A Petri Nets-based Scheduling Methodology for Multipurpose Batch Plants

Daniel Riera
Mercedes Narciso
Systems Engineering and Automatic Control Group
Universitat Autònoma de Barcelona
Bellaterra, 08193
{**daniel.riera, mercedes.narciso**}**@uab.es**

Chouaib Benqlilou
Chemical Engineering Departement,
UPC
Barcelona, Spain
chouaib.benqlilou@upc.es

Keywords: Petri nets, Coverability tree, Simulation, Scheduling, Batch

Abstract

This article presents an optimization methodology of batch production processes assembled by shared resources which rely on a mapping of state-events into time-events allowing in this way the straightforward use of a well consolidated scheduling policies developed for manufacturing systems. A technique to generate the timed Petri net representation from a continuous dynamic representation (Differential-Algebraic Equations systems (DAEs)) of the production system is presented together with the main characteristics of a Petri nets-based tool implemented for optimization purposes. This paper describes also how the implemented tool generates the coverability tree and how it can be pruned by a general purpose heuristic. An example of a distillation process with two shared batch resources is used to illustrate the optimization methodology proposed.

1 INTRODUCTION

In the last few years, many methods and tools have been developed to improve production performance in the manufacturing industry. These approaches try to tackle changes in production objectives such as high production diversity (instead of high production volume), *make to order* (instead of *make to stock*), and *zero stock* (*just in time*) policies. In this paper, a methodology which uses a Petri nets-based tool to optimize a system assembled by a set of shared batch production resources is presented. Petri nets have been used as the modelling formalism since they support properties like parallelism, which is very useful for the description of production processes (Murata [1989], Silva [1985]). By using the proposed methodology, it is possible to optimize a wide range of scheduling problems from the process industry taking advantage of some algorithms developed in the manufacturing industry field (Proth and Xie [1996], Silva and Valette [1989]).

Traditionally, optimization of batch plants was reduced to a task sequencing problem where the processing units to be visited by the different products and the time to be spent in each one was specified (Fishwick [1995]). According to each particular configuration between processing resources (e.g. No Intermediate Storage, Unlimited Intermediate Storage, Finite Intermediate Storage, etc. (Bahri et al. [2000])), different planning policies can be obtained.

Since planning methods developed for manufacturing industries cope with the NP-hard problems that appear due to the flexibility and reusability of shared production resources (robots, Computer Numeric Control machines (CNCs), belts, etc.), the proposed methodology has been evaluated on high flexible batch plants. Thus, in the batch plant considered in this paper, there are r ($r > 1$) products to be produced, s ($s > 1$) multipurpose shared processing units, a variable processing time according to the amount of product and type of product to be processed in each resource, and a preemptive policy in the interaction between resources. The objective is to optimize the process, finding the fastest way to treat a certain amount of raw material in order to obtain the manufactured product.

By generating the Petri net of the process (see Section 2) and using a tool that prunes the Petri net coverability tree (see Section 3), the manufacturing lead time of the system can be optimized.

Figure 1 illustrates the representative system that will be used to introduce the modelling methodology, which is formed by two shared batch resources (boilers) with capacity 2 and 5 litres respectively, and with different performance. The raw material is stored in different tanks which hold ethanol mixed with water at different concentrations. Boilers are used to improve the mix purity by heating it to a vaporization phase, and maintaining the pressure and temperature at a certain liquid-vapour equilibrium phase during a variable amount of time for the mix distillation. Raw material is placed in six tanks each filled with different concentration of ethanol (from 0.7 to 0.2), and the processed mix is stored in six more empty tanks (with target concentrations from 0.6 to 0.1). Given a set of initial quantities (e.g. 50 l.(0.7)[1], 70 l.(0.6) and 50 l.(0.4)), the aim is to find the fastest process (using both boilers) to achieve a desired output (e.g. 30 l.(0.2) and 10 l.(0.1)).

According to the distillation dynamic, the process is modelled as a sequence of 2 tasks:

1) *initial heating phase*. The mix is heated to the evaporation temperature and pressure. From this point, temperature and pressure remain in a steady state. During this phase the concentration decreases $\triangle C = -0.1$.

[1] 50 l.(0.7) signifies 50 litres of liquid with concentration 0.7.

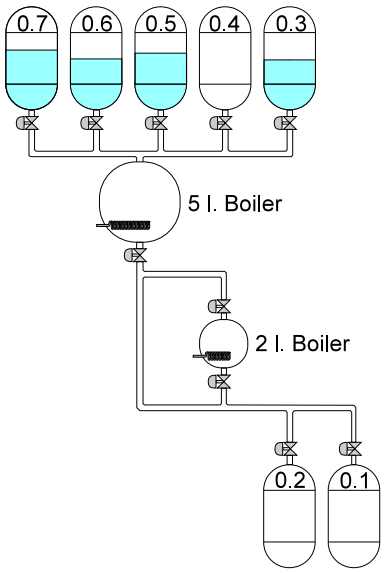
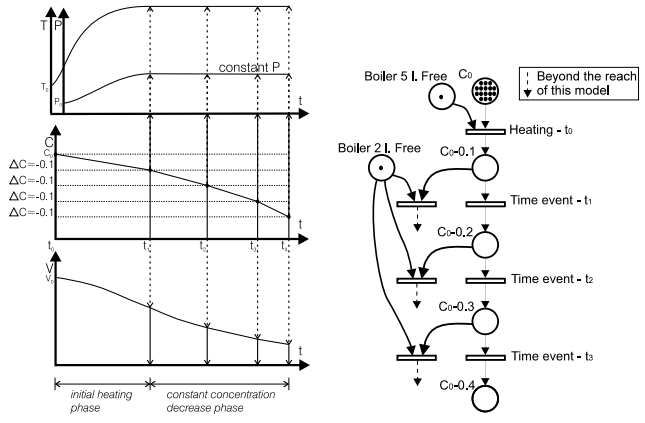


Fig. 1. The representative system

2) *constant concentration decrease phase*. The mix is maintained at a constant pressure while the concentration of ethanol in the mix decreases a fixed rate ($\triangle C = -0.1$).

Thus, the distillation process can be modelled as a sequence of an initial *heating phase* followed by *c constant concentration decreasing phases*, where *c* will depend on the initial concentration of the mix, and the target concentration. Since a preemptive policy is allowed, at the end of any *constant concentration decrease phase* an event can occur to move a certain amount of the mix from one boiler (5-litre boiler) to the other (2-litre boiler).



(a) (b)

Fig. 2. State-event to time-event mapping

Figure 2(a) illustrates the evolution of the variables of the system (temperature, pressure, volume and concentration). In order to make the mapping from state-events to a time-events system, the process is discretised in terms of increments of concentration ($\triangle C = -0.1$).

Figure 2(b) shows the transformation into discretised system states and transitions fired by time-events.

In this paper, several assumptions about the batch plant are made in order to simplify the system and to focus on attention to the main phases of the proposed optimization methodology:

- The setup time to pour the liquid either to the five-litre boiler or from this to the two-litre one, is zero. In the latter case, the liquid does not lose neither temperature nor pressure in the exchange (restricted model).
- Because of the need to have sufficient surface for the evaporation, the boilers can not be fully filled. The five-litre boiler can be filled up to 4.5 litres and the two-litre boiler to 1.4.
- The *initial heating phase* happens once for every quantity of liquid processed.
- The five-litre boiler can perform both phases while the two-litre one can only perform the second phase due to its low performance for the *initial heating phase*.

2 CONTINUOUS MODEL TO PETRI NET METHODOLOGY

A. Petri nets background

A *Petri net* (PN) is a particular kind of directed graph, together with an initial state called the *initial marking*. An ordinary PN is a 5-tuple $N = (P, T, I, O, M_0)$:

- $P = \{p_1, \ldots, p_n\}$ is the set of places, represented graphically by circles.
- $T = \{t_1, \ldots, t_m\}$ is the set of transitions, represented graphically by bold lines or rectangles.
- $I : (P \times T) \rightarrow \mathbb{N}$ is a function that defines the weight of directed arcs from places to transitions.
- $O : (T \times P) \rightarrow \mathbb{N}$ is a function that defines the weight of directed arcs from transitions to places.
- M_0 is the initial marking.

A *marking* is an array that assigns to each place a non-negative integer. If a marking assigns to place p a value k ($k \in \mathbb{Z}, k \geq 0$), p is marked with k tokens, which are represented graphically by black dots.

A transition t_i is said to be enabled by a marking M, if $\forall p \in M : M(p) \geq I(p, t_i)$. The firing transition generates a new marking M' which can be computed by withdrawing $I(p_k, t_i)$ tokens from each p_k input place of t_i, and by adding $O(p_j, t_i)$ tokens to each p_j output place of t_i.

In manufacturing terms, *transitions* are used to model operations (firing a transition can represent a task (or process) initiation or an ending of a task), *places* are used to model buffers and resources status, connecting *arcs* specify logical relationships and resource constraints among operations, and *tokens* represent material and resources conditions. Metamodelling in simulation of complex systems is traditionally used to carry out simulation in a computationally more efficient way than the simulation itself. By associating time functions to transitions (*timed Petri nets*) a system whose behaviour is time dependent can be described (Popova-Zeugmann [1993], Gu and Bahri [1999]). Tables 1–3 summarize the time functions obtained by discretising the continuous dynamics (*initial*

heating phase, *constant concentration decrease phase*) of each boiler.

Given a batch production system which is composed by a set of shared resources the specification of the system topology, subsystem dynamics, raw material and production targets in the Petri nets formalism can be achieved by the following four steps:

- Analysis of System Topology.
- Selection of Places and Transitions.
- Time Transition Functions Specification.
- Cost Function Specification.

B. Analysis of System Topology

Given a system, the analysis of its topology will lead to a specification of subsystem components together with the subsystem interaction. By using a bottom-up approach each subsystem together with the interactions with its context can be formalized in one sub-Petri net. Once a Petri net has been specified for each shared resource, the Petri net specification of the overall system can be obtained by assembling shared places of the different sub-Petri Nets.

In the considered process (see Figure 1), two subsystems (shared boilers) have been identified:

- *5-litre boiler*: This subsystem consists of six entry tanks (with liquid concentrations from 0.7 to 0.2), a five-litre boiler and six exit tanks (from 0.6 to 0.1)(notice that in Figure 1 only the relevant tanks for later experiments are pictured).
- *2-litre boiler*: This subsystem consists of a mass flow entry from the 5 litre boiler, (with mix concentrations from 0.7 to 0.2), a two-litre boiler and six exit tanks (from 0.6 to 0.1).

It cannot be assumed that the two subsystems are independent. Although they are modelled separately, they have some dependent dynamics which must be taken into account when the subsystems are assembled to obtain the representation of the global production process.

C. Selection of Places and Transitions

The second step is to determine the Petri net components that must be used to formalize each subsystem description:

- Places: The indivisible parts of each subsystem (atomic elements). Usually they are queues (tanks) and servers (processing units).
- Transitions: All the events that represent a possible change in the state of the system.
- Arcs: Specify the relationship between events (Transitions) and queues or servers (Places).
- Tokens: Elements that are used to describe the matter that flows through the system, or Boolean information that describes the state of a shared resource.

Thus, the whole production process can be formalized by a set of tasks (represented by Petri net components) that cannot be decomposed any further: filling a boiler, filling a tank, heating phase, concentration decrement phases and draining the product.

Two types of places are used: queue and binary places. The former — which can contain a number of tokens between 0 and *u* — are used to represent the material, the available space in a boiler, the number of repetitions of an operation, etc. The binary places — with 0 or 1 tokens — are used to indicate whether a resource is currently being used or not, to check if a decision has been taken, etc.

The Petri net of the system being considered is shown in Figure 3.

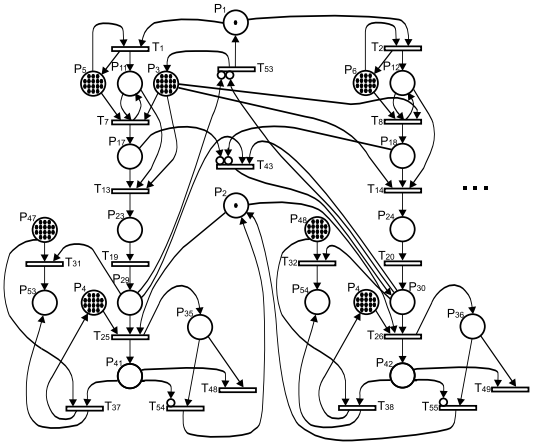


Fig. 3. Petri net representing the system

where the places represent:

- p_1,p_2: Boiler Free (5 l. and 2 l.).
- p_3,p_4: Boiler available free space (5 l. and 2 l.).
- p_5,...,p_{10}: Initial volume[2].
- p_{11},...,p_{16}:Initial concentration selected[2].
- p_{17},...,p_{22}: Amount of mix in 5 l. boiler[2].
- p_{23},...,p_{28}: Heating mix in 5 l. boiler[2].
- p_{29},...,p_{34}: Amount of hot mix in 5 l. boiler[3].
- p_{35},...,p_{40}: Ethanol concentration in 2 l. boiler[3].
- p_{41},...,p_{46}: Amount of hot liquid in 2 l. boiler[3].
- p_{47},...,p_{52}: Required amount of final product[3].
- p_{53},...,p_{58}: Amount of mix processed[3].

and the transitions:

- t_1,...,t_6: Initial filling of 5 l. boiler from raw mix tanks[2].
- t_7,...,t_{12}: Filling a 5 l. boiler from a tank[2].
- t_{13},...,t_{18}: Heating start in 5 l. boiler[2].
- t_{19},...,t_{24}: Heating end in 5 l. boiler[2].
- t_{25},...,t_{30}: Liquid movement into 2 l. boiler[3].
- t_{31},...,t_{36}: Draining the mix from 5 l. boiler[3].
- t_{37},...,t_{42}: Draining the mix from 2 l. boiler[3].
- t_{43},...,t_{47}: Initiating one step *constant concentration decrease phase* in 5 l. boiler[4].
- t_{48},...,t_{52}: Initiating one step *constant concentration decrease phase* in 2 l. boiler[4].
- t_{53},t_{54}: Boiler release[3].

[2]For each concentration from 0.7 to 0.2
[3]For each concentration from 0.6 to 0.1
[4]For each concentration from 0.6 to 0.2

D. Metamodels: Specification of Time Transition Functions

Every transition has an associated time which corresponds to the time spent by the system to 'move' tokens from the entry places to the exit places of that transition. This time can be either constant or variable according to the event dynamic represented by the transition. When the time value is a constant, it is usually zero — no transition time is consumed to move the tokens from the entry places to the exit ones (e.g. a logical change in the state of a shared resource: from busy to available). When the transition time is variable, it is represented by a function which is formulated by experimenting with the mathematical continuous model (DAE) in a simulator.

Computer intelligence metamodels are used in simulation for different goals such as prediction, optimization, verification or validation of simulation models. Here, regression functions are specified by simulating the heating and evaporation phases according to different operating conditions. Results obtained are applied to perform what-if analysis and to determine the best scheduling policy.

Thus, to derive the time function of a transition it is necessary first to obtain the mathematical continuous representation of each subsystem behaviour. After the subsystem has been modelled and simulated, the results are used to generate the time functions. If the resulting function depends on the number of tokens received by the transition, it is simulated for different entry values (represented in the Petri net by different number of entry tokens to a transition). Otherwise, it has a value which will correspond to a unique simulation result. Once the times have been calculated, it is possible to find a polynomial that fits with these values.

A transition may also have an associated function to represent a relationship between input and output tokens which depends on the quantity of the input tokens (e.g. consumption of the mass mix due to evaporation). Functions describing this relationship are calculated in a similar way as time functions. Tables I to III present the polynomials for time and volume consumption associated to the corresponding transitions. Once these functions have been calculated, the Petri net of the overall production process can be formalized.

C_0	$\triangle$V	$\triangle$T
0.7	$0.447 \cdot V_0 - 0.209$	$3.022 \cdot V_0 - 3.721$
0.6	$0.338 \cdot V_0 - 0.078$	$2.700 \cdot V_0 - 3.500$
0.5	$0.272 \cdot V_0 - 0.005$	$2.530 \cdot V_0 - 4.000$
0.4	$0.231 \cdot V_0 + 0.151$	$2.468 \cdot V_0 - 3.427$
0.3	$0.207 \cdot V_0 + 0.250$	$2.515 \cdot V_0 - 2.015$
0.2	$0.206 \cdot V_0 + 0.294$	$2.766 \cdot V_0 + 0.574$

TABLE I

TIME AND VOLUME CONSUMPTION POLYNOMIALS FOR THE HEATING IN THE FIVE-LITRE BOILER

In Tables I to III, V_0 corresponds to the initial amount of mix with concentration C_0, and $\triangle T$ and $\triangle V$ are the time and volume consumption polynomials corresponding to a step ($\triangle C = -0.1$) in the mixing process.

C_0	$\triangle$V	$\triangle$T
0.6	$0.326 \cdot V_0 + 0.005$	$5.685 \cdot V_0 + 0.103$
0.5	$0.263 \cdot V_0 + 0.044$	$4.799 \cdot V_0 + 0.790$
0.4	$0.214 \cdot V_0 + 7.988$	$4.291 \cdot V_0 + 30.124$
0.3	$0.200 \cdot V_0 - 0.035$	$4.229 \cdot V_0 - 0.604$
0.2	$0.198 \cdot V_0 + 0.015$	$4.823 \cdot V_0 + 0.365$

TABLE II

TIME AND VOLUME CONSUMPTION POLYNOMIALS FOR THE TWO-LITRE BOILER

C_0	$\triangle$V	$\triangle$T
0.6	$0.340 \cdot V_0 + 0.110$	$1.429 \cdot V_0 + 1.867$
0.5	$0.274 \cdot V_0 + 0.067$	$1.217 \cdot V_0 + 3.216$
0.4	$0.233 \cdot V_0 + 0.327$	$1.106 \cdot V_0 + 5.708$
0.3	$0.210 \cdot V_0 - 0.136$	$1.094 \cdot V_0 + 4.266$
0.2	$0.209 \cdot V_0 - 0.035$	$1.262 \cdot V_0 - 4.619$

TABLE III

TIME AND VOLUME CONSUMPTION POLYNOMIALS FOR THE FIVE-LITRE BOILER

E. Cost Function Specification

Once the Petri net is finished, the weights associated to the places are allocated. These weights are used by the tool which prunes the coverability tree. This tool uses heuristics (see Section 3) associated with these weights to calculate the 'quality' of any possible discretised state of the system (a node of the coverability tree).

These weights may be used to give information to the optimization tool about the use (and cost) of the different resources of the system.

3 COVERABILITY TREE

Given a Petri net and an initial marking M_o (initial state of the system: amount of raw material, state of each boiler,etc), the coverability tree is a very suitable representation to describe both:

- All the states (markings) which can be reached starting from these initial system conditions (M_o), and:
- The transition sequence to be fired to drive the system from the initial condition to a desired end-state or final marking (M_f).

The initial marking (M_o) can be represented by a vector of *n* elements (see Equation 1), where *n* corresponds to the number of places used in the Petri net representation of the system.

$$M_o = [\ TokPlace_1 \quad \cdots \quad TokPlace_n\ ] \tag{1}$$

The coverability tree evaluates for each system state (marking) the possible events that can appear (enabled transitions) and generates for each event the new marking (new state) obtained after the event has been processed.

Figure 4 illustrates the first 4 levels of the coverability tree of the system being studied. It can be seen that given an initial marking there are only 2 events (represented by transitions T1 and T2) which can appear, leading to two new states (markings

in the first level of the coverability tree). For each one of these new states, the enabled events are fired obtaining new states and so on. It can be readily noted that the coverability will expand according to the flexibility (number of shared resources to be assigned at each time) of the production process, which will lead to a NP-hard problem when evaluating scheduling policies of complex systems. Such problems require a great amount of computer resources (like CPU time and memory).

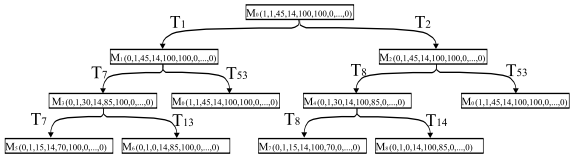


Fig. 4. Example of first levels of a coverability tree

Since it is not always possible to build the total coverability tree, two solutions are available:

- Choose a higher abstraction level to describe the system characteristics.
- Use heuristics to prune the tree.

The former option can reduce the size of the coverability tree considerably, however, it makes generally impossible to generate a plan with sufficient detail for all the production architecture decision variables. On the other hand, although the latter option does not guarantee an optimal solution, it offers a good enough answer for industrial scheduling and re-scheduling requirements. These are forced (due to time constraints) to accept a prompt sub-optimal solution rather than a delayed optimal solution. Note that fast re-scheduling is essential to quickly react to changes (e.g. machine shutdowns).

A. Defining the Objective Function

The formalization of an objective function to drive the program through the search space will allow the inclusion of certain expert knowledge and express it in the mathematical formalism used by the search algorithms. The knowledge expressed through the objective function can be used to select those markings (states) within the solutions space which could lead to the optimal solution.

In industry, production requirements are usually defined as a compromise between time and cost. To assess a production process, the engineer has to be aware of performance indexes such as: total time that a part spends in a queue; total time that parts spend in transport systems; equipment utilization; proportions of time a machine is broken, blocked (waiting for a finished part to be removed), or undergoing setup operations, etc.

The production performance indexes are group together into a suitably defined cost function. This cost function is formed by two components: a place or 'work in process' (WIP) component and a time component.

'Work in process' is the current number of pieces (or quantity of material, in the processes industry case) in the production line. In terms of Petri nets, the WIP can be obtained by computing the sum of tokens in every place representing a stock. Thus, the cost that a company pays for pieces stored in particular queues (places), can be expressed mathematically by an objective *P-function* (see Equation 2).

$$J_p = \sum_{i=1}^{n} A_i \cdot P_i \tag{2}$$

where P_i represents the internal value of the place i (number of pieces or tokens stored in queue i), and A_i is a weighting parameter defined by the user.

Note that the performance of the *P-function* depends on the selection of the weighting parameters. Thus, the user can penalize those places where tokens should not encounter long delay (row material places).

In the case under consideration, different policies have been studied by changing the weight of the tokens in places such as the initial liquid tanks (in order to empty stocks as soon as possible), or the two and five-litre boilers (in order to use them for shorter or longer), etc.

The second component of the cost function corresponds to the processing time required to drive the system from an original work load to a desired particular work load. It is computed by evaluating the firing time of each activated transition. An objective *T-function* (J_t) is proposed to describe the time of a given sequence of transitions. The *T-function* is defined in Equation 3.

$$J_t = \max(t_1, \ldots, t_m) \tag{3}$$

where t_i represents the time when transition i is fired.

In order to look for a good compromise between the manufacturing lead time and the work in process, a cost function (see Equation 4) has been formulated containing the values J_t and J_p.

$$J = (\mathrm{e}^{\frac{1}{J_t} \cdot \ln \frac{J_p}{J_{p0}}} - 1) \cdot 100 \tag{4}$$

where J_{p0} is the value of the *P-function* for the initial marking (M_0).

B. Use of a Heuristic Pruning Algorithm

The aim of the heuristic algorithm is to decompose the complete coverability tree of a Petri net into a set of t sub-trees of constant depth l, where l is the number of fired transitions. The values for the parameters t and l are selected by the user.

When a sub-tree is pruned, t paths are selected, where each path is a sequence of transitions from the root to a final node at the level l. Paths selection is made through the cost function (see Equation 4) comparison by using the value at level l. Selected paths are stored with the initial and final markings. A path is selected according to the following criteria:

Given a set of t selected paths ψ a non-selected path s will be selected if:

$$\forall p \in \psi : J(s) - J(p) < 0 \tag{5}$$

The t selected markings are used as root markings for the next sub-tree set. The first tree (marking root M_0) is pruned at

level l and only t paths are selected. From these t paths, a set of sub-trees is generated. Once each one of these sub-trees has been pruned, the best paths are selected and stored, obtaining in this way a set of $t \times t$ paths, from which only the best t paths are retained, and the others are removed.

This operation is repeated successively until the target mark is achieved (see Figure 5).

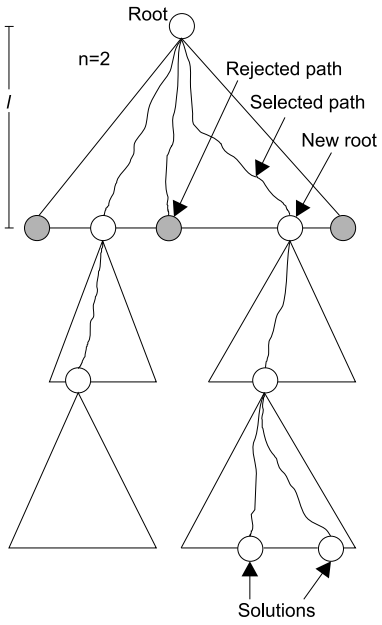


Fig. 5. Coverability tree pruning heuristic

In the case being studied, various combinations of these parameters have been tested. In Section 5, the influence of the parameters in the search over the coverability tree is discussed.

4 OPTIMISATION TOOL CHARACTERISTICS

To generate the coverability tree and to find a solution for a given problem, the PNPLAN tool (Gambin et al. [1999], Gambin [1999], Piera and Gambin [1998]) has been used. This tool reads the model, parses it for possible missing elements and syntax errors, and generates the coverability tree, using the heuristic defined in the previous section in order to prune the tree and reach a quasi-optimal solution.

A. Input Data

There are four elements which must be defined before the optimization can begin. All of them are passed to the program as a text file:

- The Petri net
  This includes the places, transitions, and the input and output weight functions for the arcs connecting them. These functions can be either constants or functions depending on the entries to the transition (see Section 2-D).
- The initial and final markings
  These indicate the initial state (how many tokens and where are they placed) of the system, and the objective state after processing the material and obtaining the results. The user has the possibility of leaving some indeterminate nodes in the final marking, corresponding to places which are unimportant for the definition of the final marking. Note that these markings determine the root node and the leaves of the coverability tree.
- The weights for each place
  These weights are used by the program to calculate the cost of every (intermediate) opened marking. A value of W_p for place p means that, when applying the cost function, the value added by that place is $W_p \cdot T_p$, where T_p is the number of tokens in place p at that moment. Using these values, the user can give higher penalty to the misuse of a shared resource by a machine, the selection of one path instead of another faster path, etc.
- The number of levels and paths
  These form the second part of the heuristic which can be defined by the user. By changing these parameters it is possible to control the quality of the solutions found. The pruning of the tree depends directly on the depth reached and the number of stored paths in every step (see Section 3-B).

B. Output Data

Once the coverability tree is complete, the program returns text files containing the fired transitions sorted by time. If there is no solution (or no solution has been found), no files are generated, and if there are multiple solutions, a maximum of t files[5] are returned.

Also, since the user has the possibility of leaving some indeterminate places in the final marking, the program also returns the final marking with all places now defined.

5 RESULTS

In order to evaluate the sensitivity of the proposed methodology in front of parameter value changes, this section summarizes the most important results obtained.

A. Precision of the Petri Net Representation

Table IV summarizes the results obtained when discretising the continuous dynamic of the boiler according to different amounts of mix. An experiment was performed to obtain 300 ml. of 0.2 ethanol concentration, given 4500 ml. of 0.7 ethanol concentration. The amount of liquid consumed together with the production time needed to achieve the final product is shown in Table IV for different discretisations: 1 token means 100 ml., 2 tokens means 200 ml., and so on.

As it could be expected, the lower precision is chosen in discretisation of the continuous system dynamics, to more time is spent in achieving the final marking and more mix is consumed.

[5] t is the parameter which represents the number of sub-trees stored in every step (see Section 3-B).

Prec. [ml]	Opt.Time	Cons.liq. [ml]	t	l
100	12064.6	2600	8	8
200	12064.6	2600	8	8
500	17020.6	3000	8	8
900	12366.8	2700	8	8
1500	17020.6	3000	8	8
4500	20735.5	4500	8	8

TABLE IV

RESULTS FOR DIFFERENT SYSTEM MODELLING PRECISION

B. Objective Function Weights

By using different scheduling policies it is possible to find different suitable solutions for specific requirements. In order to achieve the best solution the weights for every place were modified.

Experiment: Given 20 litres (200 tokens) of liquid with concentration 0.7, it is desired to find the best scheduling policy to achieve 1 litre (10 tokens) of 0.2 ethanol concentration.

The following policies have been considered:

1) Penalize the use of the five-litre boiler.
   By raising the weight of the places corresponding to liquid in the five-litre boiler (places from P_{29} to P_{34}), it can be observed in the solutions found that the liquid is moved to the two-litre boiler just after it has been heated (using transition T_{25}).
2) Penalize the use of the two-litre boiler.
   Similar to the previous policy. In this case, the penalized places are those corresponding to the two-litre boiler (places from P_{41} to P_{46}). Because of the experiment setup and the heating speed of both boilers, this policy gave the optimal solution.
3) Balanced use of boilers.
   By giving similar values to the weights representing both boilers, more balanced solutions are found. As was mentioned in the previous policy, the two-litre boiler is slower than the five-litre boiler. Thus, this solution is worse than the previous one. In a case with higher quantities of liquid to process, the parallelization would make the solution better. Figure 6 presents a Gantt diagram for this policy.

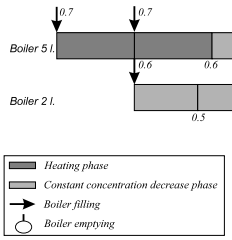


Fig. 6. Gantt diagram for a balanced policy.

4) Fast production (by penalizing the objective product).
   The processed mix is collected (in the exit tanks) at the shortest time. To satisfy this target, the places with the target tokens are penalized (places from P_{47} to P_{52}). In the solution, it can be observed that instead of using the maximum capacity of the five-litre boiler, the liquid is poured in smaller quantities in order to produce the objective liquid more quickly.

The results obtained when applying these policies is summarized in Table V.

Policy	Opt. Time	t	l
1	71918.0	8	8
2	41471.1	8	8
3	41569.5	8	8
4	67344.1	8	8

TABLE V

RESULTS FOR DIFFERENT POLICIES

C. Pruning Parameters

The parameters t and l determine the size of the coverability tree (see Section 3-B).

Experiment: Given 10 litres (100 tokens) of liquid with concentration 0.7, we seek to achieve 200 millilitres (2 tokens) of liquid with concentration 0.4, 200 millilitres (2 tokens) of liquid with concentration 0.3 and 300 millilitres (3 tokens) of liquid with concentration 0.2.

The combination of different values for t and l yielded the following results:

t\l	1	5	10	15	20
1	-	52875.1	52875.1	40543.9	52875.1
3	-	51407.3	45168.2	39608.4	52875.1
5	-	50207.0	40944.8	39608.4	52758.8
7	-	50207.0	40944.8	31342.1	43564.8
9	-	50207.0	27390.5	31342.1	43564.8
11	-	50207.0	27252.7	31342.1	43564.8

TABLE VI

RESULTS FOR DIFFERENT NUMBER OF PATHS t AND DEPTHS l

Note that the horizontal lines '-' mean that no solution has been found.

It can be observed in Table VI that the best results are obtained with a depth of 10. In this case, the different opened trees do not contain a full cycle. This means that they do not include the transitions from the pouring of liquid into the five-litre boiler to the collection of the processed liquid in the output tanks.

On the other hand, the greater the number of paths stored in memory, the better is the solution obtained. This holds until a certain number of paths, when the solutions start to be repeated, and the memory overhead becomes a liability.

6 CONCLUSIONS

In this paper, a methodology to optimize a batch production system based on the translation of state-events into time-events (see Section 1) has been presented.

A formalization of time-events into Petri nets allows the use of tools like the coverability tree. This gives the possibility of use both heuristics in the construction of the tree and external techniques like Tabu Search and Simulated Annealing.

Although such tools are usually used to optimize discrete systems, the methodology shows the steps to convert a continuous system into its corresponding discrete system (see Section 2).

The way the tool optimizes the Petri net has also been presented. It generates the coverability tree (see Section 3) and because of the need for a fast solution (instead of the optimal) the tree is pruned by the use of a heuristic (see Section 3-B) set up by the user and by other techniques related to the Petri net definition.

Using this methodology it is possible to optimize some types of continuous systems such as those from the manufacturing industry, and find a quasi-optimal solution by setting up some parameters. The tool used for the optimization takes the Petri nets-based model and initial and objective states, and finds a set of transitions firings which minimize the time (of the last transition fire) and a cost function based on the use of the system resources. From the transition sequence, the optimal policy can be derived.

References

P. A. Bahri, T. Gu, and P. Lee. Petri-net based modelling and scheduling of batch processing plants - a heuristic algorithm. *International Symposium on Advanced Control of Chemical Processes*, June 2000.

P. Fishwick. *Simulation Model Design and Execution*. Prentice Hall, 1995.

A. J. Gambin. Estado del arte del kernel para simulación de redes de petri. Internal report, Departament de Telecomunicació i Enginyeria de Sistemes. UAB, 1999.

A. J. Gambin, M. A. Piera, and D. Riera. A petri nets based object oriented tool for the scheduling of stochastic flexible manufacturing systems. In *7th IEEE International Conference on Emerging Technologies and Factory Automation*, pages 1091–1098, Barcelona, Spain, 1999.

T. Gu and P. Bahri. Timed petri-net representation for short term scheduling of multi-product batch plants. *Proceedings of the 1999 American Control Conference*, pages 4092–4096, 1999.

T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of IEEE*, 77(4):541–580, April 1989.

M. A. Piera and A. J. Gambin. Object oriented stochastic petri nets simulator: A generic kernel for heuristic planning tools. *LNAI 1415, Springer-Verlag*, pages 244–253, 1998.

L. Popova-Zeugmann. Petri nets with time restrictions. *Journal of System Analysis Modelling Simulation*, 13:13–20, 1993.

J. M. Proth and X. Xie. *Petri Nets. A Tool for Design and Management of Manufacturing Systems*. Wiley, 1996.

M. Silva. *Las Redes de Petri: en la Automática y la Informática*. AC, Madrid, 1985.

M. Silva and R. Valette. Petri nets and flexible manufacturing. *LNCS. Advances in Petri Nets*, 424:374–417, 1989.

Daniel Riera i Terrén studied Computer Science at the Universitat Autònoma de Barcelona (UAB). He has held a teaching position at the UAB and an FPI research grant from the Spanish government. After two six-month study periods at the ICParc, in the Imperial College of London, he is currently preparing his PhD dissertation. His research areas include automation, modelling and simulation, and industrial processes. He is research fellow in the UAB, teacher in the Universitat Oberta de Catalunya (UOC) and researcher of the LOGISIM.

Mercedes Narciso Farias studied Systems Engineering at the University of Los Andes, Venezuela. After working several years in a private company (consultancy and development of software in direction positions), she worked as researcher in the Computer Science Department of the University of Los Andes. In 1998 received the degree of Magister in Computer Science in this university. From 1999 she develops research works in the Universitat Autònoma de Barcelona, where she is associated professor in the Department of Telecommunications and Systems Engineering, and prepares her PhD dissertation on a tool for search and optimisation problems.

Chouaib Benqlilou Benkiran received his degree in physics from the University of Meknes, Morocco. Holds a Master's degree in Computer Science from UAB University, Spain. He is finishing his PhD at the Chemical Engineering Department of UPC University, Spain. His main research interests include design and upgrade of instrumentation process plant monitoring and process plant dynamic data reconciliation. Over the 5 past years he has worked at UPC on a broad variety of international projects related to Computer Aided Process Engineering and Process Plant data Management.