Available online at www.sciencedirect.com
ScienceDirect
Mathematics and Computers in Simulation xxx (2010) xxx–xxx
MATHEMATICS AND COMPUTERS IN SIMULATION
www.elsevier.com/locate/matcom

Original Articles

Coloured Petri net scheduling models: Timed state space exploration shortages

M.A. Piera[a], G. Mušič[b,*]

[a] Autonomous University of Barcelona, Dept. of Telecommunication and Systems Engineering, Barcelona, Spain
[b] University of Ljubljana, Faculty of Electrical Engineering, Trzaska 25, SI-1000 Ljubljana, Slovenia

Received 28 January 2010; accepted 14 October 2010

Abstract

The paper deals with the problem of timed state space generation and exploration in the frame of simulation–optimization approach for discrete-event systems. Coloured Petri net representation of a system is considered and corresponding techniques of timed state space generation and timed simulation are addressed. It is shown that the established simulation techniques do not perform adequately in some application relevant examples since in general, only a subset of a timed state space of a simulated system is represented. Two examples are provided to illustrate the effect of timed state space reduction. While the optimal solution is preserved within the reduced state space in one example, in the second example this is not the case and the optimum is missed. This indicates that the timed simulation technique has to be carefully designed in order to be suitable for the simulation–optimization approach.
© 2010 IMACS. Published by Elsevier B.V. All rights reserved.

Keywords: Petri nets; Manufacturing systems; Simulation; Optimization

1. Introduction

The quest for increased productivity, flexibility and competitiveness in present manufacturing industries, has forced new production policies to improve the customer quality of service which consider not only the quality and the price of the final product, but also time-to-market aspects. Modelling and performance evaluation play a vital role in the operational decision making activity, to deal with the best configuration to satisfy customer orders. Unfortunately, the required flexibility to improve the key performance indicators leads to a considerable amount of decision variables that must be properly evaluated in order to deal with the best possible operation scenarios. The role of performance modeling is to aid the decision making in an effective way.

Coloured Petri nets (CPNs) are a powerful framework for discrete-event modelling, simulation and analysis. In contrast to most system description languages, CPNs are state and action oriented at the same time—providing an

* Corresponding author.
E-mail addresses: MiquelAngel.Piera@uab.es (M.A. Piera), gasper.music@fe.uni-lj.si (G. Mušič).

0378-4754/$36.00 © 2010 IMACS. Published by Elsevier B.V. All rights reserved.
doi:10.1016/j.matcom.2010.10.014

explicit description of both the states and the actions [4]. Advantages of ordinary Petri nets, such as efficient modelling of concurrency and synchronization, and simple representation of production systems' specific properties, such as conflicts, deadlocks, limited buffer sizes, and finite resource constraints [9,10], are enhanced by a compact representation of the model and the underlying state space by the use of coloured tokens and hierarchy. Furthermore, a functional programming language CPN ML is added, which is used for the net inscriptions, i.e., text strings attached to the places, transitions and arcs of a CPN. This way various declarations are made and the language also facilitates modelling of data manipulation that is triggered by event occurrences. CPNs have been applied in a wide range of application areas, and many projects have been carried out in industry.

In order CPN models could help in making decisions related to finding the best routes, predicting the effects of adding or withdrawing resources and parts, or obtaining optimal schedules, it is necessary to consider the time aspects of each activity. A rich variety of extensions of the original PN and CPN formalism include the time concept, considering both deterministic and stochastic time [11]. In this way PNs and CPNs can be applied for simulation-based performance analysis, i.e., timed simulation can be performed for testing the performance of the system under certain operating conditions.

The ability of performance measures shows the possibility to use CPN models within the simulation–optimization approach as a tool to solve real industrial scheduling problems. Within this approach, an optimization algorithm can be used to arrange the simulation of a sequence of system configurations so that an optimal or near optimal system configuration could eventually be obtained [7,8].

In order to be able to reach the optimum, it is important to be able to generate any possible trace of the system behaviour. It can be observed that in most of the discrete-event simulation tools this is not always the case. They are generally able to represent only a subset of timed state space of a simulated system. The paper deals with the critical evaluation of the timed state space generation strategies found in DES simulators. In particular CPN framework and the related time mechanisms are studied.

The remainder of the paper is structured as follows. The timed state space exploration techniques are introduced in Section 2. The application of the presented techniques to optimization of schedules is described in Section 3. A simple manufacturing example is presented, where the differences in the timed state space exploration techniques are examined. Another example is given in Section 4 to illustrate the effect of timed state space exploration technique when searching for a solution to a scheduling problem.

2. Timed state space exploration

For simplicity, a place/transition (P/T) timed Petri net will be considered in the following. It can be described as a bipartite graph consisting of two types of nodes, places and transitions. The nodes are interconnected by directed arcs. The state of the system is denoted by the distribution of tokens (called marking) over the places. Compared to CPN, no token colours nor arc inscriptions other than weight are used, and transition inscriptions are limited to specification of time delay only. This allows more focused presentation of time aspects while the mechanism of time inclusion is kept as close as possible to the one of CPNs. This enables a straightforward extension of presented concepts to CPNs.

The concept of time is not explicitly given in the original definition of Petri nets. As described in [3], there are three basic ways of representing time in Petri nets: firing durations (FD), holding durations (HD) and enabling durations (ED). The FD principle says that when a transition becomes enabled it removes the tokens from input places immediately but does not create output tokens until the firing duration has elapsed. In [12] a well-defined description of this principle is given. When using HD principle, a firing has no duration but a created token is considered unavailable for the time assigned to transition that created the token. The unavailable token cannot enable a transition and therefore causes a delay in the subsequent transition firings. This principle is graphically represented in Fig. 1, where the available tokens are schematized with the corresponding number of undistinguishable (black) tokens and the unavailable tokens are indicated by empty circles. The time assigned to a transition is written beside the transition, e.g., d_1 is assigned to transition t_1. When the time is 0 this denotation is omitted. In Fig. 1, τ denotes a model time represented by a global clock and τ_f denotes the firing time of a transition.

HD and FD are in fact the same way of representing time while ED principle leads to a different timed behaviour of a model. With ED, the firing of the transitions has no duration while the time delays are represented by forcing transitions that are enabled to stay so for a specified period of time before they can fire. The main difference between

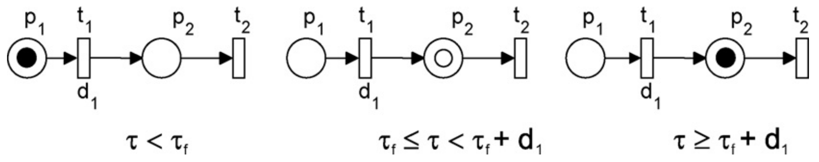


Fig. 1. Timed Petri net with holding durations.

using HD and ED can be seen in a Petri net where a conflict appears. In this case more transitions are enabled by one marking. When ED policy is used, the firing of one transition can interrupt the enabling of other transitions, as the marking, which has enabled previous situation, has changed [3].

This indicates that ED concept is more general than HD. Furthermore, in [6] an even more general concept is used, which assigns delays to individual arcs, either inputs or outputs of a transition. This way both ED and HD concepts are covered, and the enabling delay may even depend on the source of transition triggering while holding delay may differ among different activities started by the same transition. This paper builds on some ideas presented in [6] although they are used in a different context. While the primary focus in [6] is on a development of a decentralized timed state space generation approach, here the formalization is simplified and the focus is on the practical implications of using various types of timed state spaces.

When modelling several performance optimization problems, e.g., scheduling problems, such a general framework as presented in [6] is not needed. It is natural to use HD when modelling most scheduling processes as transitions represent starting of operations, and generally once an operation starts it does not stop to allow another operation to start in between. HD principle is also used in timed version of CPNs, although the unavailability of the tokens is only defined implicitly through the corresponding time stamps. While CPNs allow the assignment of delays both to transition and to output arcs, we further simplify this by allowing time delay inscriptions to transitions only. This is sufficient for the type of examples investigated here, and can be generalized if necessary.

2.1. P/T timed Petri net with holding durations

To include a time attribute of the marking tokens, which implicitly defines their availability and unavailability, the notation of [4] will be adopted. Tokens are accompanied with a timestamp, which is written next to the token number and separated from the number by @. E.g., two tokens with time stamp 10 are denoted 2@10. A collection of tokens with different time stamps is defined as a multiset, and written as a sum (union) of sets of timestamped tokens. E.g., two tokens with time stamp 10 and three tokens with timestamp 12 are written as 2@10+3@12. The timestamp of a token defines the time from which the token is available.

Time stamps are elements of a time set TS, which is defined as a set of numeric values. In many software implementations the time values are integer, i.e., $TS = \mathbb{N}$, but will be here admitted to take any positive real value including 0, i.e., $TS = \mathbb{R}_0^+$. Timed markings are represented as collections of time stamps and are multisets over TS: TS_{MS}. By using HD principle the formal representation of a P/T timed Petri net is defined as follows. $TPN = (\mathcal{N}, M_0)$ is a timed Petri net system, where: $\mathcal{N} = (P, T, I, O, f)$ is a Timed Petri net structure, $P = \{p_1, p_2, \ldots, p_k\}$, $k > 0$ is a finite set of places, $T = \{t_1, t_2, \ldots, t_l\}$, $l > 0$ is a finite set of transitions (with $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$), $I : (P \times T) \to \mathbb{N}$ is the input arc function. If there exists an arc with weight k connecting p to t, then $I(p, t) = k$, otherwise $I(p, t) = 0$. $O : (P \times T) \to \mathbb{N}$ is the output arc function. If there exists an arc with weight k connecting t to p, then $O(p, t) = k$, otherwise $O(p, t) = 0$. $f : T \to TS$ is the function that assigns a non-negative deterministic time delay to every $t \in T$. $M : P \to TS_{MS}$ is the timed marking, M_0 is the initial marking of a timed Petri net.

Functions I and O define the weights of directed arcs, which are represented by arc inscriptions. In the case when the weight is 1, this annotation is omitted, and in the case when the weight is 0, the arc is omitted. Let ${}^\bullet t \subseteq P$ denote the set of places which are inputs to transition $t \in T$, i.e., there exists an arc from every $p \in {}^\bullet t$ to t.

To determine the availability and unavailability of tokens, two functions on the set of markings are defined. The set of markings is denoted by $\mathbb{M}$. Given a marking and model time, $m : P \times \mathbb{M} \times TS \to \mathbb{N}$ defines the number of available tokens, and $n : P \times \mathbb{M} \times TS \to \mathbb{N}$ the number of unavailable tokens for each place of a TPN at a given time τ_k. Note that model time also belongs to time set TS, $\tau_k \in TS$.

Two timed markings can be added (denoted $+_\tau$) in a similar way as multisets, i.e., by making a union of the corresponding multisets. The definition of subtraction is somewhat more problematic. To start with, a comparison operator is defined. Let M_1 and M_2 be markings of a place $p \in P$. By definition, $M_1 \geq_\tau M_2$ iff $m(p, M_1, \tau_k) \geq m(p, M_2, \tau_k)$, $\forall \tau_k \in TS$.

Similarly, the subtraction is defined by the number of available tokens, and the subtrahend should not contain any unavailable tokens. Let M_1, M_2 and M_3 be markings of a place $p \in P$, $M_1 \geq_\tau M_2$, and $m(p, M_1, \tau_k)$, $m(p, M_2, \tau_k)$, and $m(p, M_3, \tau_k)$, be the corresponding numbers of available tokens at time τ_k, and $n(p, M_2, \tau_k) = 0$. The difference $M_3 = M_1 -_\tau M_2$ is then defined as any $M_3 \in \mathbb{M}$ having $m(p, M_3, \tau_k) = m(p, M_1, \tau_k) - m(p, M_2, \tau_k)$.

Clearly, the subtraction is not uniquely defined this way, since it is not clear, which of the available tokens, eventually having different timestamps, will be removed from the minuend. However, the subtraction is only used to define removal of tokens from input places when a transition is fired. If the HD principle is used and there are several available tokens in an input place with different timestamps, it is not important which of them is removed. All the subsequent transition firings only deal with present or future points in time, and all currently available tokens remain available for the eventual enablement of future firings. The situation would be changed, of course, if the ED timing principle was used, since there also the past is important for the enablement of transitions. For such a case, the subtraction operator is defined more generally in [6].

Even when applying HD principle, it can however be practical to uniquely define the subtraction operation on timed markings. In this paper the comparison of each newly generated marking to all previously reached markings is applied when generating timed state space. This is simplified if the new markings are always generated in the same way, e.g., by always removing the token with the most recent timestamp first.

Using the above definitions, the firing rule of a TPN can be defined. Given a marked $TPN = (\mathcal{N}, M)$, a transition t is time enabled at time τ_k, denoted $M[t\rangle_{\tau_k}$ iff $m(p, M, \tau_k) \geq I(p, t)$, $\forall p \in {}^\bullet t$. An enabled transition can fire, and as a result removes tokens from input places and creates tokens in output places. If transition t fires, then a new marking is given by $M'(p) = M(p) -_\tau I(p, t)@\tau_k +_\tau O(p, t)@(\tau_k + f(t))$, $\forall p \in P$. If marking M_2 is reached from M_1 by firing t at time τ_k, this is denoted by $M_1[t\rangle_{\tau_k} M_2$. The set of markings of TPN $\mathcal{N}$ reachable from M is denoted by $R(\mathcal{N}, M)$.

2.2. Classes of timed state spaces

In the definition of the firing rule of a TPN, the enabled transition is described as a transition that can fire. This is a common definition of the firing rule in the Petri net literature. Nothing is said about the exact moment of firing. In the timed state space generation, time of the firing is a key attribute. Depending on a more detailed definition of the time of transition firing one can distinguish among several classes of timed state spaces.

Most general timed state space [6] is TSS = (N, A), where $N = R(\mathcal{N}, M_0)$ is a node set and A is the set of arcs given as $A = \bigcup_{M \in N} \{(M, t, M')_{\tau_k} | M[t\rangle_{\tau_k} M'\}$. In such a TSS a firing $M[t\rangle_{\tau_k} M'$ is not tied to any specific firing time τ_k. This can be any time greater or equal to τ_0 when transition becomes enabled.

To better define the firing time it may be required that a transition fires at the earliest possible time. This way an earliest time state space is defined: ESS = (N, A), where $N = R(\mathcal{N}, M_0)$ and A is given as $A = \bigcup_{M \in N} \{(M, t, M')_{\tau_k} | M[t\rangle_{\tau_k} M', \nexists \tau_1 < \tau_k, : M[t\rangle_{\tau_1}\}$.

In ESS a firing $M[t\rangle_{\tau_k} M'$ is tied to the earliest firing time at a given marking. But this does not prevent the inclusion of other transitions from the same marking into the ESS, which become enabled and fire at later times. E.g., if two transitions t_1 and t_2 are in conflict and t_1 is enabled before t_2, also t_2 participates in the ESS.

The above definitions of TSS and ESS are somewhat anomalous, since only the time of transition enablement is considered, while the time when a certain marking is reached is not taken into account. In certain situations this allows subsequent firings to occur in TSS or ESS at a time smaller than the time of the last firing.

For example, consider the net in Fig. 2. Places p_1 and p_2 are marked by a token each, but with different timestamps. Transitions t_1 and t_2 have time delay 2 and 1, respectively, which is denoted beside transitions using standard CPN notation (@ + delay). Transition t_1 is enabled before t_2 but nevertheless, according to the above definition of ESS, both firing sequences are considered: t_1, t_2 and t_2, t_1. Furthermore, from the marking M_2 obtained after firing t_2 at a time $\tau_{21} = 5$, t_1 can be fired at time $\tau_{12} = 2$, i.e., $\tau_{12} < \tau_{21}$. A corresponding part of the state transition diagram of ESS is shown in Fig. 3a.

To avoid such anomalies, the definition of TSS and ESS is changed in a way that a marking creation time is adjoined to every node of the state space.

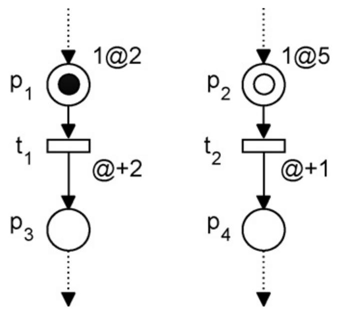


Fig. 2. Example of a timed Petri net.

The node set of TSS is then $N \subset R(\mathcal{N}, M_0) \times TS$, and $N_i = (M_i, \tau_i)$, where τ_i is the time when the last transition in a sequence leading from M_0 to M_i was fired. The corresponding arc set is $A = \bigcup_{N_j \in N}\{(N_j, t, N_k)_{\tau_k} | N_j = (M_j, \tau_j), N_k = (M_k, \tau_k), M_j[t\rangle_{\tau_k} M_k\}$. When exploring further evolution from the marking M_j, in contrast to [6], the creation time τ_j of M_j is taken into account. Therefore τ_k can be any time greater or equal to τ_0 when transition becomes enabled, and τ_0 itself must be greater or equal to τ_j: $\tau_k \geq \tau_0 \geq \tau_j$, $M[t\rangle_{\tau_0}$, $\nexists \tau_1 : \tau_j \leq \tau_1 < \tau_0$, $M[t\rangle_{\tau_1}$.

Similarly, the ESS is redefined as ESS = (N, A), where $N \subset R(\mathcal{N}, M_0) \times TS$ and A is given as $A = \bigcup_{N_j \in N}\{(N_j, t, N_k)_{\tau_k} | N_j = (M_j, \tau_j), N_k = (M_k, \tau_k), M_j[t\rangle_{\tau_k} M_k, \nexists \tau_1 : \tau_j \leq \tau_1 < \tau_k, M[t\rangle_{\tau_1}\}$.

For the net in Fig. 2 this results in a different ESS. Part of the corresponding state transition diagram is shown in Fig. 3b. Clearly, the nodes N_3 and N_4 are not equivalent.

The possibility that two transitions in conflict, which are enabled at different times both participate in the timed state space generation is eliminated by reduced earliest time state space RSS. This class of timed state space only allows the inclusion of transitions in conflict that can fire at the same time. It is defined as RSS = (N, A), where $N \subset R(\mathcal{N}, M_0) \times TS$ and A is given as $A = \bigcup_{N_j \in N}\{(N_j, t, N_k)_{\tau_k} | N_j = (M_j, \tau_j), N_k = (M_k, \tau_k), M_j[t\rangle_{\tau_k} M_k, \nexists t_1, \tau_1 : \tau_j \leq \tau_1 < \tau_k, M[t_1\rangle_{\tau_1}\}$.

For a P/T timed Petri net the three classes of time state spaces are related as follows: RSS $\subseteq$ ESS $\subset$ TSS. E.g., the RSS for the net in Fig. 2 results in a subgraph of ESS in Fig. 3b, where only the nodes N_0, N_1 and N_3 are included.

Most of the timed state space generating algorithms found in various software tools actually generate RSS. An example is CPN Tools software [1,5].

Next, algorithms for ESS and RSS generation are briefly sketched. For simplicity, the algorithms will be presented for P/T timed Petri nets only. The extension to CPNs is straightforward.

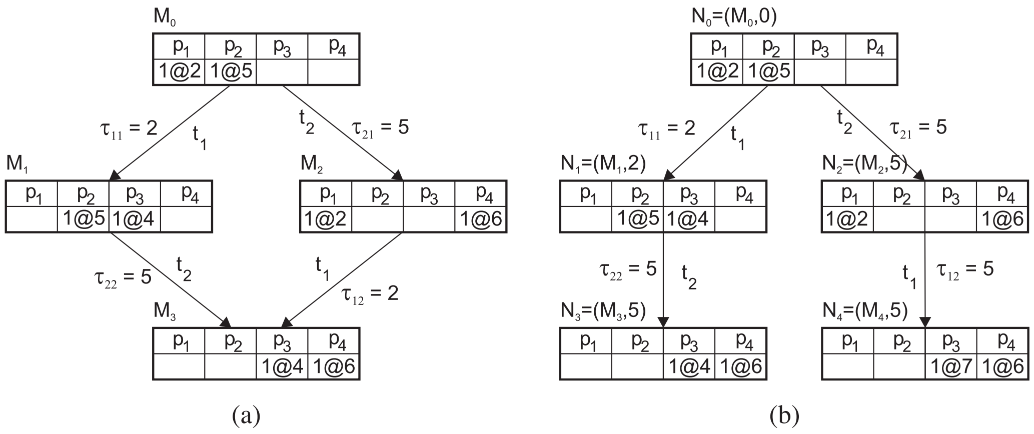


Fig. 3. Two possible ESSs of the example.

Algorithm 1. ESS generation

```
set($max\_time$);
ESS.$N$:=$\{(M_0, 0)\}$;
ESS.$A$:=$\varnothing$;
$U$:=$\{(M_0, 0)\}$;
$\tau$:=0;
while $U \neq \varnothing$ do
  for all $N \in U$ do
    $\tau$:= creation_time($N$);
    $M$:= marking($N$);
    repeat
      for all $t \in T : M[t\rangle_\tau M' \wedge \nexists \tau' < \tau : M[t\rangle_{\tau'}$ do
        $N'$:=$(M', \tau)$;
        if $N' \notin$ ESS.$N$ then
          ESS.$N$:= ESS.$N \cup \{N'\}$;
          $U$:=$U \cup \{N'\}$;
        end if
        ESS.$A$:= ESS.$A \cup \{(N, t, N')_\tau\}$;
      end for
      $\tau$:=next_time_hit($\tau$, $M$);
    until $n(p, M, \tau) = 0, \forall p \in P \bigvee \tau > max\_time$
    $U$:=$U - \{N\}$;
  end for
end while
```

The algorithm for generating ESS is shown in Algorithm 1. It is assumed that reached markings are stored with an additional attribute, i.e., the time of their occurrence. When comparing a new node to previously generated nodes of the state space, this attribute is also taken into account, i.e., two identical marking that were reached at different times belong to different nodes. Such an implementation is used in order to be able to distinguish behaviours that produce the same event (sub)sequence at different times. The function "creation_time(N)" returns a value of this attribute associated with node N while the function "marking(N)" returns the corresponding marking. The function "next_time_hit(τ, M)" returns the value of the lowest timestamp value in M greater than τ. Parameter max_time is used to define a time limit beyond which the transition firings are not explored.

The algorithm for generating RSS is shown in Algorithm 2. It differs in the way unavailable tokens are treated at the currently processed marking. The time is not incremented to wait for the tokens to become available at this point, but only after all the markings reachable in one step are determined. This way only the earliest transition(s) fireable from a given marking are kept. To enable this, the domain of function "next_time_hit(τ, M)" is extended from a single marking to the whole set of timestamps included in the set of markings related to unexplored nodes U.

Algorithm 2. RSS generation

```
set($max\_time$);
RSS.$N$:={($M_0$, 0)};
RSS.$A$:=$\varnothing$;
$U$:={($M_0$, 0)};
$\tau$:=0;
while $U \neq \varnothing \wedge \tau \leq max\_time$ do
  repeat
    $U'$:=$U$;
    for all $N \in U$ do
      $M$:= marking($N$);
      for all $t \in T$ : $M[t\rangle_\tau M'$ do
        $N'$:=($M'$, $\tau$);
        if $N' \notin$ RSS.$N$ then
          RSS.$N$:= RSS.$N \cup \{N'\}$;
          $U$:=$U \cup \{N'\}$;
        end if
        RSS.$A$:=RSS.$A \cup \{(N, t, N')_\tau\}$;
        $U$:=$U - \{N\}$;
      end for
    end for
  until $U = U'$
  $\tau$:= next_time_hit($\tau$, $U$);
end while
```

Simulation can be regarded as an exploration of a single path in the timed state space of the model. Most of the available simulation packages for TPNs or CPNs are based on RSS generation principle. Of course only one transition is chosen to fire at every simulation step. Clearly, this rules out some of the possible paths in the timed state space. When using simulation for performance optimization, this may cause the obtained solutions being not optimal. This will be demonstrated by examples in the following sections.

3. RSS policy applied to a scheduling problem

A well known job shop scheduling problem (JSSP), will be used to illustrate the shortages of using RSS policy for scheduling purposes. To solve a JSSP means to schedule a group of jobs in a set of machines, subject to the restriction that each machine can process only one job at a time and each job has an order of processing specified in each machine, and the JSSP belongs to a class of NP-hard problems.

There is a high number of predefined problem instances for different numbers of jobs and machines [2], which are usually considered by the optimization community as benchmark problems to test algorithms and combinatorial optimization methods in which the main goal is to minimize the time of completion of all jobs (makespan).

Recently, the JSSP has also been used by the simulation community to test simulation–optimization approach in which random variables are used to specify the different operation times. In [8] a state space exploration algorithm to tackle the state space explosion by means of heuristics and deal with the optimal solution is presented.

To illustrate the RSS policy shortages, a job shop production system with two machines and two jobs will be considered. Fig. 4 illustrates graphically the system in which Job 1 is represented by a ring and Job 2 is represented by a cylinder (transport subsystems are not considered relevant to the problem).

The machine sequence of each job is known (see Table 1), and each machine can serve only one job simultaneously. Fig. 5 illustrates the Petri Net model, in which: Transition **T1** is used to describe machine M1 processing Job 1. Transition **T2** is used to describe machine M1 processing Job 2. Transition **T3** is used to describe machine M2 processing Job 1. Transition **T4** is used to describe machine M2 processing Job 2. Place **M1** is used to describe machine M1 free. Place **M2** is used to describe machine M2 free. Place **J1O1** is used to describe Job 1 waiting for the first operation in machine M1. Place **J1O2** is used to describe Job 1 waiting for the second operation in machine M2. Place **J2O1** is used to describe Job 2 waiting for the first operation in machine M2. Place **J2O2** is used to describe Job 2 waiting for the second operation in machine M1. Place **Prod** is used to represent the processed jobs.

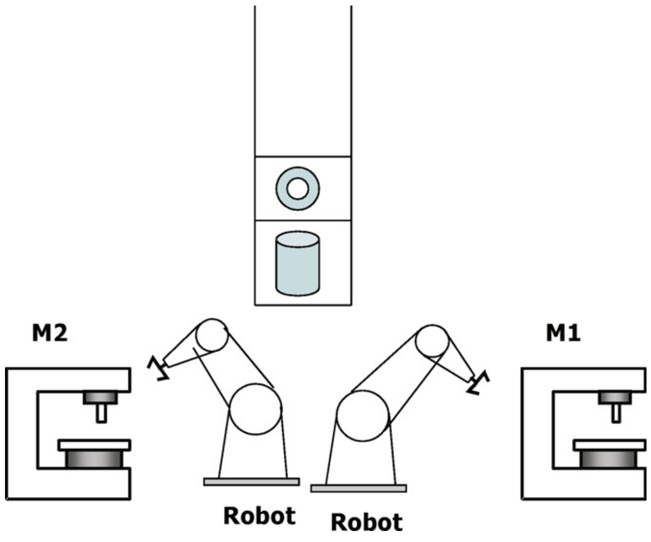


Fig. 4. Job shop 2×2.

Table 1
Machine sequence.

Job 1		Job 2	
Machine	Operation time	Machine	Operation time
1	3	2	3
2	6	1	3

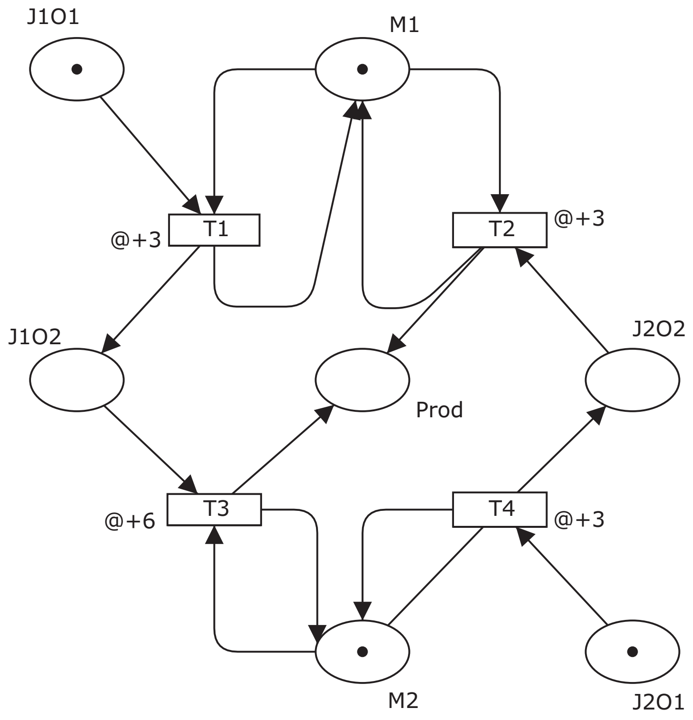
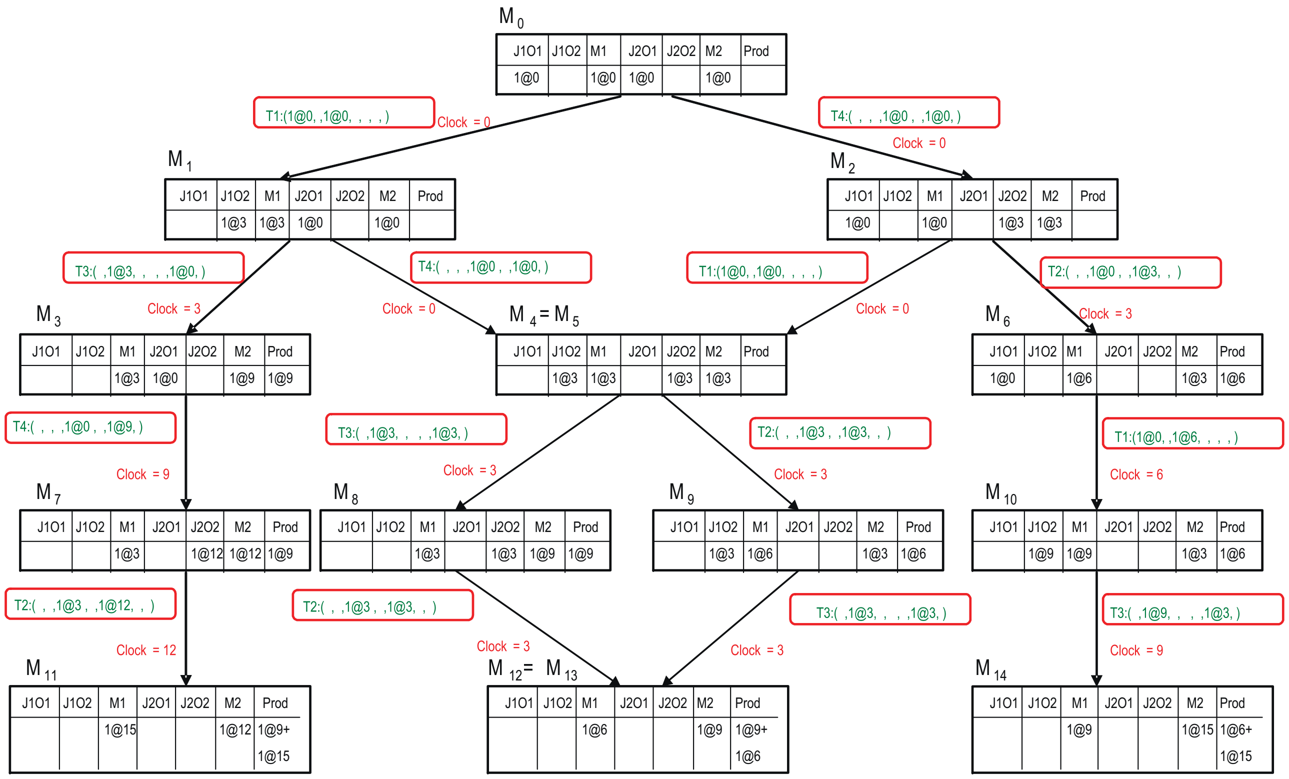


Fig. 5. Petri Net model for the 2×2 job shop problem.



M_0

J1O1	J1O2	M1	J2O1	J2O2	M2	Prod
1@0		1@0	1@0		1@0	

M_1

J1O1	J1O2	M1	J2O1	J2O2	M2	Prod
	1@3	1@3	1@0		1@0	

M_2

J1O1	J1O2	M1	J2O1	J2O2	M2	Prod
1@0		1@0		1@3	1@3	

M_3

J1O1	J1O2	M1	J2O1	J2O2	M2	Prod
		1@3	1@0		1@9	1@9

$M_4 = M_5$

J1O1	J1O2	M1	J2O1	J2O2	M2	Prod
	1@3	1@3		1@3	1@3	

M_6

J1O1	J1O2	M1	J2O1	J2O2	M2	Prod
1@0		1@6			1@3	1@6

M_7

J1O1	J1O2	M1	J2O1	J2O2	M2	Prod
		1@3		1@12	1@12	1@9

M_8

J1O1	J1O2	M1	J2O1	J2O2	M2	Prod
		1@3		1@3	1@9	1@9

M_9

J1O1	J1O2	M1	J2O1	J2O2	M2	Prod
	1@3	1@6			1@3	1@6

M_{10}

J1O1	J1O2	M1	J2O1	J2O2	M2	Prod
	1@9	1@9			1@3	1@6

M_{11}

J1O1	J1O2	M1	J2O1	J2O2	M2	Prod
		1@15			1@12	1@9+ 1@15

$M_{12} = M_{13}$

J1O1	J1O2	M1	J2O1	J2O2	M2	Prod
		1@6			1@9	1@9+ 1@6

M_{14}

J1O1	J1O2	M1	J2O1	J2O2	M2	Prod
		1@9			1@15	1@6+ 1@15

Fig. 6. State space (ESS) for the 2×2 job shop problem.

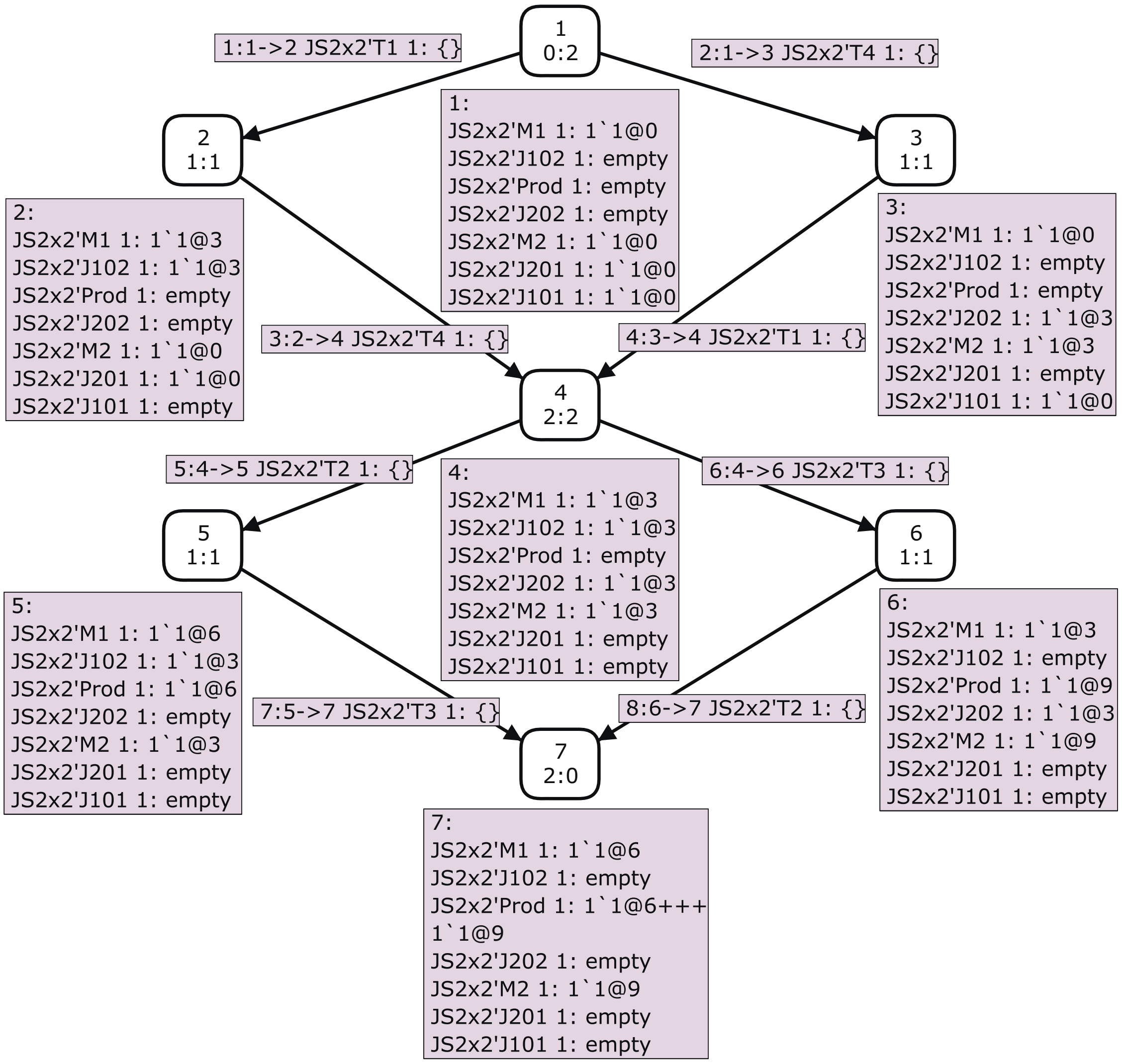


Fig. 7. RSS for the 2× 2 job shop problem.

According to the discrete event system representation proposed for the given job shop, the different plans to process the two jobs can be found in the state space graph (ESS) represented in Fig. 6 (nodes $\mathbf{M}_4$ and $\mathbf{M}_5$ are merged because they are equivalent, similarly $\mathbf{M}_{12} \equiv \mathbf{M}_{13}$). Node $\mathbf{M}_0$ represents the initial marking, and nodes $\mathbf{M}_{11}$, $\mathbf{M}_{12} = \mathbf{M}_{13}$ and $\mathbf{M}_{14}$ represent the final markings in which both jobs have been processed. As it can be easily seen, both jobs could be processed in 9 time units (node $\mathbf{M}_{12} = \mathbf{M}_{13}$) by taking the advantage that machine M1 and M2 can work in parallel (consider for example sequence **T1**, **T4**, **T3**, **T2**). However, a bad planning policy could lead a make-span of 15 time units (nodes $\mathbf{M}_{11}$, $\mathbf{M}_{14}$) in which all operations are performed sequentially (consider for example sequence **T1**, **T3**, **T4**, **T2**). In case a RSS policy would be used to analyze the state space of the given job shop system, marking $\mathbf{M}_3$ would not be reachable from marking $\mathbf{M}_1$. In a similar way, marking $\mathbf{M}_6$ would not be reachable from marking $\mathbf{M}_2$. Fig. 7 illustrates the corresponding RSS graph obtained by means of CPN tools software.

The node inscriptions in the state space graph within CPN Tools consist of a node number followed by a number of parent nodes and a number of child nodes. Next to the nodes are the node descriptors where the corresponding place markings are shown in the form ⟨ Page name⟩′⟨ Place name⟩ Instance: Marking. In the example shown in Fig. 7 the CPN was drawn in a page named JS2×2 and the place names are shown in Fig. 5. The place markings are written in a standard CPN notation as explained in Section 2.1. Next to the arcs are placed the arc descriptors showing the arc number, source and destination nodes and related transition.

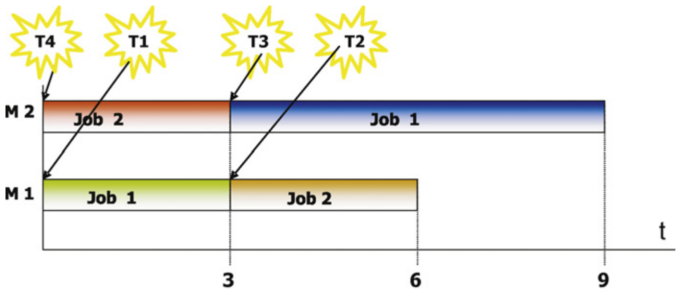


Fig. 8. Gantt representation of transition sequences 1–4.

Fig. 8 illustrates the related Gantt representation of the schedule obtained by firing the transition sequences: Sequence 1: **T4**, **T1**, **T3**, **T2** Sequence 2: **T1**, **T4**, **T3**, **T2** Sequence 3: **T4**, **T1**, **T2**, **T3** Sequence 4: **T1**, **T4**, **T2**, **T3**

All four sequences can be observed both using the ESS and the RSS policies. However, there are also some undesirable sequences (sequences 5 and 6) that could lead the production system to poor make-span, but can not be analyzed using the RSS policy because there is at least one time-enabled event that prevents to update the clock to the next colour-enabled event. Fig. 9 illustrates the Gantt representation of sequence 5: **T1**, **T3**, **T4**, **T2**.

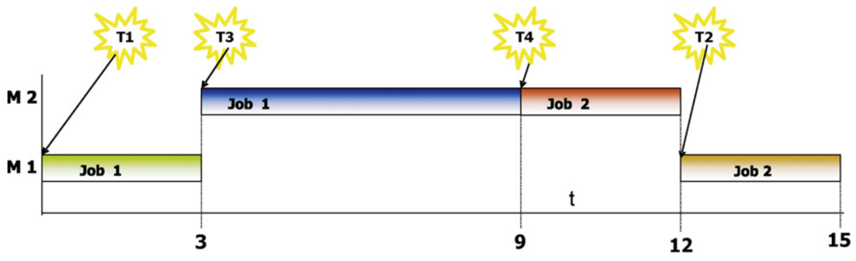


Fig. 9. Gantt representation of transition sequence 5.

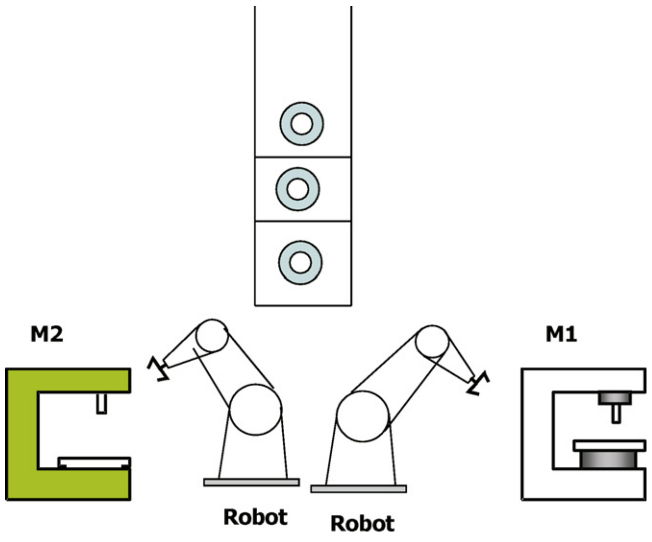


Fig. 10. Different performance production subsystems.

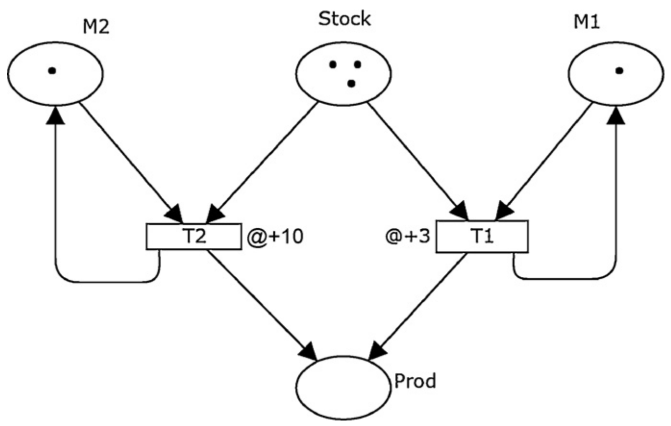


Fig. 11. Petri net of the production system.

4. Some shortages of RSS policy in the frame of simulation–optimization approach

The basic idea underlying the use of CPN models for performance analysis by means of dynamic generation of the states space, is to translate a problem of planning, scheduling or routing into a search problem in the states space of the system. An academical scheduling problem is presented in this section to show up that the RSS policy cannot be used to guarantee optimality because some states are missed.

Fig. 10 illustrates a production system with a stock of raw material and two machines (transport subsystems dynamics are neglected) with the same characteristics but with different performances: Machine M2 is an old generation machine that still works but with a rate 3/10th of a new CNC machine (M1). The problem is in establishing the optimal schedule to process 3 details using 2 machines (M1 and M2) in which the processing time of a detail in machine M1 is 3 time units while the processing time in machine M2 is 10 time units.

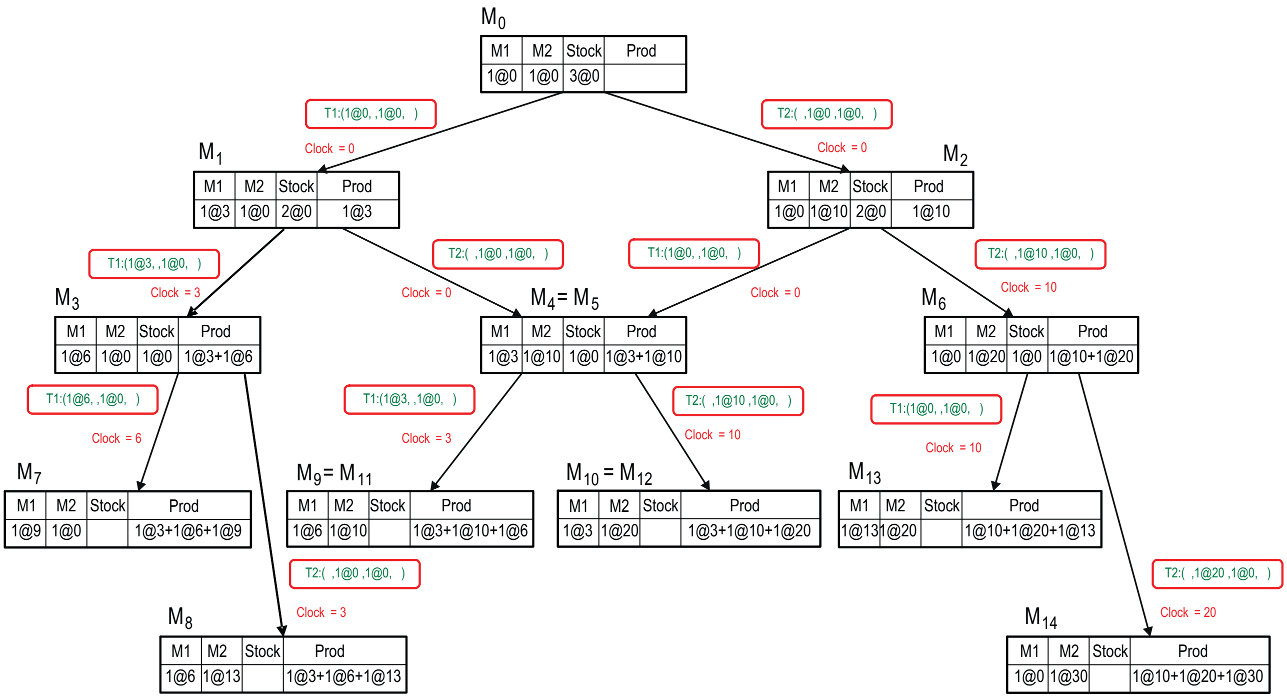


Fig. 12. State space (ESS) for the production system.

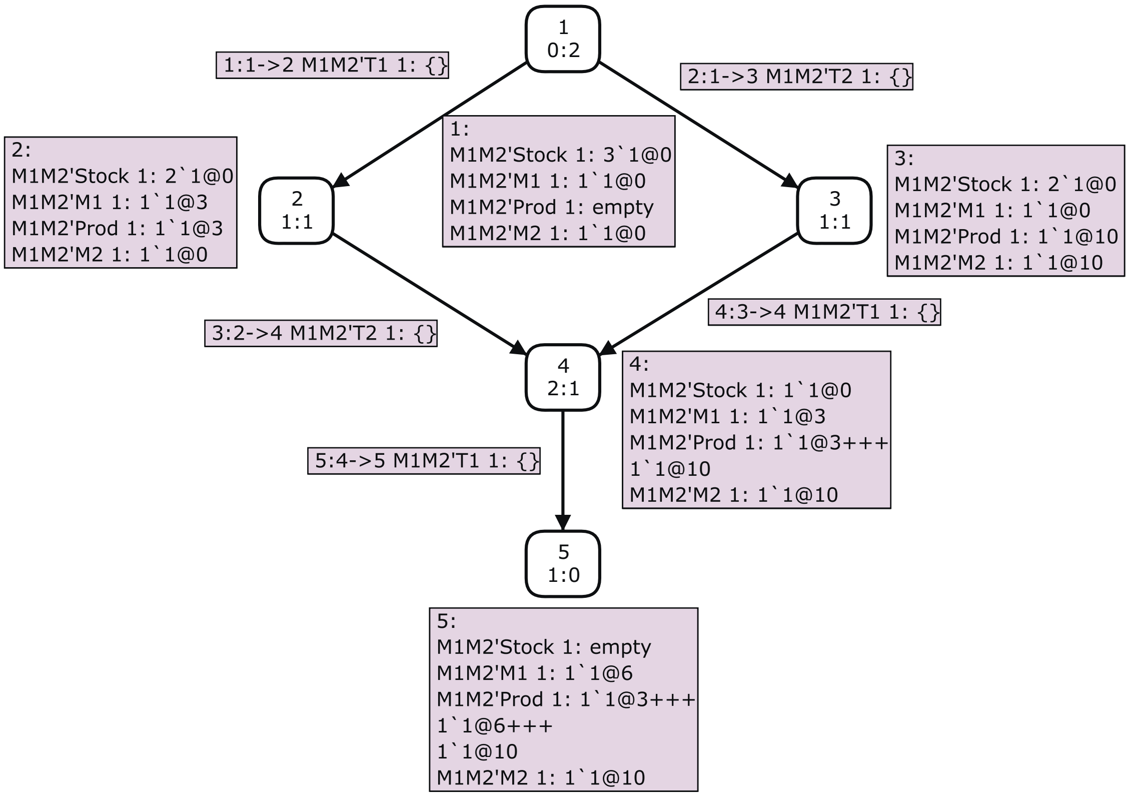


Fig. 13. RSS for the production system.

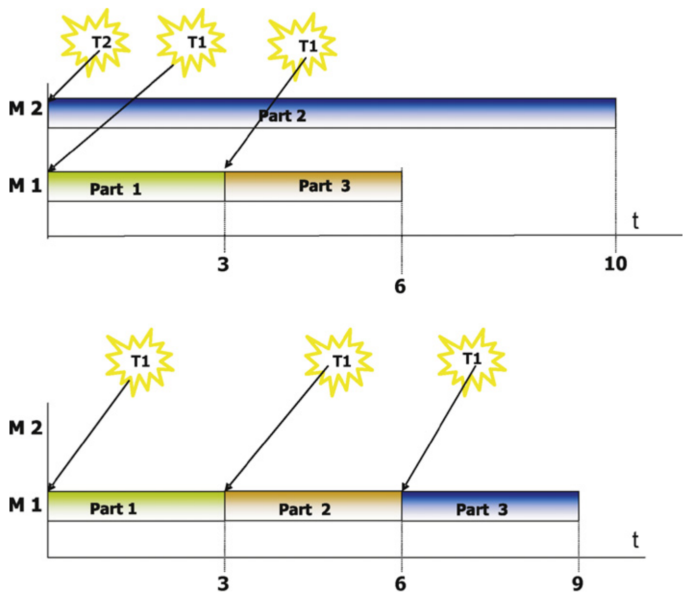


Fig. 14. Gantt representation of optimal sequences applying ESS and RSS policies.

Fig. 11 illustrates the Petri net model of the production system in which transition **T**1 represents machine M1 processing a detail while transition **T**2 represents machine M2 processing a detail.

Fig. 12 illustrates the state space (ESS) of the system in which it is easy to note that the optimal policy would assign the whole workload to machine M1 (sequence **T1**, **T1**, **T1**) obtaining a make-span of 9 time units (marking $\mathbf{M}_7$), while the worst policy requires 30 time units (marking $\mathbf{M}_{14}$) which is obtained by assigning the whole workload to machine M2 (sequence **T**2, **T**2, **T**2).

As it can be checked in Fig. 13, the use of reduced earliest time state space algorithms to analyze the state space of a CPN model, only can result in a policy that requires 10 time units to process the 3 details.

Fig. 14 shows in the lower part the Gantt representation of an optimal sequence that can not be reached using the RSS policy because transition **T**2 is enabled at time 0. In the upper part of the figure, the best feasible sequence that can be obtained using the RSS policy is illustrated also using a Gantt representation.

5. Conclusions and future work

When using state space analysis for optimization of discrete-event systems it is important to be able to generate any possible trace of the system behaviour. Corresponding software tools are often based on the reduced earliest time state space generation principle, which rules out some of the possible paths in the timed state space. Similarly, discrete event simulation is based on a similar principle, i.e., the scheduled event list is ordered on a smallest-scheduled-time-first basis and only the first event in the list occurs in the next simulation step. This way some of the possible behaviour scenarios are not explored and optimal solution can be missed. A simple illustrating example is presented in this paper. This indicates that the underlying simulation technique has to be carefully designed in order to be suitable for the simulation–optimization approach.

References

[1] CPN tools home page. http://people.brunel.ac.uk/~mastjjb/jeb/info.html.
[2] J. Beasley, OR-library. http://people.brunel.ac.uk/mastjjb/jeb/info.html.
[3] F.D.J. Bowden, A brief survey and synthesis of the roles of time in Petri nets, Mathematical and Computer Modelling 31 (2000) 55–68.
[4] K. Jensen, Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use, vol. 1, 2nd ed., Springer-Verlag, Berlin, 1997.
[5] K.L. Jensen, W.L. Kristensen, Coloured Petri nets and CPN tools for modelling and validation of concurrent systems, International Journal on Software Tools for Technology Transfer 9 (2007) 213–254.
[6] C. Lakos, L. Petrucci, Modular state space exploration for timed Petri nets, International Journal on Software Tools for Technology Transfer 9 (2007) 393–411.
[7] M. Mujica, M. Piera, M. Narciso, Optimizing time performance in reachability tree-based simulation, in: Proceedings of the 20th European Modeling & Simulation Symposium, Campora S. Giovanni, Amantea, CS, Italy, 2008.
[8] M. Piera, M. Narciso, A. Guasch, D. Riera, Optimization of logistic and manufacturing systems through simulation: a colored Petri net-based methodology, SIMULATION, Transactions of the Society for Modeling and Simulation International 80 (2004) 121–129.
[9] J. Proth, X. Xie, Petri nets. A Tool for Design and Management of Manufacturing Systems, John Wiley & Sons, Ltd., 1996.
[10] G. Tuncel, G.M. Bayhan, Applications of Petri nets in production scheduling: a review, International Journal of Advanced Manufacturing Technology 34 (2007) 762–773.
[11] N. Viswanadham, Y. Narahari, Performance Modeling of Automated Manufacturing Systems, Prentice-Hall, Englewood Cliffs, USA, 1992.
[12] W.M. Zuberek, Timed Petri nets: definitions, properties and applications, Microelectronics and Reliability 31 (4) (1991) 627–644.