ defined on the discrete input space $V = \{v_1, \ldots, v_{n_d}\}$;
- a finite set of *locations* $Z = \{z_1, \ldots, z_{n_z}\}$;
- an *invariant* mapping $inv : Z \to 2^X$ which assigns a polyhedral set $inv(z_j) = \{x \mid \exists\, n_{p_j} \in \mathbb{N}, C_j \in \mathbb{R}^{n_{p_j} \times n_x}, d_j \in \mathbb{R}^{n_{p_j}}, x \in X : C_j \cdot x \leq d_j\}$ to each location $z_j \in Z$;
- the set $\Theta \subseteq Z \times Z$ of *transitions*, denoted by (z_1, z_2) for a transition from $z_1 \in Z$ into $z_2 \in Z$;
- a *guard* mapping $g : \Theta \to 2^X$ that associates a polyhedral set $g((z_1, z_2)) \subseteq X$ with each $(z_1, z_2) \in \Theta$. For each location $z \in Z$, it is required for all pairs of transitions originating from z that the corresponding guard sets are disjoint;
- a *reset function* $r : \Theta \times X \to X$ assigning an updated state $x' \in X$ to each $(z_1, z_2) \in \Theta$ and $x \in g((z_1, z_2))$;
- a *flow function* $f : Z \times X \times U \times V \to \mathbb{R}^{n_x}$ defining a continuous vector field $\dot{x} = f(z, x, u, v)$ for each pair $z \in Z$, $v \in V$.

Let $T = \{t_0, t_1, t_2, \ldots\}$ be an ordered set of time points, such that T contains the initial time t_0 and all points of time at which an input change or a transition occurs. The continuous states x_k, the locations z_k, and the inputs u_k and v_k are defined for the points $t_k \in T$. The values of u_k and v_k are piecewise constant on each interval $[t_k, t_{k+1}[$, $k \in \mathbb{N} \cup \{0\}$. Σ denotes the set of all valid hybrid states $\sigma_k := (z_k, x_k)$ with $z_k \in Z$ and $x_k \in inv(z_k)$. For given sequences of values u_k and v_k, a *feasible run* ϕ_σ of HA is then given as a sequence $\phi_\sigma = (\sigma_0, \sigma_1, \sigma_2, \ldots)$ with $\sigma_k \in \Sigma$ such that:

- σ_0 is initialized to a given $z_0 \in Z$ and $x_0 \in inv(z_0)$ (but x_0 not contained in any guard set).
- σ_{k+1} results from σ_k by: (1) continuous evolution: $\chi : [0, \tau] \to X$ with $\tau = t_{k+1} - t_k$ and $\chi(0) = x_k$, $\chi(\tau) = \int_0^\tau f(z_k, \chi(t), u_k, v_k)dt$ with $\chi(t) \in inv(z_k)$ and for all $t \in [0, \tau[$: $\chi(t) \notin g((z_k, \bullet))$ for any guard set associated with transitions leaving z_k; and (2) a transition $(z_k, z_{k+1}) \in \Theta$ if $\chi(\tau) \in g(z_k, z_{k+1})$, such that $x_{k+1} = r((z_k, z_{k+1}), \chi(\tau)) \in inv(z_{k+1})$, else $z_{k+1} = z_k$, $x_{k+1} = \chi(\tau)$. $\diamondsuit$

The optimal control problem considered in this paper is posed as follows: Assume that an initial state $\sigma_0 \in \Sigma$, a target set $\Sigma_t \subset \Sigma$ with $\Sigma_t = \{(z_t, x) | \exists_1 z_t \in Z : x \in X_t \subset inv(z_t)\}$, as well as a forbidden state set $F = \bigcup_{j=1}^{n_f} F_j \subset \Sigma$ with $F_j = \{(z_{f,j}, x) | \exists_1 z_{f,j} \in Z : x \in X_{f,j} \subset inv(z_{f,j})\}$ are given. ($X_{f,j}$ is polyhedral, and $X_{f,j} \cap X_t =$

$\emptyset$). Assume furthermore that the ordered set of time points $T = \{t_0, t_1, t_2, \ldots, t_f\}$ is finite, and that the inputs can only be changed at $t_k \in T_s \subset T$. Let $\Phi_{u,s}$ contain all possible continuous input trajectories $\phi_u = (u_0, u_1, u_2, \ldots)$ defined on T_s, and $\Phi_{v,s}$ correspondingly all discrete input trajectories $\phi_v = (v_0, v_1, v_2, \ldots)$. Note that ϕ_σ remains defined on T, and that a run of HA according to Def. 1 is deterministic for any choice of ϕ_u and ϕ_v.

The control task is to determine input trajectories ϕ_u^* and ϕ_v^* that lead to a run ϕ_σ^* of HA from σ_0 into Σ_t such that no hybrid state in F is encountered and a cost function Ω is minimized:

$$\min_{\phi_u \in \Phi_{u,s}, \phi_v \in \Phi_{v,s}} \Omega\left(t_f, \phi_\sigma\right) \tag{1}$$

$s.t.$ $\phi_\sigma = (\sigma_0, \ldots, \sigma_f)$ with $\sigma_0 = (z_0, x_0)$, $\sigma_f := (z(t_f), x(t_f)) \in \Sigma_t$, and for ϕ_σ applies in each phase of continuous evolution acc. to Def. 1: $(z_k, \chi(t)) \notin F_j \ \forall\ F_j \in F,\ \forall\ t \in [0, \tau]$.

σ_f denotes the first hybrid state along ϕ_σ which is contained in Σ_t. Since computing the solution of the problem in Eq. 1 requires that $|T_s|$ choices are made for the values of v and u, the solution space of the optimization problem grows exponentially with $|T_s|$. In order to allow for a computationally tractable solution also for larger sets T_s, the approach presented here approximates the problem in Eq. 1 by employing a moving horizon scheme. The optimization problem is divided into $|T_s| - 1$ subproblems which are solved iteratively and yield a solution $\hat{\phi}_u \in \Phi_{u,s}$, $\hat{\phi}_v \in \Phi_{v,s}$ of the control task, which leads to a run $\hat{\phi}_\sigma$ of HA. Starting from t_0, a subproblem is solved for every $t_k \in T_s$ as follows: for a given $h \in \mathbb{N}^{>0}$, an ordered time set $T_k = \{t_k, \ldots, t_{k+h-1}\}$ denotes the time horizon for which the problem:

$$\min_{\overline{\phi}_u \in \Phi_{u,h}, \overline{\phi}_v \in \Phi_{v,h}} \Omega\left(t_k, \ldots, t_{k+h}, \overline{\phi}_\sigma\right), \tag{2}$$

is solved. The input trajectories are defined as $\overline{\phi}_u = \{u_k, \ldots, u_{k+h-1}\}$, $\overline{\phi}_v = \{v_k, \ldots, v_{k+h-1}\}$ and are taken from the sets $\Phi_{u,h}$ and $\Phi_{v,h}$ of all input trajectories of length h. The value of h is either equal to a user-specified parameter, or, if Σ_t is reached within the horizon, equal to the number of time steps required to reach Σ_t. $\overline{\phi}_\sigma = (\sigma_k, \ldots, \sigma_{k+h})$ denotes the corresponding feasible run of HA, and in each iteration k, σ_k is set equal to the hybrid state σ_{k+1} of the run obtained from the optimization in the preceding iteration. The optimization according to Eq. 2 is subject to the same constraint for the forbidden sets as the optimization according to Eq. 1. The cost function Ω depends on the time points $t_k \in T_k$, since the fraction of each time interval $[t_k, t_{k+1}]$, for which the evolution of HA is outside of Σ_t, leads to a cost contribution. A second contribution is determined as the distance between each $\sigma_k \in \overline{\phi}_\sigma$ and Σ_t. (All state variables are normalized to the interval $[0, 1]$ for the distance computation.)

When all subproblems are solved, the complete continuous input trajectory $\hat{\phi}_u$ is obtained from concatenating the first elements u_k of every trajectory $\overline{\phi}_u$. $\hat{\phi}_v$ is constructed accordingly.

3. A GA-BASED OPTIMIZATION ALGORITHM

The subproblems in Eq. 2 are solved employing genetic algorithms (GAs), which were first introduced in (Holland, 1975). GAs provide an efficient means to solve optimization problems, even if discontinuities in the cost function or the constraints are present. Fig. 1 shows the scheme of the algorithm consisting of three modules: the genetic algorithm, an evaluation module, and a module that performs the simulation of the hybrid dynamics of HA. In an iteration k of the MPC scheme, the GA is initialized with the initial hybrid state σ_k and the horizon T_k (see Sec. 2). It performs n_{max} iterations to update the *population* $P_n = \{s_{1,n}, \ldots, s_{\mu,n}\}$, which is the set of *individuals* considered in iteration n, and $\mu \in \mathbb{N}^{>0}$ is an even number which represents the population size. n_{max} is determined as the number of generations after which no increase in the best fitness value (see below) is observed. Each individual s represents a combination of continuous and discrete input trajectories. While continuous inputs are represented using real-valued numbers, the discrete inputs are encoded as bit sequences. These elements, the *genes*, lead to individuals defined by:

$$s_{i,n} := (s_{i,n,1}, s_{i,n,2}, \ldots, s_{i,n,h-1}), \tag{3}$$

$$\text{with} \quad s_{i,n,j} := (a_1, \ldots, a_{n_u}, b_1, \ldots, b_{n,v}). \tag{4}$$

The parameter $a_l \in \mathbb{R}$ with $l \in \{1, \ldots, n_u\}$ represents the value of the continuous input u_l, and a bit sequence $b_p \in \mathbb{B}^{1 \times \lceil log(c_p) \rceil}$, $p \in \{1, \ldots, n_v\}$ encodes the value of the discrete input v_p (where c_p is the number of possible discrete values of v_p). The individuals of the initial population P_1 are generated randomly.

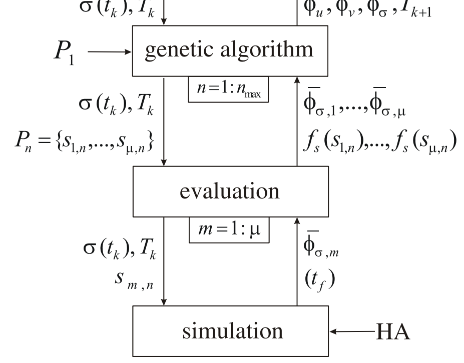
$\sigma(t_k), T_k$
$\overline{\phi}_u, \overline{\phi}_v, \overline{\phi}_\sigma, T_{k+1}$
P_1
genetic algorithm
$n = 1 : n_{max}$
$\sigma(t_k), T_k$
$P_n = \{s_{1,n}, \ldots, s_{\mu,n}\}$
$\overline{\phi}_{\sigma,1}, \ldots, \overline{\phi}_{\sigma,\mu}$
$f_s(s_{1,n}), \ldots, f_s(s_{\mu,n})$
evaluation
$m = 1 : \mu$
$\sigma(t_k), T_k$
$s_{m,n}$
$\overline{\phi}_{\sigma,m}$
(t_f)
simulation
HA

Fig. 1. Scheme of the algorithm.

In every iteration n, the algorithm carries out the following steps: For the input trajectories $\overline{\phi}_{u,i}$, $\overline{\phi}_{v,i}$ encoded by an individual $s_{i,n}$ in P_n, the corresponding state trajectory $\overline{\phi}_{\sigma,i,n}$ is computed. This is accomplished by numerical simulation which approximates the run of HA for the time horizon T_k according to the semantics in Def. 1. The *fitness* of $\overline{\phi}_{\sigma,i,n}$ is then determined by evaluating the function:

$$f_s(s_{i,n}) = \frac{1}{\Omega'(t_k, \ldots, t_{k+h}, \overline{\phi}_{\sigma,i,n})}, \quad i \in \{1, \ldots \mu\},$$

in which Ω' is the cost function from Eq. 2, but with two extensions: (a) an additional term is added which penalizes state trajectories that enter the forbidden hybrid state sets F; (b) the hybrid states in $\overline{\phi}_{\sigma,i,n}$ are weighted over the time horizon T_k. The weighting factors w for the hybrid states are determined according to

$$w(t_{k+o}) = \begin{cases} 1 & , o = h \\ f_w \cdot w(t_{k+o+1}) & , o = 1, \ldots, h-1 \end{cases},$$

with $f_w \in \mathbb{R}^{>0}$.

After the fitness evaluation, the population P_{n+1} for the next iteration of the GA is determined. Fig. 2 shows the algorithm used for generating P_{n+1} from P_n.

```
$P_{n+1} := \emptyset$;
WHILE ($|P_{n+1}| < |P_n|$) DO {
    $p_1 := select(P_n)$;
    $p_2 := select(P_n)$;
    $(c_1, c_2) := crossover(p_1, p_2, w_c)$;
    $c_1 := mutate(c_1)$;
    $c_2 := mutate(c_2)$;
    $P_{n+1} := P_{n+1} \cup \{c_1, c_2\}$;
}
$P_{n+1} := elitist(P_n, P_{n+1})$
```

Fig. 2. Algorithm for generating P_{n+1} from P_n.

First, two individuals p_1 and p_2 (the *parents*) are selected from P_n by the function *select*. This function stochastically chooses an individual using a rank-based selection mechanism. In a large number of experiments in which several selection mechanisms were tested, it was found that this mechanism allows for good optimization results for the application example described in the following section. Assuming that P_n is sorted in ascending order with respect to the fitness of its individuals, the probability that an individual $s_{i,n}$, $i \in \{1, \ldots, \mu\}$, is selected from P_n is computed as:

$$w_s(s_{i,n}) = \frac{pos(s_{i,n})}{\sum_i pos(s_{i,n})},$$

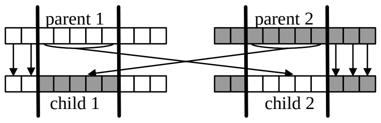

Fig. 3. Two-point crossover.

where the operator *pos* returns the position of $s_{i,n}$ within P_n.

The function *crossover* either performs a two-point crossover of p_1 and p_2 (with probability $w_c \in [0, 1]$) or leaves p_1 and p_2 unchanged (with probability $1 - w_c$). The main idea of the two-point crossover is to generate individuals, the *children*, which combine the advantages of both parents. Two cut-off points of the parents are determined randomly, and the children c_1 and c_2 are constructed by exchanging the segment between the cut-off points (Fig. 3), which may involve continuous and discrete variables. If no crossover is performed, p_1 and p_2 are just copied to c_1 and c_2.

The function *mutate* is used to randomly change genes of the two children. Each gene of a child c_i is changed with probability $w_m = 1/|c_i|$, with the number $|c_i|$ of genes of c_i. Modified real-valued genes a'_j are created from the original gene a_j according to $a'_j = a_j + m_n$; m_n is a realization of a normally distributed random variable with expected value 0 and decreasing variance over the iterations n. Binary-valued genes are changed according to $b'_j = 1 - b_j$.

The procedure of selection, crossover, and mutation is repeated until the number of individuals in P_{n+1} and P_n is equal. Finally, the function *elitist* replaces an arbitrary individual in P_{n+1} with the individual from P_n with the highest fitness value. This function is used to avoid that a good solution for the optimization according to Eq. 2 is lost by applying crossover and mutation.

After n_{max} iterations of the GA, the input trajectories encoded by the individual of $P_{n_{max}}$ with the highest fitness value determine the solution of the optimization in iteration k of the MPC scheme.

The set of time points T_{k+1} for the next iteration of the scheme is computed according to $T_{k+1} = (t_{k+1}, \ldots, t_{k+h-1}, t_{new})$. The last time t_{new} is obtained from $t_{new} = t_{k+h-1} + \Delta t$ with

$$\Delta t = \frac{f_{\Delta t}}{(\|f(z_{k+h}, x_{k+h}, u_{k+h-1}, v_{k-h-1})\|_2)}, \quad (5)$$

where $f_{\Delta t}$ is a tuning factor, $(z_{k+h}, x_{k+h}) = \sigma_{k+h}$ is the last hybrid state of $\overline{\phi}_\sigma$, and u_{k+h-1} and v_{k+h-1} are the last entries of $\overline{\phi}_u$ and $\overline{\phi}_v$. This procedure adjusts the time interval Δt to the dynamics of HA for the current value of the state variables and the inputs.

4. APPLICATION EXAMPLE

The approach is illustrated for the start-up procedure of a chemical reactor as described in (Stursberg, 2004*a*). It consists of a tank equipped with two inlets, a heater, a cooling device, and one outlet.

The state variables considered for optimization are the volume of liquid V_R, the temperature T_R, and the concentrations c_A and c_B of two substances A and B, which react exothermically to form a product. The system has five input variables: the inlet flows F_1, F_2 as well as a variable s_H (representing the status of the heater: *on* or *off*) can be switched between two values. In this example, however, the discrete input set V is restricted to four combinations of values by assuming that the inlet flows F_1 and F_2 are always equal. Hence, the discrete input vector is given by $v := (F_1, F_2, s_H)^{\mathrm{T}}$, $F_1 = F_2$. The continuous inputs of the system comprise the outlet flow F_3 and a cooling flow F_C, thus $u := (F_3, F_C)^{\mathrm{T}}$. According to Def. 1, the state vector is defined as $x := (V_R, T_R, c_A, c_B)^{\mathrm{T}} \in X$. The vector of locations is given by $Z = \{z_1, z_2\}$, with $inv(z_1) = \{x \mid x \in X : V_R \in [0.1, 0.8[\}$ and $inv(z_2) = \{x \mid x \in X : V_R \geq 0.8\}$ to account for the fact that the heating is only effective for a certain range of V_R (i.e. in z_2). The set of transitions is $\Theta = \{(z_1, z_2), (z_2, z_1)\}$ with $g((z_1, z_2)) = g((z_2, z_1)) = \{x \mid x \in X : V_R = 0.8\}$. The flow functions for the two locations are specified as:

$$f^I := f(z_1, x, u, v) =$$
$$\begin{pmatrix} F_1 + F_2 - F_3 \\ (F_1(T_1 - T_R) + F_2(T_2 - T_R))/V_R \\ \quad + F_C k_1 (T_C - T_R)(k_2/V_R + k_3) - k_4 q \\ (F_1 c_{A,1} - c_A(F_1 + F_2))/V_R + k_9 q \\ (F_2 c_{B,2} - c_B(F_1 + F_2))/V_R + k_{10} q \end{pmatrix}, \tag{6}$$

$$f^{II} := f(z_2, x, u, v) =$$
$$\left(f_1^I,\ f_2^I + s_H k_6 (T_H - T_R)(k_7 - \frac{k_8}{V_R}),\ f_3^I,\ f_4^I \right)^{\mathrm{T}}, \tag{7}$$

with $q = c_A c_B^2 \exp(-k_5/T_R)$ and appropriate constants T_1, T_2, T_C, $c_{A,1}$, $c_{B,2}$, T_H and k_1 to k_{10}. State resets do not occur, and the model comprises three forbidden regions $F_1 = (z_1, \{x \mid x \in X : T_R \geq 360\})$, $F_2 = (z_2, \{x \mid x \in X : T_R \geq 360\})$, and $F_3 = (z_2, \{x \mid x \in X : V_R \geq 1.8\})$.

The optimization task is to drive the system from an initial state $\sigma_0 = (z_1, x_0)$ with $x_0 = (0.1, 300, 0, 0)^{\mathrm{T}}$, which corresponds to an almost empty reactor, into an operating region in which the reactor is filled, the temperature has a desired level, and the production rate is sufficiently high, i.e. the concentrations of A and B have low values.

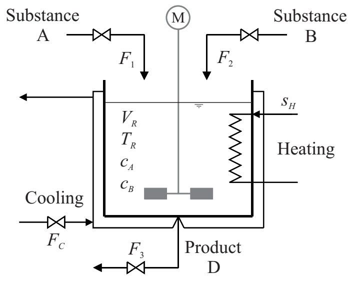

Fig. 4. Scheme of the CSTR.

The target location is z_2, and $X_t \subset inv(z_2)$ is a hyper-box given by $X_t = [1.49, 1.51] \times [343, 347] \times [0.46, 0.5] \times [0.18, 0.22]$.

Several parameters of the algorithm presented in Sec. 3 can be adjusted to tune the performance for the given example. These parameters include the optimization horizon h, the factor determining the weighting of the hybrid states in the fitness computation f_w, the population size μ, the crossover probability w_c, and the tuning factor for the computation of the time steps $f_{\Delta t}$.

In a large number of simulation studies with systematic variation of the values of these parameters, the following effects were observed: The cost function value obtained for the complete transition has a minimum for an optimization horizon of length $h = 4$. The fact that the costs increase for higher values of h may be attributed to the exponential growth of the search space with an increasing value of h, and thus a relatively sparse sampling of the search space for fixed numbers of generations and individuals in the population. Increasing μ or decreasing $f_{\Delta t}$ increases, not surprisingly, the optimization performance (i.e. leads to lower transition costs) but also the computation time. However, since the gain in optimization performance is negligible for higher values of these parameters, it is reasonable to limit them to relatively low values (see Tab. 1) for the sake of a small computation time. Maybe surprisingly, it was found that the crossover probability has only a very small influence on the optimization

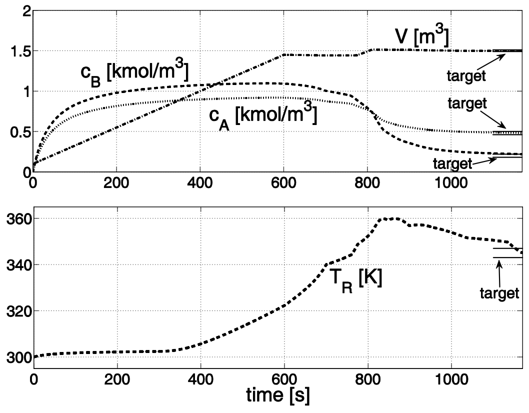

Fig. 5. Optimized state trajectories.

Table 1. An efficient parameterization.

h	f_w	μ	w_c	$f_{\Delta t}$
4	0.2	60	0.4	0.5

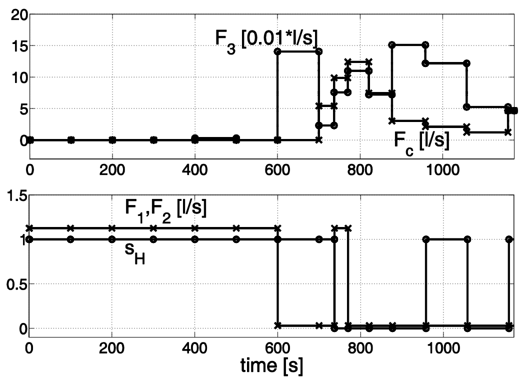

Fig. 6. Optimized input trajectories. F_1 and F_2 are switched between the values 0.03 $\frac{l}{s}$ and 1.125 $\frac{l}{s}$.

performance. The numbers shown in Tab. 1 represent the parameterization that was identified as a suitable compromise between the optimization performance and the computation effort. The figures 5 and 6 depict the trajectories of the state and input variables determined as the optimization result for the parameterization in Tab. 1.

5. CONCLUSION

This paper describes an approach for using GA embedded into an MPC scheme for the optimization of hybrid dynamic models. The solution is qualitatively and quantitatively comparable to the one obtained with the approach in (Stursberg, 2004*a*). In contrast to the latter, the method proposed here has no difficulties to cope with the discrete dynamics of HA. For an extended version of the example, this result has been obtained also for the case that reset functions introduce discontinuities into the state trajectory of the hybrid model (rather than only in the flow functions). By defining an absolute upper limit for the number of generations n_{max} and choosing suitable parameters of the algorithm, the required optimization time per MPC iteration can be held sufficiently small for online application. In this case, the outputs u_k and v_k are applied to the plant in order to obtain the real state (z_{k+1}, x_{k+1}) for the next iteration.

REFERENCES

Bemporad, A., A. Giua and C. Seatzu (2002). A master-slave algorithm for the optimal control switched affine systems. In: 41st *IEEE Conf. on Dec. and Control.* pp. 1976–1981.

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

Buss, M., O. von Stryk, R. Bulirsch and G. Schmidt (2000). Towards hybrid optimal control. *Automatisierungstechnik* **9**, 448–459.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* MIT Press.

Lee, C. K. and P. I. Barton (2003). Determining the optimal mode sequence. In: *IFAC Conf. on Analysis and Design of Hybrid Systems.* pp. 153–158.

Olaru, S., J. Thomas, D. Dumur and J. Buisson (2004). Genetic algorithm based model predictive control for hybrid systems under a modified mld form. *Int. Journal of Hybrid Systems* **4**(1-2), 113–132.

Shah, V.D. Dimitriadis N. and C.C. Pantelides (1996). Optimal design of hybrid controllers for hybrid process systems. In: *Hybrid Systems III.* Vol. 1066 of *LNCS.* Springer. pp. 224–257.

Shaikh, M.S. and P.E. Caines (2003). On the optimal control of hybrid systems. In: *Hybrid Systems: Comp. and Control.* Vol. 2623 of *LNCS.* Springer. pp. 466–481.

Stein, O., J. Oldenburg and W. Marquardt (2004). Continuous reformulations of discrete - continuous optimization problems. *Comp. and Chemical Eng.* **28**(10), 1951–1966.

Stursberg, O. (2004*a*). Dynamic optimization of processing systems with mixed degrees of freedom. In: 7th *Int. IFAC Symp. Dyn. and Control of Process Systems.* number 164.

Stursberg, O. (2004*b*). A graph-search algorithm for optimal control of hybrid systems. In: 43rd *IEEE Conf. on Decision and Control.* pp. 1412–1417.

Stursberg, O. and S. Engell (2002). Optimal control of switched continuous systems using mixed-integer programming. In: 15th *IFAC Congr. on Automatic Control.* Vol. Th-A06-4.

Sussmann, H.J. (1999). A maximum principle for hybrid optimal control problems. In: 38th *IEEE Conf. Dec. and Control.* pp. 425–430.

Till, J., S. Engell, S. Panek and O. Stursberg (2004). Applied hybrid system optimization - an empirical investigation of complexity. *Control Eng. Practice* **12**(10), 1269–1278.

Wegele, S., E. Schnieder and M. Chouikha (2002). Automatic design of controllers for hybrid systems using genetic algorithms. In: *Mod., Analysis, and Design of Hybrid Systems.* Vol. 279 of *LNCIS.* Springer. pp. 285–294.

Zhang, P. and C. Cassandras (2001). An improved forward algorithm for optimal control of a class of hybrid systems. In: 40th *IEEE Conf. Decision and Control.* pp. 1235–1236.