Available online at www.sciencedirect.com
SCIENCE DIRECT®
Control Engineering Practice 12 (2004) 1127–1137
CONTROL ENGINEERING PRACTICE
www.elsevier.com/locate/conengprac

Hybrid modelling and optimal control of a Multiproduct Batch Plant

Boštjan Potočnik[a,*], Alberto Bemporad[b], Fabio Danilo Torrisi[c], Gašper Mušič[a], Borut Zupančič[a]

[a] *Laboratory of Modelling, Simulation and Control, Faculty of Electrical Engineering, University of Ljubljana, Tržaška 25, SI-1000 Ljubljana, Slovenia*
[b] *Dip. Ingegneria dell'Informazione, Università di Siena, Via Roma 56, I-53100 Siena, Italy*
[c] *Automatic Control Laboratory, ETH – Swiss Federal Institute of Technology, CH-8092 Zürich, Switzerland*

Received 14 October 2002; accepted 28 November 2003

Abstract

This paper addresses the problem of optimally selecting the production plan for a Multiproduct Batch Plant. The proposed approach can also be applied to a broader class of optimal control problems for systems with discrete inputs. The plant is modelled as a Discrete Hybrid Automaton (DHA) using the high level modelling language, HYbrid System DEscription Language (HYSDEL), which allows conversion of the DHA model into an Mixed Logical Dynamical (MLD) model. The solution algorithm, which takes into account a model of a hybrid system described as an MLD system, is based on reachability analysis ideas. The algorithm abstracts the behaviour of the hybrid system into a "tree of evolution", where nodes of the tree represent reachable states of the system, and branches connect two nodes if a transition exists between the corresponding states. To each node a cost function value is associated and, based on this value, the tree exploration is driven, searching for the optimal control profile.

Keywords: Hybrid systems; Modelling; Optimal control; Reachability analysis; Branch-and-bound methods

1. Introduction

Hybrid systems are dynamic systems that involve the interaction of continuous dynamics (modelled as differential or difference equations) and discrete dynamics (modelled by finite state machines (FSM)). Hybrid systems have been a topic of intense research activity in recent years, primarily because of their potential importance in applications, e.g. the process industry. Hybrid models are important to a number of problems in system analysis, such as the computation of trajectories, control, stability and safety analysis, etc.

Mathematical models represent the basis of any system analysis and design such as simulation, control, verification, etc. The model should not be too complicated, in order to efficiently define the system behaviour, and not too simple, otherwise it is too far from the behaviour of the real process. We model a hybrid system as a *discrete hybrid automaton* (DHA) using the modelling language HYbrid System DEscription Language (HYSDEL) (Torrisi & Bemporad, 2002). Using an appropriate compiler, a DHA model can be translated to different modelling frameworks, such as *mixed logical dynamical* (MLD), *piecewise affine* (PWA), *linear complementarity* (LC), *extended linear complementarity* (ELC) or *max-min-plus-scaling* (MMPS) systems (Torrisi & Bemporad, 2002; Heemels, De Schutter, & Bemporad, 2001). In this paper the MLD modelling framework presented in Bemporad and Morari (1999) will be adopted, as it is mostly suitable for solving optimal control problems. Moreover, it enables the incorporation of additional working constraints and heuristics that usually appear in industry. Indeed, several control procedures based on the MLD description of a process have been proposed in the literature. A model predictive control technique is presented in Bemporad and Morari (1999) which is able to stabilize an MLD system on a desired reference trajectory, where on-line optimisation procedures are solved through *mixed integer quadratic programming* (MIQP) (Bemporad & Mignone, 2000). A verification approach for hybrid systems is presented in Bemporad, Giovanerdi, and Torrisi (2001).

Optimal control laws for hybrid systems have been widely investigated in recent years, and many results can be found in control engineering literature. Optimal

*Corresponding author. Tel.: +386-1-4768-764; fax: +386-1-4264-631.
E-mail address: bostjan.potocnik@fe.uni-lj.si (B. Potočnik).

0967-0661/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.conengprac.2003.11.010

control of hybrid systems in manufacturing is addressed in Antsaklis (2000), Cassandras, Pepyne, and Wardi (2001), and Gokbayrak and Cassandras (1999), where the authors combine time-driven and event-driven methodologies to solve optimal control problems. An algorithm to optimise switching sequences for a class of switched linear problems is presented in Lincoln and Rantzer (2001), where the algorithm searches for solutions arbitrarily close to the optimal ones. A similar problem is addressed in Barton, Banga, and Galan (2000), where the potential for numerical optimisation procedures to make optimal sequencing decisions in hybrid dynamic systems is explored. A computational approach based on ideas from dynamic programming and convex optimisation is presented in Hedlund and Rantzer (1999). Piecewise linear quadratic optimal control is addressed in Rantzer and Johansson (2000), where the use of piecewise quadratic cost functions is extended from a stability analysis of piecewise linear systems. Optimal control based on reachability analysis and where the inputs of the system are continuous is addressed in Bemporad et al. (2000). A similar idea is here applied to hybrid systems with discrete inputs only. More precisely, in this paper we will address the time optimal control problem of processes with discrete inputs only. The solution to the problem is applied to the Multiproduct Batch Plant example. The solution to the time optimal problem for the example under consideration can be considered as a scheduling problem, as we try to compute the times at which certain decisions are taken. We will, rather, refer to optimal control, as the proposed approach can also be used for the control of hybrid systems where it is difficult to characterise the control problem as a scheduling problem.

The paper is organised as follows. In Section 2 we address the DHA and MLD modelling frameworks. The problem formulation and proposed solution are addressed in Section 3. The proposed algorithm is applied to the Multiproduct Batch Plant and is discussed in Section 4. The conclusions are given in Section 5.

2. DHA and MLD systems

In this section we introduce the modelling framework used for the modelling of a hybrid system. A hybrid system is modelled as a DHA using the HYSDEL modelling language. Using an appropriate compiler, such a model can be translated into a MLD modelling framework that is used by the optimisation algorithm.

DHA are formulated in discrete time and result from the interconnection of a finite state machine (FSM), which provides the discrete part of the hybrid system, with a *switched affine system* (SAS), which provides the continuous part of the hybrid dynamics. The interaction between the two is based on two connecting elements: the *event generator* (EG), which extracts logic signals from the continuous part, and the *mode selector* (MS), which defines the mode (continuous dynamics) of the SAS based on logic variables (states, inputs and events) Torrisi and Bemporad, 2002. At this point we have to stress that we are dealing with a special case where the system includes only discrete inputs, i.e. continuous inputs are not present. Note that we will introduce below a modified DHA system based on this fact. The modified DHA system is shown in Fig. 1.

A SAS without continuous inputs represents a sampled continuous system that is described by the following set of linear affine equations:

$$\begin{aligned}\mathbf{x}_r(k+1) &= \mathbf{A}_{i(k)}\mathbf{x}_r(k) + \mathbf{f}_{i(k)},\\ \mathbf{y}_r(k) &= \mathbf{C}_{i(k)}\mathbf{x}_r(k) + \mathbf{g}_{i(k)},\end{aligned} \tag{1}$$

where $k \in \mathbb{Z}_{\geqslant 0}$ represents the independent variable (time step) ($\mathbb{Z}_{\geqslant 0} \triangleq \{0, 1, \ldots\}$ is a set of non-negative integers), $\mathbf{x}_r \in \mathscr{X}_r \subseteq \mathbb{R}^{n_r}$ is the continuous state vector, $\mathbf{y}_r \in \mathscr{Y}_r \subseteq \mathbb{R}^{p_r}$ is the continuous output vector, $\{\mathbf{A}_i, \mathbf{f}_i, \mathbf{C}_i, \mathbf{g}_i\}_{i \in \mathscr{I}}$ is a set of matrices of suitable dimensions, and $\mathscr{I}$ is a set of variables that select the linear state update dynamics.

An EG generates a logic signal according to the satisfaction of linear affine constraints $\boldsymbol{\delta}_e(k) = f_H(\mathbf{x}_r(k), k)$, where $f_H : \mathbb{R}^{n_r} \times \mathbb{Z}_{\geqslant 0} \to \mathscr{D} \subseteq \{0,1\}^{n_e}$ is a vector of descriptive functions of a linear hyperplane. The relation f_H for time events is modelled as $[\delta_e^i(k) = 1] \leftrightarrow [kT_s \geqslant t_i]$, where T_s is the sampling time, while for threshold events it is modelled as $[\delta_e^i(k) = 1] \leftrightarrow [a_i^T \mathbf{x}_r(k) \leqslant c_i]$, and where a_i and c_i represent the parameters of a linear hyperplane. δ_e^i denotes the ith component of a vector $\boldsymbol{\delta}_e(k)$.

A FSM is a discrete dynamic process that evolves according to a logic state update function $\mathbf{x}_b(k+1) = f_B(\mathbf{x}_b(k), \mathbf{u}_b(k), \boldsymbol{\delta}_e(k))$, where $\mathbf{x}_b \in \mathscr{X}_b \subseteq \{0,1\}^{n_b}$ is the Boolean state, $\mathbf{u}_b \in \mathscr{U}_b \subseteq \{0,1\}^{m_b}$ is the Boolean input, $\boldsymbol{\delta}_e(k)$ is the input coming from the EG, and $f_B : \mathscr{X}_b \times \mathscr{U}_b \times \mathscr{D} \to \mathscr{X}_b$ is a deterministic logic function. A FSM may also have associated Boolean output $\mathbf{y}_b(k) = g_B(\mathbf{x}_b(k), \mathbf{u}_b(k), \boldsymbol{\delta}_e(k))$, where $\mathbf{y}_b \in \mathscr{Y}_b \subseteq \{0,1\}^{p_b}$.

A MS selects the dynamic mode $i(k)$ of the SAS using Boolean function $f_M : \mathscr{X}_b \times \mathscr{U}_b \times \mathscr{D} \to \mathscr{I}$ while considering the Boolean states $\mathbf{x}_b(k)$, the Boolean inputs $\mathbf{u}_b(k)$ and the events $\boldsymbol{\delta}_e(k)$. The output of this function $i(k) = f_M(\mathbf{x}_b(k), \mathbf{u}_b(k), \boldsymbol{\delta}_e(k))$ is called the *active mode*.

DHA models can be built by using the HYSDEL modelling language, which was designed particularly for this class of systems. The HYSDEL modelling language allows the description of hybrid dynamics in textual form. Using an associated compiler this form can be translated into MLD form (Bemporad & Morari, 1999). For a more detailed description of the syntax and the functionality of the HYSDEL modelling language and the associated compiler (HYSDEL tool) the reader is

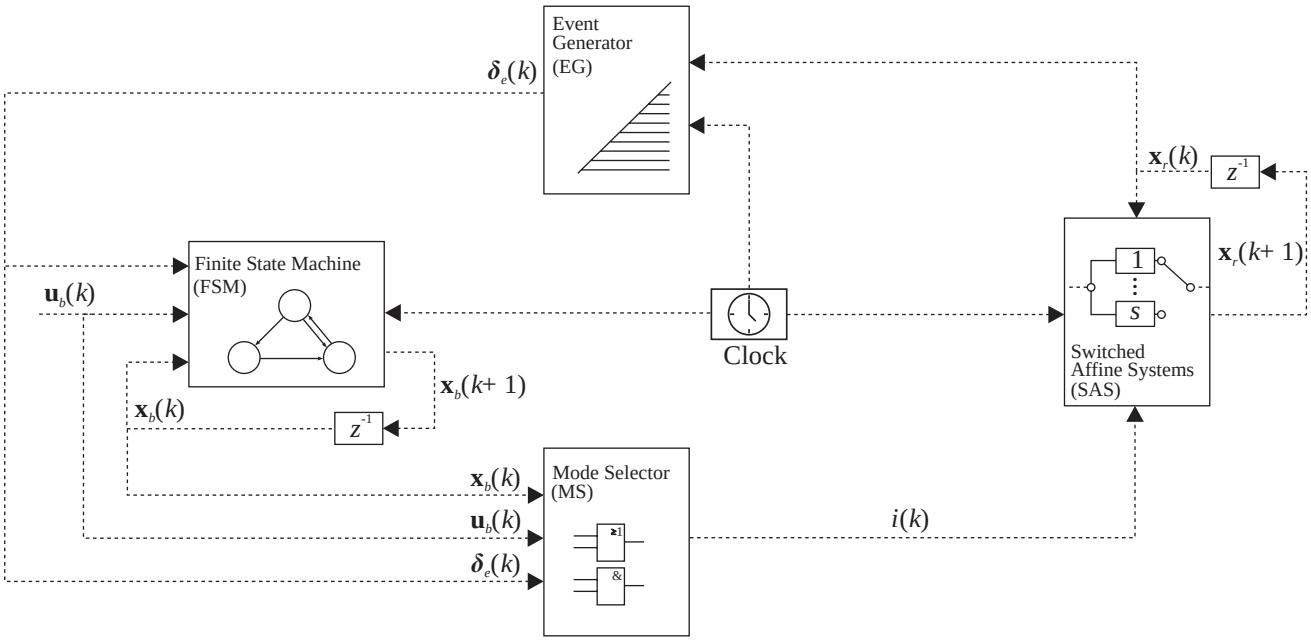

Fig. 1. A DHA without continuous inputs.

referred to Torrisi and Bemporad (2002) and Torrisi et al. (2002).

Once a DHA system is modelled by the HYSDEL modelling language, the companion HYSDEL compiler generates the equivalent MLD model of the form (2). The transformation of a DHA into an equivalent MLD form is presented in Torrisi and Bemporad (2002) and will not be reported here due to space limitations. An MLD system, restricted to discrete inputs only, is described by the following relations:

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}_1\mathbf{u}_b(k) + \mathbf{B}_2\boldsymbol{\delta}(k) + \mathbf{B}_3\mathbf{z}(k), \tag{2a}$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}_1\mathbf{u}_b(k) + \mathbf{D}_2\boldsymbol{\delta}(k) + \mathbf{D}_3\mathbf{z}(k), \tag{2b}$$

$$\mathbf{E}_2\boldsymbol{\delta}(k) + \mathbf{E}_3\mathbf{z}(k) \leqslant \mathbf{E}_1\mathbf{u}_b(k) + \mathbf{E}_4\mathbf{x}(k) + \mathbf{E}_5, \tag{2c}$$

where $\mathbf{x} = [\mathbf{x}_r, \mathbf{x}_b]' \in \mathbb{R}^{n_r} \times \{0,1\}^{n_b}$ is a vector of continuous and logic states, $\mathbf{u}_b \in \{0,1\}^{m_b}$ are the logic (discrete) inputs, $\mathbf{y} = [\mathbf{y}_r, \mathbf{y}_b]' \in \mathbb{R}^{p_r} \times \{0,1\}^{p_b}$ the outputs, $\boldsymbol{\delta} \in \{0,1\}^{r_b}, \mathbf{z} \in \mathbb{R}^{r_r}$ auxiliary logic and continuous variables, respectively, and $\mathbf{A}$, $\mathbf{B}_1$, $\mathbf{B}_2$, $\mathbf{B}_3$, $\mathbf{C}$, $\mathbf{D}_1$, $\mathbf{D}_2$, $\mathbf{D}_3$, $\mathbf{E}_1, \ldots, \mathbf{E}_5$ are matrices of suitable dimensions. Inequalities (2c) can also contain additional constraints on the variables (states, inputs and auxiliary variables). This permits the inclusion of additional constraints and the incorporation of heuristic rules into the model.

Using the current state $\mathbf{x}(k)$ and input $\mathbf{u}_b(k)$, the time evolution of (2) is determined by solving $\boldsymbol{\delta}(k)$ and $\mathbf{z}(k)$ from (2c), and then updating $\mathbf{x}(k+1)$ and $\mathbf{y}(k)$ from Eqs. (2a) and (2b). The MLD system (2) is assumed to be completely well-posed if, for a given state $\mathbf{x}(k)$ and input $\mathbf{u}_b(k)$, inequalities (2c) have a unique solution for $\boldsymbol{\delta}(k)$ and $\mathbf{z}(k)$. A simple algorithm to test well-posedness is given in Bemporad and Morari (1999).

3. A class of optimal control problems

Optimal control amounts to finding the control sequence $U_0^{k_{\text{fin}}-1} = \{\mathbf{u}_b(0), \ldots, \mathbf{u}_b(k_{\text{fin}}-1)\}$ which transfers the initial state $\mathbf{x}_0$ to the final state $\mathbf{x}_{\text{fin}}$ in a finite time $T = k_{\text{fin}} \cdot T_s$ (T_s is the sampling time) while minimising a certain performance index. In this paper we will tackle a class of time optimal control problems where the system can be influenced through discrete inputs only, i.e. $\mathbf{u}_b(k) \in \{0,1\}^{m_b}$. A hybrid system will be modelled as an MLD system due to its compact and powerful description. An optimal control problem will be solved by extending ideas described in Bemporad et al. (2000), where the optimal control of hybrid systems with continuous inputs using reachability analysis is proposed.

3.1. Complexity of the problem

The solution to the posed optimal control problem is the optimal control sequence $U_0^{k_{\text{fin}}-1} = \{\mathbf{u}_b(0), \ldots, \mathbf{u}_b(k), \ldots, \mathbf{u}_b(k_{\text{fin}}-1)\}$, where $\mathbf{u}_b(k)$ represents the discrete input to the system at step k. Due to the fact that we are dealing with a system with m_b discrete inputs and no continuous inputs ($\mathbf{u}_b(k) \in \{0,1\}^{m_b}$ and $U_0^{k_{\text{fin}}-1} \in \{0,1\}^{m_b \cdot k_{\text{fin}}}$), there are $2^{m_b \cdot k_{\text{fin}}}$ possible combinations for $U_0^{k_{\text{fin}}-1}$. Hence the optimisation problem is NP-hard and the computational time required to solve the problem grows exponentially with the problem size

(a^n, for $a > 1$), so that any enumeration method would be impractical.

3.2. Optimisation based on reachability analysis

In general, not all the combinations of inputs are feasible, due to the constraints (2c). One approach to rule out infeasible inputs is to use reachability analysis. The idea for hybrid systems with continuous inputs presented in Bemporad et al. (2000) is adapted here to hybrid systems with discrete inputs.

Using reachability analysis it is possible to determine the admissible control sequences $U_0^{k_{\text{fin}}-1}$. As many of them will be far away from the optimal one, it is reasonable to selectively propagate the search for admissible control sequences. More precisely, the propagation is driven in accordance with the value of a cost function J that evaluates the efficiency of the propagation. We will introduce below the basic parts of the algorithm, i.e. reachability analysis, the construction of a tree of evolution, cost selection criteria and node selection criteria. The whole procedure is a kind of *branch and bound* strategy, namely by searching for reachable states we branch the evolution tree, and by removing non-optimal ones we bound it.

3.2.1. Reachability analysis

Let $\mathbf{x}(k)$ be the state at step k. Reachability analysis computes all the reachable states $\mathbf{x}^j(k+1)$ at the next step considering all possible discrete inputs $\mathbf{u}_b(k)$ to the system and where $j \in \{1, 2, \ldots\}$ is an index marking reachable states. If a system has m_b discrete inputs, then 2^{m_b} possible next states may exist. However, because of the operating constraints that are usually contained in a system, only a smaller number of states can actually be reached. The reachable states $\mathbf{x}^j(k+1)$ are computed by applying the state $\mathbf{x}(k)$ and all possible inputs $\mathbf{u}_b(k)$ at step k to the MLD model (2) of a hybrid system. Reachable (feasible) states are actually defined by the inequalities (2c).

3.2.2. Tree of evolution

By exploiting the reachability analysis technique, we are able to abstract the possible evolution of the system over a horizon of maximum $k_{\max}$ steps into a "tree of evolution" (Bemporad et al., 2000) as shown in Fig. 2. The nodes of the tree represent reachable states, and branches connect two nodes if a transition exists between corresponding states. Each branch has an associated discrete input applied to the system causing the transition. For a given root node $\mathscr{V}_0$, representing the initial state $\mathbf{x}_0$, reachable states are computed and inserted into the tree as nodes $\mathscr{V}_i$, while the corresponding discrete inputs $u_b^i(k)$ are associated with the corresponding branches connecting two nodes. $i \in \{0, 1, \ldots\}$ represents the successive index of the nodes

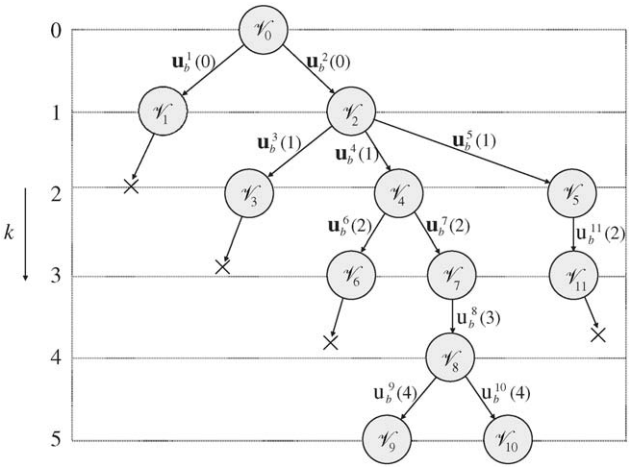

Fig. 2. Tree of evolution.

and branches inserted into the tree. To each new node $\mathscr{V}_i$ a cost value J_i is associated. The search for the optimal control sequence is propagated from a new starting node, whose selection is based on the associated cost value J_i. As soon as a new starting node is selected, new reachable states are computed. The construction of the "tree of evolution" proceeds according to a depth first strategy until one of the following conditions occurs:

- the horizon limit of $k_{\max}$ steps has been reached
- the final state has been reached ($\mathbf{x}(k) = \mathbf{x}_{\text{fin}}$)
- the value of the cost function at the current node is greater than the current optimal one ($J_i \geqslant J_{\text{opt}}$, where initially $J_{\text{opt}} = \infty$).

A node that satisfies one of the above conditions is labelled as explored. If a node satisfies the first or second condition, the associated value of the cost function J_i becomes the current optimal one ($J_{\text{opt}} = J_i$), the corresponding step k indicates the number of steps leading to the current $J_{\text{opt}}(k_{\text{opt}} = k)$ and the control sequence $U_0^{k_{\text{opt}}-1}$ which leads from the initial node $\mathscr{V}_0$ to the current node $\mathscr{V}_i$ becomes the current optimiser. The exploration continues until there are no more unexplored nodes in the tree and the temporary control sequence $U_0^{k_{\text{opt}}-1}$ becomes the optimal one.

3.2.3. Cost function and node selection criterion

The selection of the cost function and the node selection criterion exert great influence on the search propagation of the "tree of evolution" and, indirectly, on the size and consequently on the time efficiency of the optimisation algorithm. The best node selection criterion is to propagate the search in a direction that minimises the value of the cost function, as this means better performance. At the same time, the cost value J_i associated with a node $\mathscr{V}_i$ is used to detect nodes that are not going to lead to the optimal solution, which

prevents an unnecessary growth of the "tree of evolution". To achieve that, the cost function must have certain properties.

As the goal is to reach the final state $\mathbf{x}_{\text{fin}}$ as soon as possible, we chose the following cost function:

$$J_i(\mathbf{x}, k) = h(\mathbf{x}) + g(k), \tag{3}$$

where $h(\mathbf{x})$ presents a "distance measure" to the final state $\mathbf{x}_{\text{fin}}$, with the following properties:

$$h(\mathbf{x}_{\text{fin}}) = 0, \tag{4a}$$

$$h(\mathbf{x}(k+1)) - h(\mathbf{x}(k)) \leqslant 0, \tag{4b}$$

while $g(k)$ is a function that gives a "measure" of elapsed time from the start, with the following property:

$$g(k+1) - g(k) > 0. \tag{5}$$

As mentioned in Section 3.2, the cost function value J is used to drive the propagation of the tree. Therefore, it is reasonable to detect nodes $\mathscr{V}_i$ which do not lead to the optimal solution at step instance $k < k_{\text{opt}}$ (k_{opt} is the time instance of the optimiser) by comparing $J_i(k)$ to $J_{\text{opt}}(k_{\text{opt}})$. To be sure that by continuing the exploration from this node no better solution than the current one can be found, the cost function (3) has to be monotonically increasing, i.e. in the next steps the cost value J_i can only increase. To this end, we impose

$$J(\mathbf{x}(k+1), k+1) - J(\mathbf{x}(k), k) \geqslant 0, \tag{6a}$$

i.e.

$$(h(\mathbf{x}(k+1)) + g(k+1)) - (h(\mathbf{x}(k)) + g(k)) \geqslant 0. \tag{6b}$$

Reaching the final state $\mathbf{x}_{\text{fin}}$ can be detected using cost function (3). Due to Eq. (4a), it can be easily noticed that the cost value at final state $\mathbf{x}_{\text{fin}}$ reached at step k_{fin} is

$$J_{\text{fin}} = g(k_{\text{fin}}). \tag{7}$$

4. A case study: Multiproduct Batch Plant

The proposed approach was applied to the model of a Multiproduct Batch Plant, designed and built at the Process Control Laboratory of the University of Dortmund (Bauer, 2000; Bauer, Kowalewski, Sand, and Löhl, 2000). The demonstration plant is relatively simple compared to industrial-scale plants, but poses complex control tasks.

4.1. Description of the plant

The process under consideration is a batch process that produces two liquid substances, one blue, one green, from three liquid raw materials. The first is coloured yellow, the second red and the third is colourless. The colourless Sodium hydroxide (NaOH) will be addressed by white below.

Table 1
Colours of raw materials and corresponding products

	Raw material	Product
Indicator 1	Yellow	Blue
Indicator 2	Red	Green

The chemical reaction behind the change of colours is the neutralisation of diluted hydrochloric acid (HCl) with diluted NaOH. The diluted HCl acid is mixed with two different pH indicators to make the acid look yellow if it is mixed with the first one and red when mixed with the second one. During the neutralisation reaction, pH indicators change their colour when the pH value reaches approximately 7. The first indicator changes from yellow to blue, and the second from red to green (see Table 1).

The plant consists of three different layers, which can be seen in Fig. 3.

- *The upper layer* consists of the buffering tanks B11, B12 and B13, which are used for holding the raw materials "Yellow", "Red" and "White", respectively. Each of the buffer tanks is used exclusively for one raw material and can hold two batches of substance.
- *The middle layer* consists of three stirred tank reactors R21, R22 and R23. Each reactor can be filled from any raw material buffer tank. This means that each reactor can produce either "Blue" or "Green". The production is done by first filling the reactor with one batch of "Yellow" or "Red" and then neutralising it with one batch of "White". Each reactor can contain one batch of product resulting from two batches of raw material.
- *The lower layer* consists of two buffer tanks B31 and B32 in which the products are collected from the middle layer. Each of them is used exclusively for "Blue" or "Green" and can contain three batches of product.

The processing times of the plant (data of the optimal control problem), presented in Table 2, are based on the following specifications:

- The sampling time is $T_s = 1$ s.
- One batch of raw material is 850 ml (one batch of product is therefore 1700 ml).

Neutralisation takes place while the NaOH is drained into the reactor and does not need any additional time.

To summarise, the system can be influenced through 20 inputs, pumps P1-P5 and valves V111-V113, V121-V123, V131-V133, V211-V212, V221-V222 (see Fig. 3). Given that no additional time is needed to finish the neutralisation, it is natural to empty reactors R21-R23 as soon the reactor is full. Similarly, the same can be

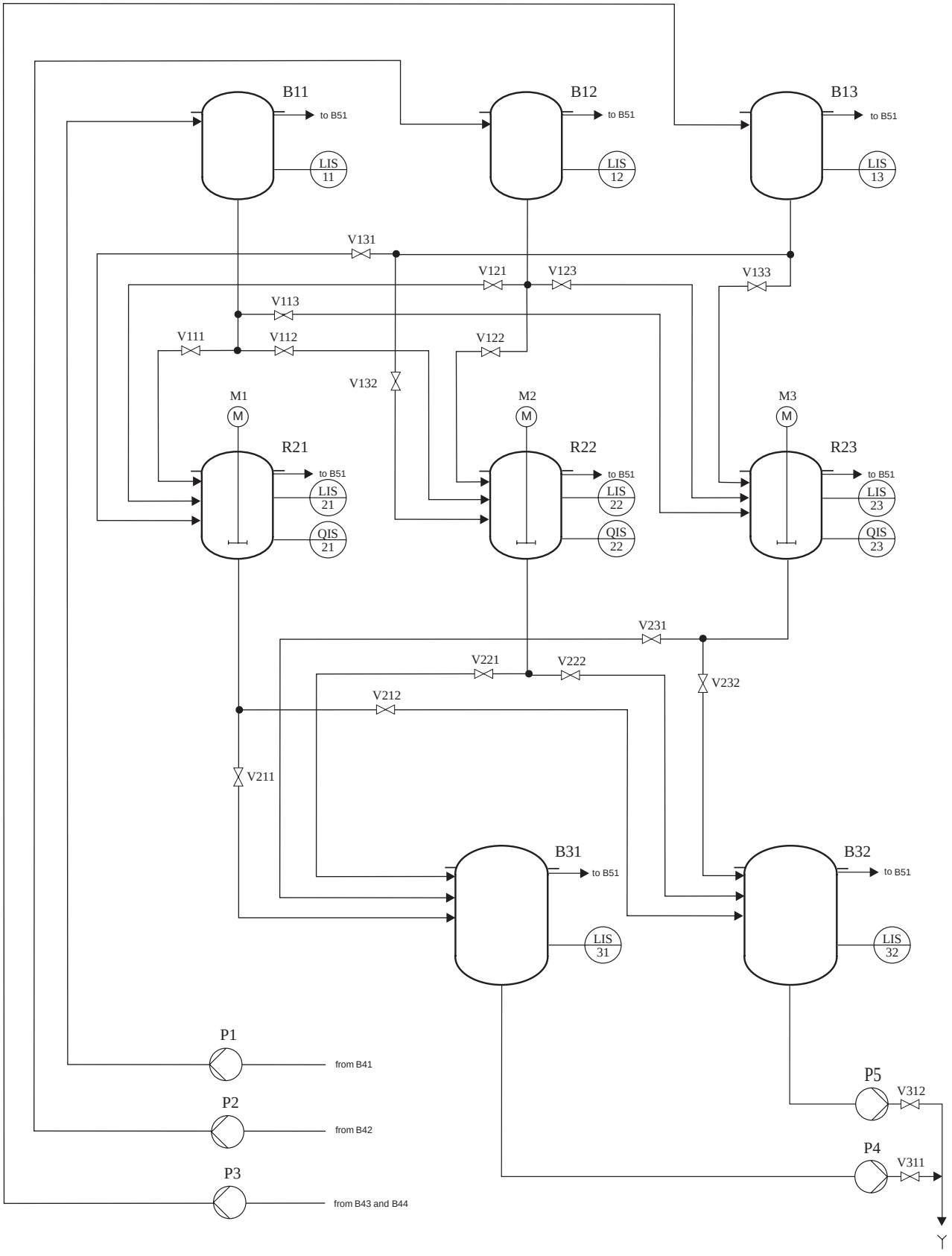

Fig. 3. Multibatch process.

Table 2
Processing times

Processes	Times (s)
Pumping 1 batch "Yellow" into B11	12
Pumping 1 batch "Red" into B12	12
Pumping 1 batch "White" into B13	12
Draining 1 batch "Yellow" into R21	15
Draining 1 batch "Red" into R21	11
Draining 1 batch "White" into R21	10
Draining 1 batch "Yellow" into R22	12
Draining 1 batch "Red" into R22	13
Draining 1 batch "White" into R22	9
Draining 1 batch "Yellow" into R23	12
Draining 1 batch "Red" into R23	14
Draining 1 batch "White" into R23	13
Draining 1 batch "Blue" from R21 into B31	12
Draining 1 batch "Green" from R21 into B32	13
Draining 1 batch "Blue" from R22 into B31	12
Draining 1 batch "Green" from R22 into B32	12
Draining 1 batch "Blue" from R23 into B31	12
Draining 1 batch "Green" from R23 into B32	12
Pumping 3 batches "Red" out of B31	30
Pumping 3 batches "Green" out of B32	30

also considered for buffer tanks B31 and B32. It is obvious that the number of inputs can be reduced to 12, i.e. three pumps (P1-P3) and nine valves (V111-V113, V121-V123, V131-V133) (see Fig. 3).

4.2. DHA and MLD model of a Multiproduct Batch Plant

The Multiproduct Batch Plant was modelled as a DHA system. Due to the extensiveness of the resulting DHA model for the Multiproduct Batch Plant, we will provide only the model for the reactor R21 (see Fig. 3).

Reactor R21 can be filled from buffer tanks B11-B13 and emptied into buffer tanks B31 and B32. The mathematical model of the dynamics for reactor R21, considering all inputs and outputs, can be described as

$$\frac{\mathrm{d}V_{R21}}{\mathrm{d}t} = \Phi_{V_{B11}} u_{b_{V111}} + \Phi_{V_{B12}} u_{b_{V121}} + \Phi_{V_{B13}} u_{b_{V131}} - \Phi_{V_{B31}} u_{b_{V211}} - \Phi_{V_{B32}} u_{b_{V212}}, \quad (8)$$

where V_{R21} represents the volume in reactor R21, $\Phi_{V_{B11}}, \Phi_{V_{B12}}$ and $\Phi_{V_{B13}}$ represent the volume inflows from buffer tanks in the first layer, while $\Phi_{V_{B31}}$ and $\Phi_{V_{B32}}$ represent the volume outflows to the buffer tanks in the third layer. $u_{b_{V111}}, u_{b_{V121}}, u_{b_{V131}}, u_{b_{V211}}, u_{b_{V212}}$ are discrete inputs opening the corresponding valves. The inflow $\Phi_{V_{B11}}$ can be modelled as a nonlinear relation

$$\Phi_{V_{B11}} = K_{B11}\sqrt{2gh_{B11}}, \quad (9)$$

where K_{B11} represents the appropriate constant parameter. As we are actually interested only in the upper bound of the finish times needed to fill or empty the reactor, the Eq. (9) can be simplified into

$$\Phi_{V_{B11}} = L_{B11}, \quad (10)$$

where is constant $L_{B11} = V_{\max_{B11}}/T_{\max_{B11}}$ and is defined by the volume of the batch and the maximum time $T_{\max_{B11}}$ needed to empty one batch of raw material into the reactor R21 (see Table 2). Once the input $u_{b_{V111}}$ is set to 1, it is controlled to have a form (Fig. 4) by using the reset timer. Of course, all remaining volume inflows and outflows are modelled accordingly. Given that the DHA system is given in a discrete time and that the sample time $T_s = 1$ s, we obtain the following difference equation for reactor R21:

$$V_{R21}(k+1) = V_{R21}(k) + L_{B11}u_{b_{V111}}(k) + L_{B12}u_{b_{V121}}(k) + L_{B13}u_{b_{V131}}(k) - L_{B31}u_{b_{V211}}(k) - L_{B32}u_{b_{V212}}(k). \quad (11)$$

The DHA system for reactor R21 is as follows. As, at most, one input can be active at a time, the SAS of the form (1) contains six different operating modes each describing one possible input combination. For the active input $u_{b_{V111}} = 1$ is

$$\begin{bmatrix} V_{R21}(k+1) \\ T_{B11}(k+1) \\ T_{B12}(k+1) \\ T_{B13}(k+1) \\ T_{B31}(k+1) \\ T_{B32}(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} V_{R21}(k) \\ T_{B11}(k) \\ T_{B12}(k) \\ T_{B13}(k) \\ T_{B31}(k) \\ T_{B32}(k) \end{bmatrix} + \begin{bmatrix} L_{B11} \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \text{if } i(k) = 1, \quad (12)$$

where T_{B11}, T_{B12}, T_{B13}, T_{B31}, T_{B32} represent the dynamics of the counters (timers). Of course, for $u_{b_{V111}} = 1$ only counter T_{B11} is active. Inactive mode and reaching the instance $T_{\max_{B11}}$ of the counter T_{B11} is

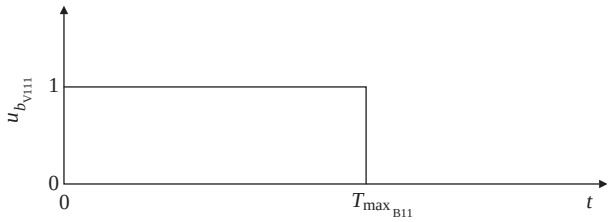

Fig. 4. The form of the input signal.

detected through the function of the EG:

$$[\delta_{1_{B11}} = 1] \leftrightarrow [T_{B11} \leqslant 0],$$
$$[\delta_{2_{B11}} = 1] \leftrightarrow [T_{B11} \geqslant T_{\max_{B11}}]. \quad (13)$$

Note that for the remaining five models the SAS and EG can be modelled accordingly. As there are no logic states, there is no FSM. The mode of the SAS is selected through the following MS:

$$i(k) = \begin{cases} 1 & \text{if } (\bar{\delta}_{1_{B11}} \& \bar{\delta}_{2_{B11}})|u_{b_{V111}} = 1 \\ 2 & \text{if } (\bar{\delta}_{1_{B12}} \& \bar{\delta}_{2_{B12}})|u_{b_{V112}} = 1 \\ 3 & \text{if } (\bar{\delta}_{1_{B13}} \& \bar{\delta}_{2_{B13}})|u_{b_{V113}} = 1 \\ 4 & \text{if } (\bar{\delta}_{1_{B31}} \& \bar{\delta}_{2_{B31}})|u_{b_{V211}} = 1 \\ 5 & \text{if } (\bar{\delta}_{1_{B32}} \& \bar{\delta}_{2_{B32}})|u_{b_{V212}} = 1 \\ 6 & \text{if } (\bar{\delta}_{1_{B11}}|u_{b_{V111}}|\bar{\delta}_{1_{B12}}|u_{b_{V112}}|\bar{\delta}_{1_{B13}}|u_{b_{V113}}| \\ & \quad \bar{\delta}_{1_{B31}}|u_{b_{V211}}|\bar{\delta}_{1_{B32}}|u_{b_{V212}}) = 0. \end{cases} \quad (14)$$

The HYSDEL code for the presented DHA system of reactor R21 is given in Appendix A.

The modelling procedure presented for reactor R21 can be also applied to all the remaining reactors and tanks. The obtained model is valid as long as the processing times reported in Table 2 are representing the upper bounds, i.e. robust according to the processing times uncertainties. Due to the extensiveness of the complete DHA system for the Multiproduct Batch Plant, modelled using the HYSDEL modelling language, the description is not given here but can be found on the web site http://msc.fe.uni-lj.si/potocnik.

Providing the DHA model of a Multiproduct Batch Plant described by the HYSDEL modelling language description, the HYSDEL tool generated the equivalent MLD form (2). The dimensions of the corresponding variables are: $\mathbf{x}(k) \in \mathbb{R}^{28} \times \{0,1\}^6$, $\mathbf{u}_b(k) \in \{0,1\}^{12}$, $\boldsymbol{\delta}(k) \in \{0,1\}^{85}$, and $\mathbf{z}(k) \in \mathbb{R}^{40}$. Matrices $\mathbf{A}$, $\mathbf{B}_1$, $\mathbf{B}_2$, $\mathbf{B}_3$, $\mathbf{C}$, $\mathbf{D}_1$, $\mathbf{D}_2$ and $\mathbf{D}_3$ have suitable dimensions. Matrices $\mathbf{E}_1$ to $\mathbf{E}_5$ define 511 inequalities. It can be easily noticed that the system has 12 discrete inputs $\mathbf{u}_b(k)$ and no continuous ones.

4.3. Control of the Multiproduct Batch Plant

Problem formulation:

For a given initial condition control the production of "Blue" and "Green" to minimise the production time.

Given that the raw material batches "Yellow", "Red" and "White" are delivered at fixed, given times which are known in advance (see Table 3), the degrees of freedom are:

- the times at which one batch of "Yellow", "Red" or "White" is emptied into a reactor
- the selection of a reactor in which the raw material will be emptied.

Table 3
Delivery times in [min:s] for raw material

Yellow	Red	White	
0:00	0:10	0:00	4:20
0:30	0:40	0:30	5:30
1:10	1:50	2:00	6:20
4:20	5:30	2:20	6:55
5:40	6:50	3:00	7:10
6:45	7:50	3:35	8:00

We focus on the problem of producing six batches of "Blue" and "Green" constrained by the delivery times of batches "Yellow", "Red" and "White" given in Table 3.

The solution of a control problem is a control sequence $U_0^{k_{\text{fin}}-1} = \{\mathbf{u}_b(0), \ldots, \mathbf{u}_b(k_{\text{fin}}-1)\}$. Given that we cannot influence the system through the first three inputs (pumps P1-P3) because they are predefined by the delivery times given in Table 3, at each time step only nine remaining inputs (valves V111-V113, V121-V123 and V131-V133) must be determined. The minimal production time can be over-bounded from Tables 2 and 3, although such a time may not be feasible. The last batch of white is delivered after 480 seconds; additionally, $12 + 9 + 12 = 33$ seconds must be added to finish the production, so the minimum time cannot be less than $T_{\min} = 513$ s and consequently $U_0^{k_{\text{fin}}-1} \in \{0,1\}^{4617}$ given that $T_s = 1$ s. Because all the inputs are discrete, 2^{4617} possible combinations of the solution vector $U_0^{k_{\text{fin}}-1}$ exist and searching for the solution through all the combinations is practically impossible.

According to the initial conditions and the cost function criterion introduced earlier (see Eqs. (3)–(5)), we propose the following cost function:

$$J_i(v,k) = (3 \cdot V_{\max} - v)F + k, \quad (15)$$

where $V_{\max}$ is the *volume of all products*, v represents the current volume of material pumped/emptied into the first, second and third layer (see Fig. 3), k is the current step and F is a factor whose properties will be explained later. The goal (final state $\mathbf{x}_{\text{fin}}$) is reached when $v = 3 \cdot V_{\max}$. The cost function value in the feasible solution is

$$J_i = k. \quad (16)$$

According to (6), the cost function (15) has to be monotonically increasing, i.e.

$$\begin{aligned} &J(v(k+1), k+1) - J(v(k), k) \\ &\quad = (3 \cdot V_{\max} - v(k+1))F + (k+1) \\ &\qquad - (3 \cdot V_{\max} - v(k))F - k \\ &\quad = -(v(k+1) - v(k))F + 1 \geqslant 0. \end{aligned} \quad (17)$$

and hence the parameter F must satisfy the condition

$$F \leqslant \frac{1}{v(k+1) - v(k)}. \tag{18}$$

Note that according to property (4b), it holds that the denominator in Eq. (18) $-v(k+1) + v(k) \leqslant 0$ or $v(k+1) - v(k) \geqslant 0$ for all k. In the worst case, the value for $\Delta v \triangleq v(k+1) - v(k)$ is defined through the estimation of the maximum change of volume in the system neglecting the delivery times. If all three raw materials are delivered, two reactors are filled and the third is emptied into a buffer tank at the same time, i.e. $3 \cdot 0.85/12$ (see Table 2) for filling all the three tanks in upper layer, $0.85/9$ and $0.85/11$ represent the filling of two reactors using maximum flow (shortest delivery time) and $1.7/12$ the emptying of the reactor into buffer tank, then the following estimation for Δv is obtained

$$\Delta v = 3\frac{0.85}{12} + \frac{0.85}{9} + \frac{0.85}{11} + \frac{1.7}{12} = 0.526. \tag{19}$$

Hence it follows that

$$F \leqslant \frac{1}{\Delta v} = \frac{1}{0.526} = 1.90. \tag{20}$$

Regarding the node selection criterion, it is reasonable to choose the node that leads to the best (current) optimal solution, i.e. the node with the smallest associated cost function value J_i at step k (the influence of step instance k is the same, but the influence of v is greater).

4.4. Results

Because of the complexity of the optimal control problem for the Multiproduct Batch Plant analysed in the previous section, the algorithm provides an optimal solution for a bounded time interval and a sub-optimal solution for the complete optimal control problem. The computational times refer to a MATLAB implementation running on a Pentium III 667 MHz machine.

The properties of the applied algorithm are presented on a time interval $k \in [0, 50]$ and $k \in [0, 40]$ for two values of parameter, $F = 1.80$ and 1.90. For the time interval $k \in [0, 50]$, an approach based on enumeration would examine $2^{9\times 50} \doteq 2.9 \times 10^{135}$ possible solutions. The results are summarised in Table 4.

Table 4
Results on bounded time interval

Time int.	Param. F	Tree size	CPU time
$k \in [0, 40]$	1.90	32205	0:22:26
$k \in [0, 40]$	1.80	36852	0:25:48
$k \in [0, 50]$	1.90	497799	5:48:10
$k \in [0, 50]$	1.80	608826	7:11:23

The results illustrate that parameter F exerts a great influence on the size of the tree and, indirectly, on the time needed to solve the optimisation problem.

As the complexity of the optimisation problem grows exponentially with the time horizon, a sub-optimal algorithm based of additional knowledge on the process is preferred and gives satisfactory results in an acceptable time.

We added a constraint on the tree size of 100,000 nodes ($\approx$1 hour) to the proposed algorithm and set the parameter F to 1.90. The solution is presented in Table 5

Table 5
Production time [min:s] and corresponding reactor

Blue	Reactor	Green	Reactor
0:33	R22	0:52	R21
2:42	R23	2:21	R22
4:00	R21	3:21	R22
4:53	R22	5:52	R21
6:41	R22	7:32	R21
7:18	R22	8:23	R21

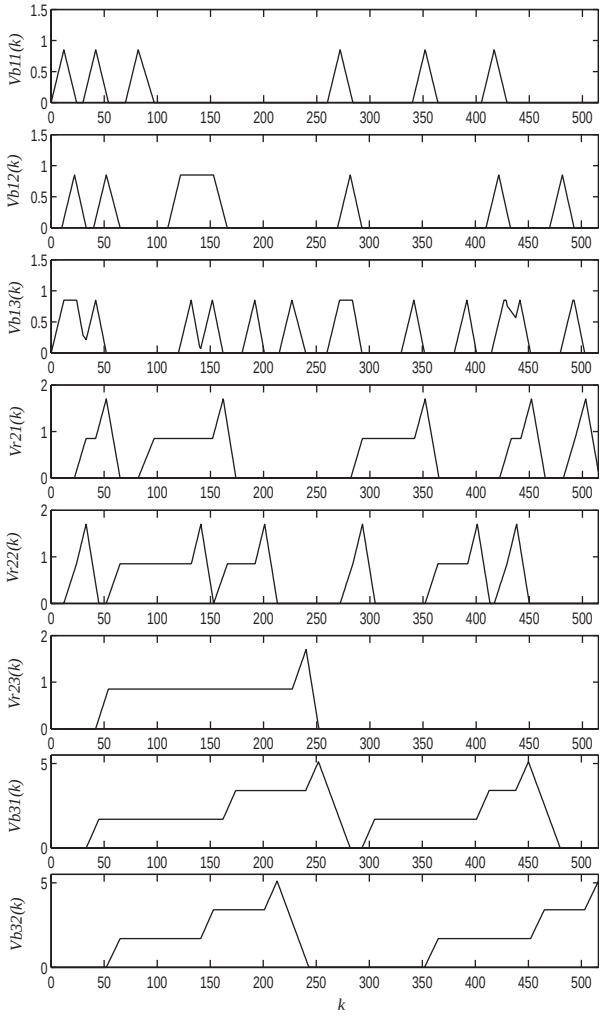

Fig. 5. Volumes in the tanks.

and in Fig. 5. The first, and in this case the final, sub-optimal solution was obtained in just 27 seconds. The production of six batches of "Blue" and "Green" can be achieved in 515 s. The lower bound to the global optimal solution (513 s) is missed only by two seconds. We remark that the sub-optimal solution is feasible and could also be optimal, but the algorithm would need much more time to prove optimality.

5. Conclusions

An optimal control problem for systems with discrete inputs only has been addressed. The problem was solved by combining hybrid modelling and reachability analysis with a branch-and-bound technique. The algorithm abstracts the behaviour of the hybrid system into a "tree of evolution". The main advantage of the approach is that the "tree" is cut from both sides, top and bottom, which result in considerable size reduction of the tree. The proposed approach was applied to a Multiproduct Batch Plant with satisfactory results. Future research will be devoted to adapting the proposed optimal control approach to model predictive control techniques.

Appendix A

HYSDEL CODE - Reactor R21

```
SYSTEM reactorR21 {
INTERFACE{
                        STATE {
                                        REAL Tb11 [0,15], Tb12 [0,11], Tb13 [0,10]; /* Timers */
                                        REAL Vr21 [0,1.7]; /* Volume in R21 */
                                        REAL Tb31 [0,12], Tb32 [0,13];} /* Timers */
                        INPUT {
                                        BOOL uv111, uv112, uv113, uv211, uv212; }/* Inputs */
                        PARAMETER {
                                        REAL VB11=0.85, VB12=0.85, VB13=0.85; /* Batch volume */
                                        REAL TB11=15, TB12=11, TB13=10; /* Process times */
                                        REAL VR21=1.7; /* Batch volume */
                                        REAL TB31=12, TB32=13;} } /* Process times */
IMPLEMENTATION {
                        AUX {
                                        REAL Tb11p, Vb11p; /* Aux. variables for states */
                                        BOOL d1b11, d2b11; /* Aux. variable for timer status */
                                        REAL Tb12p, Vb12p;
                                        BOOL d1b12, d2b12;
                                        REAL Tb13p, Vb13p;
                                        BOOL d1b13, d2b13;
                                        REAL Tb31p, Vb31p;
                                        BOOL d1b31, d2b31;
                                        REAL Tb32p, Vb32p;
                                        BOOL d1b32, d2b32;}
                        AD {
                                        d1b11 = Tb11 <= 0; /* Set timer status */
                                        d2b11 = TB11 - Tb11 <= 0;
                                        d1b12 = Tb12 <= 0;
                                        d2b12 = TB12 - Tb12 <= 0;
                                        d1b13 = Tb13 <= 0;
                                        d2b13 = TB13 - Tb13 <= 0;
                                        d1b31 = Tb31 <= 0;
                                        d2b31 = TB31 - Tb31 <= 0;
                                        d1b32 = Tb32 <= 0;
                                        d2b32 = TB32 - Tb32 <= 0;}
```

```
DA {
                Tb11p = {IF uv111 THEN Tb11+1}; /* Set timer value */
                Vb11p = {IF uv111 THEN VB11/TB11}; /* Set volume */
                Tb12p = {IF uv112 THEN Tb12+1};
                Vb12p = {IF uv112 THEN VB12/TB12};
                Tb13p = {IF uv113 THEN Tb13+1};
                Vb13p = {IF uv113 THEN VB13/TB13};
                Tb31p = {IF uv211 THEN Tb31+1};
                Vb31p = {IF uv211 THEN VR21/TB31};
                Tb32p = {IF uv212 THEN Tb32+1};
                Vb32p = {IF uv212 THEN VR21/TB32};}
CONTINUOUS {
                Vr21 = Vr21+Vb11p+Vb12p+Vb13p-Vb31p-Vb32p; /* Update states */
                Tb11 = Tb11p;
                Tb12 = Tb12p;
                Tb13 = Tb13p;
                Tb31 = Tb31p;
                Tb32 = Tb32p;
MUST {
                (~ d1b11& ~ d2b11)->uv111; /* Control the input to have */
                (~ d1b12&~ d2b12)->uv112; /* a specified form */
                (~ d1b13&~ d2b13)->uv113;
                (~ d1b31&~ d2b31)->uv211;
                (~ d1b32&~ d2b32)->uv212;
                /* Constrain inputs to one active input at a time */
                (REAL uv111)+(REAL uv112)+(REAL uv113)+(REAL uv211)+(REAL
                uv212)<=1;}}
}
```

References

Antsaklis, P. J. (Ed.). (2000). Special issue on hybrid systems: Theory and applications. *Proceedings of the IEEE*, *88* (7).

Barton, P. I., Banga, J. R., & Galán, S. (2000). Optimization of hybrid discrete/continuous dynamic systems. *Computers and Chemical Engineering*, *24*, 2171–2182.

Bauer, N. (2000). A demonstration plant for the control and scheduling of multi-product batch operations. Techinical Report—VHS Case Study 7, University of Dortmund, Dortmund.

Bauer, N., Kowalewski, S., Sand, G., & Löhl, T. (2000). A case study: Multi product batch plant for the demonstration of control and scheduling problems. *ADPM2000 conference proceedings*, Dortmund, Germany (pp. 383–388).

Bemporad, A., Giovanardi, L., & Torrisi, F. D. (2000). Performance driven reachability analysis for optimal scheduling and control of hybrid systems. In *Proceedings of the 39th IEEE conference on decision and control*, Sydney, Australia (pp. 969–974).

Bemporad, A., & Mignone, D. (2000). MIQP.M: A Matlab function for solving mixed integer quadratic programs, ETH Zurich, code available at http://control.ethz.ch/~hybrid/miqp.

Bemporad, A., & Morari, M. (1999). Control of systems integrating logic, dynamic, and constraints. *Automatica*, *35*(3), 407–427.

Bemporad, A., Torrisi, F. D., & Morari, M. (2001). Discrete-time hybrid modeling and verification of the batch evaporator process benchmark. *European Journal of Control*, *7*(4), 382–399.

Cassandras, C. G., Pepyne, D. L., & Wardi, Y. (2001). Optimal Control of a Class of Hybrid Systems. *IEEE Transanctions on Automatic Control*, *46*(3), 398–415.

Gokbayrak, K., & Cassandras, C. G. (1999). A hierarchical decomposition method for optimal control of hybrid systems. In *Proceedings of the 38th IEEE conference on decision and control*, Phoenix, AZ, USA (pp. 1816–1821).

Hedlund, S., & Rantzer, A. (1999). Optimal control of hybrid systems. In *Proceedings of the 38th IEEE conference on decision and control*, Phoenix, AZ, USA (pp. 3972–3976).

Heemels, W. P. M. H., De Schutter, B., & Bemporad, A. (2001). Equivalence of hybrid dynamical models. *Automatica*, *37*(7), 1085–1091.

Lincoln, B., & Rantzer, A. (2001). Optimizing linear system switching. In *Proceedings of the 40th IEEE conference on decision and control*, Orlando, FL, USA (pp. 2063–2068).

Rantzer, A., & Johansson, M. (2000). Piecewise linear quadratic optimal control. *IEEE Transanctions on Automatic Control*, *45*(4), 629–637.

Torrisi, F. D., & Bemporad, A. (2002). HYSDEL—A tool for generating computational hybrid models, *IEEE Transactions on Control Systems Technology*, in press.

Torrisi, F. D., Bemporad, A., Bertini, G., Hertach, P., Jost, D., & Mignone, D. (2002). HYSDEL—User Manual, Techinical Report AUT02-10, Automatic Control Laboratory, ETH, Zrich.