

---

## Performance optimisation of a CNC machine through exploration of timed state space

---

Miguel A. Mujica\* and Miquel Angel Piera

Department of Telecommunications and Systems Engineering,  
Autonomous University of Barcelona,  
Campus UAB edif. Q,  
08193 Bellaterra, Barcelona, Spain  
E-mail: MiguelAntonio.Mujica@uab.es  
E-mail: MiquelAngel.Piera@uab.cat  
\*Corresponding author

**Abstract:** Flexible production units provide very efficient mechanisms to adapt the type and production rate according to fluctuations in demand. The optimal sequence of the different manufacturing tasks in each machine is a challenging problem that can deal with important productivity benefits. In this paper a Coloured Petri Net model of an eyeglass flexible production machine is presented. The problem of finding an optimal sequence for this machine is faced using an algorithm which uses the model together with the exploration of the state space to find an optimal or close to the optimal scheduling that minimises the makespan of different workloads.

**Keywords:** timed coloured Petri nets; state space; manufacture; production systems; optimisation.

**Reference** to this paper should be made as follows: Mujica, M.A. and Piera, M.A. (2010) 'Performance optimisation of a CNC machine through exploration of timed state space', *Int. J. Simulation and Process Modelling*, Vol. 6, No. 2, pp.165–174.

**Biographical notes:** Miguel A. Mújica Mota studied Chemical Engineering at Autonomous Metropolitan University of Mexico. He also studied an MSc in Operations Research in the National Autonomous University of México and he received a Master's in Industrial Informatics from the Autonomous University of Barcelona. Actually he is a PhD student at Autonomous University of Barcelona. He has professional experience in manufacture and production planning in the cosmetic industry. His research interest focuses on optimisation techniques using the Coloured Petri Nets formalism aiming to solve industrial problems.

Miquel Àngel Piera received his MSc (Control Engineering) from the University of Manchester Institute of Technology in 1990 and his PhD Degree from the Autonomous University of Barcelona (Spain) in 1994. He participates in industrial research projects in the logistics and manufacturing field and at present he is Co-director of LogiSim, a Modelling and Simulation Institution sponsored and founded by the local government of Catalonia. Recently, he has published a modelling and simulation book that is being used for teaching in many Spanish universities.

---

### 1 Introduction

A manufacturing process can be understood as a sequence of different activities that transform raw materials into goods. According to each particular industry, different transformation tasks (each one with a particular set of constraints) must be properly sequenced. The coordination of production resources through production steps must be carried out under the time restrictions inherent to the dynamic of the system (scheduling).

Because of the inherent difficulties to deal with a sequence that could minimise non-productive time operations and maximise a cost function preserving all the production constraints, it is necessary to develop decision-support tools that help the scheduler or

decision-maker to give an acceptable schedule or solution, which fulfils all the requirements.

Industrial systems can be modelled in detail using available simulation tools (Arena, PROMODEL, etc.), whose results are very useful to evaluate the performance of different scenarios. Unfortunately, simulation-based optimisation as a decision-support tool has the drawback that it only covers a small set of the whole possible scenarios of the system, which does not guarantee that the obtained production sequence is the best one.

To ensure optimality, all the possible scenarios of a system must be analysed, but this process cannot be performed using current commercial simulation tools. CPN is a modelling formalism, which has been widely used to

model and verify discrete event systems. The Timed State Space (TSS) of CPN is an analysis tool, which allows storing the different states of the model along with their time characteristics. State Space analysis has been used traditionally to verify properties such as liveness, boundedness and reachability among others, which determine the behaviour of the model (Jensen and Kristensen, 2009). The TSS can also be explored to find sequences of nodes that go from an initial node (initial state of the model) to an objective one representing feasible solutions to the scheduling problem. The problem that arises with state space approaches is the state explosion (Valmari, 1998), which sometimes forces a partial exploration of the TSS or the implementation of reduction techniques (Christensen and Mailund, 2002; Jensen and Kristensen, 2009), which cannot be applied in all types of models.

In Narciso et al. (2005), an approach to explore the most promising branches of the TSS of a system specified under the CPN formalism is presented. It evaluates the time stamp values of each node to determine if a branch must be explored or not. In Mujica and Piera (2008, 2009), the original algorithm has been redesigned to separate the generation of the TSS from the analysis of repeated nodes supporting specific heuristics for both phases (two-step algorithm), which produces a better overall performance.

Analysing the TSS under this perspective does not ensure optimality yet, but the performance is improved yielding results close to optimal configurations or even obtaining sometimes the optimal solutions after the second phase. The algorithm in two phases has been implemented to give a solution for the model of an eyeglass production machine.

The number of nodes in the TSS of the developed model varies according to the workload to be sequenced. The normal workload for this kind of machines ranges from 20 to 25 eyeglasses.

The rest of this paper is organised as follows. In the following section, the modelling formalism and the analysis tool is introduced. In Section 3, the optimisation algorithm is briefly explained. In Section 4, the CPN model of the CNC machine is presented. Section 5 presents the different scenarios analysed for this problem. The results and comparison tables are presented in Section 6. Finally, conclusions and future work are given in Section 7.

## 2 Modelling formalism

Coloured Petri Net (CPN) is a simple yet powerful modelling formalism, which allows the modelling of complex systems, which present an asynchronous, parallel or concurrent behaviour and can be considered discrete event dynamic systems (Jensen, 1997a, 1997b). The formalism allows modelling not only the dynamic behaviour of systems but also the information flow, which is an important characteristic and very useful in industrial systems modelling.

### 2.1 Coloured Petri nets

A CPN can be defined as the tuple (Jensen, 1997a):

$$\text{CPN} = (\Sigma, P, T, A, N, C, G, E, I)$$

where

$\Sigma = \{C_1, C_2, \dots, C_{cn}\}$  represents the finite and non-empty set of colours. They allow the attribute specification of each modelled entity.

$P = \{P_1, P_2, \dots, P_{np}\}$  represents the finite set of place nodes.

$T = \{T_1, T_2, \dots, T_m\}$  represents the set of transition nodes, which normally are associated to activities in the real system.

$A = \{A_1, A_2, \dots, A_{an}\}$  represents the directed arc set, which relates transitions and place nodes.

$N$  is the node function  $N(A_i)$ , which is associated with the input and output arcs. If one is a place node, then the other must be a transition node and vice versa.

$C$  is the colour set function,  $C(P_i)$ , which specifies the combination of colours for each place node.

$$C(P_i) = C_j \quad P_i \in P, C_j \in \Sigma.$$

$G$  is a guard function, it is associated to transition nodes,  $G(T_i)$ , and it is normally used to inhibit the event associated with the transition based on the attribute values of the processed entities. If the processed entities satisfy the arc expression but not the guard, the transition will not be enabled.

$E$  is the function for the arc expressions  $E(A_i)$ . For the input arcs, it specifies the quantity and type of entities that can be selected among the ones present in the place node to enable the transition. When it deals with an output place, it specifies the values of the output tokens for the generated state when transition fires.

$I =$  Initialisation function  $I(P_i)$ , it allows the value specification for the initial entities in the place nodes at the beginning of the simulation. It is the initial state for a particular scenario.

The number of tokens and the token colours in each individual place represent the state or *marking* of the model.

To evaluate the system performance, it is more useful to develop timed CPN models. They differ from the ordinary CPN in the way that the tokens have time stamps representing the time at which the correspondent tokens are available for transition evaluation purposes and each state has a global clock representing the simulation time (Jensen, 1997a, 1997b).

### 2.2 The Timed State Space

The state space of CPN models is the set of all possible states that can be reached from an initial state; therefore,

it is also called the *Reachability Tree* (RT). Traditionally, the RT has been used to evaluate CPN properties such as *reachability*, *boundedness*, *liveness* among others (Christensen and Mailund, 2002; Jensen, 1997a, 1997b), which determine the behaviour of the model.

The RT is a directed graph in which the nodes represent the states of the timed CPN model that are connected with their successor nodes through directed arcs. The characteristics of the RT used in this work are:

- the root node represents the initial marking of the system
- there must be generated as many successor nodes as combination of tokens enable a transition in the marking
- the successor nodes correspond to the new states once the firings have taken place
- the connecting arcs represent transition firings and also contain the information of the elements that caused the firings
- if it is detected that a successor state has the same colour combination but different time values of one that has been previously generated, it will be labelled as *old node* and it will not be evaluated again
- if a state does not have any enabled transition, this state will be called a *Dead Marking*
- if a node is neither a dead marking nor an old one, it is a new state (which denotes a state never reached before).

When dealing with TSSs, it is important to specify which type of clock policy is used. It has been demonstrated (Piera and Mušič, 2009) that certain clock policies are insufficient for evaluating all the feasible configurations of a system. The TSS implemented in this work is an Event-Driven State Space (EDSS) (Mujica and Piera, 2009), which falls into the definition of the so-called *Early State Space* mentioned in Piera and Mušič (2009). The main characteristic of this SS is that it generates the successors of a state firing the transitions that are colour-enabled (relaxing the time restrictions in the evaluation) and thereafter the global clock is advanced until its value matches the time stamp with the highest value of the tokens that participate in the firing so the time restrictions are not violated. With the use of this clock policy, all the possible events or fireable transitions from the same marking are included, e.g., if two transitions T1 and T2 are in conflict and T1 is enabled before T2 it generates the resulting successor state from the firing of T1 and also the successor from the firing of T2. The advantage of using this type of SS is that all the feasible configurations are covered but the size of the SS generated is bigger than the one generated with a different approach such as the *Reduced Earliest Time State Space* (RSS) (Piera and Mušič, 2009) used in software tools such as CPNTools.

### 3 The optimisation algorithm

An algorithm in two phases, also referred as the *two-step algorithm* (Mujica and Piera, 2008), had been developed based on the previous work of Narciso et al. (2005). It avoids the generation of similar branches, which differ between them only by their time values with the time analysis of the old nodes.

In the first step, the algorithm generates the different states that can be reached from the initial state using a depth-first traversal algorithm (DFS) until the complete RT is generated or the final conditions are satisfied (Mujica and Piera, 2008). When a timed state is generated in this phase and its underlying untimed state is exactly the same as the one from a previous generated timed state but with different time values, the repeated state is not evaluated and the original state (*old node*) is stored in a data structure along with the time elements of the repeated state. This procedure is carried out for all the repeated states consequently reducing the computational effort through the avoidance of branch exploration in the RT.

#### 3.1 State generation step

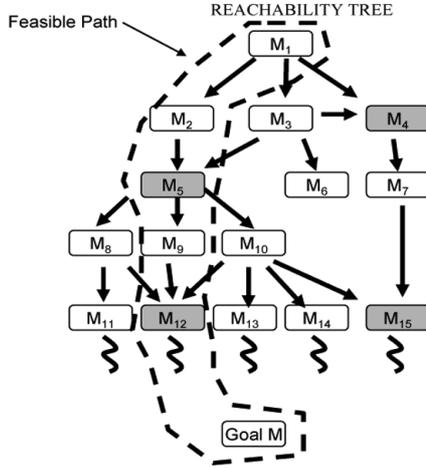
The first step of the algorithm generates the EDSS of the correspondent model. The algorithm generates as many new states as different token combinations enable the transitions based on the restrictions imposed by the colours and managing the global clock as mentioned in the previous section. In the first step, the state-generation algorithm also searches for the objective state. Once the objective state is found, it stores the path that goes from the initial state to the objective one.

The successor states are generated in the first phase using a Depth-First Search (DFS) algorithm. It avoids the storage of states that are differentiated only by their time stamps each time an old node is found. In such a case, the repeated state is not evaluated again, but its time elements are stored for a later time analysis, making reference to the node that has been previously generated in the RT.

Figure 1 illustrates how the old nodes are stored during the generation phase in a separated list. On the left-hand side of the figure, the grey-shaded  $M_4$ ,  $M_5$ ,  $M_{12}$  and  $M_{15}$  nodes represent the timed nodes that first appeared during the generation phase. The nodes with more than one incoming arc represent old nodes whose timed elements will be evaluated in the second phase. The right-hand side of the figure represents the list with the time information of the repeated old nodes that will be analysed in the second phase of the algorithm. To illustrate the generation phase, let us take as example the old node  $M_4$  that can be reached from  $M_1$  and  $M_3$ . In this case, the correspondent element in the list is composed by the information of  $M_4$  node and the time information generated by the firings of states  $M_1$  and  $M_3$ ; in a similar way, the stored information for  $M_5$  is the one generated by nodes  $M_2$  and  $M_3$ . The analysis of the remaining nodes  $M_{12}$  and  $M_{15}$  is straightforward from

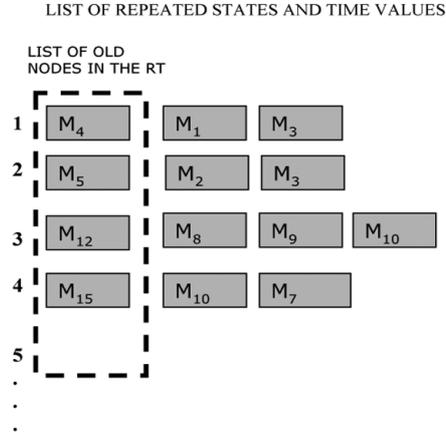
the figure. This management of old nodes allows storing at most the same number of states as the ones for the underlying untimed state space since these old nodes are stored and evaluated only once during the first step. The remaining nodes that generate an old node are referenced to

Figure 1 A feasible path and the old-node list



the original timed node and their time information is stored in the list.

In this first step, some questions about the model properties can be answered as well (e.g., deadlocks, reachability, etc.).



### 3.2 The optimisation step

The second phase of the algorithm analyses similar branches in the RT through the evaluation of the time elements of the old nodes stored in the old-node structure (right-hand side of Figure 1) so that it can select the ones that optimise the feasible paths. It analyses the repeated nodes to determine which time stamp combination is the best for the optimisation objectives (reduction of makespan in this study). The old nodes are evaluated under the decision rule outlined in Narciso et al. (2005), which can be formalised as follows.

Given any two old nodes  $M_i$  (the first generated state) and  $M_j$ , and the time stamp lists of size  $k$  of each node  $T_i$  and  $T_j$ , the time stamp comparison is made with the use of the  $\Psi$  and  $\Phi_k$  functions:

$$\Psi : \mathbb{R} \times \mathbb{R} \rightarrow \{0, 1\}$$

$$(x, y) \mapsto \begin{cases} 0, & x \geq y \\ 1, & x < y \end{cases} \quad (1)$$

$$\Phi_k : \mathbb{N}^k \times \mathbb{N}^k \rightarrow \mathbb{N}$$

$$(T_i, T_j) \mapsto \sum_{l=1}^k \Psi(T_{il}, T_{jl}) \quad (2)$$

Applying the function  $\Phi_k$  to the old nodes  $M_i$  and  $M_j$  ( $M_i$  is the old node that appeared first in the SS and  $M_j$  the old node with different time stamps), the following three possible outcomes can be obtained:

$\Phi_k = 0$ : The time stamp values of the  $M_i$  state are equal or higher than the ones of the  $M_j$  state. In this case, the best branch is chosen from among the successors of  $M_i$  (since the best branch of  $M_i$  will also be the best one for  $M_j$ ) and the time values of the path that goes to the objective state are

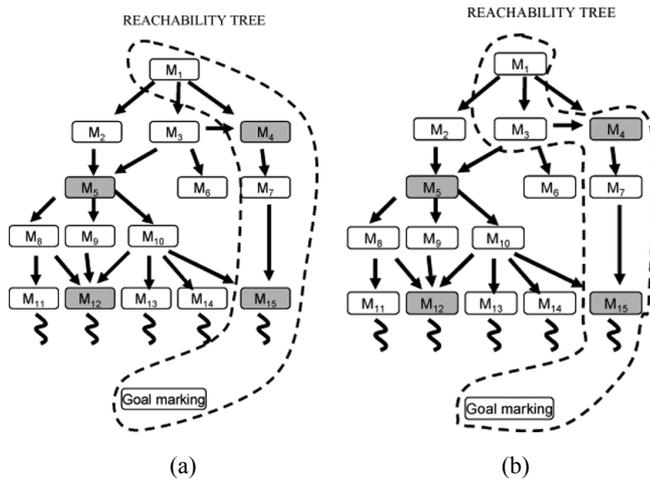
updated by analytical evaluation of the new time values using the time stamps of the  $M_j$  state. The previous procedure is done using token and transition information stored from the generation of  $M_i$ .

$\Phi_k = k$ : The time stamp values of the  $M_i$  state are smaller than the ones of  $M_j$  state. In this case, it is not necessary to evaluate again the successor nodes of the  $M_j$  state because the underlying untimed states will be the same as those of  $M_i$ , and the time values will not be smaller than the ones generated by  $M_i$ . The previous procedure is done using token and transition information stored from the generation of  $M_i$ .

$0 < \Phi < k$ : Some time stamp values of the  $M_j$  state are equal or higher than the ones of  $M_i$  state and others are less than the ones of  $M_i$  state. In this case, it is not possible to decide whether the  $M_j$  state produces better results in the objective marking or not. Therefore, an exploration through the best branch has to be done until the objective marking is found and then the final comparison can be made.

With the use of this decision rule, those states that produce an objective state with poor time values are not analysed. For the old nodes, which give an unclear result (3rd outcome), an exploration to the objective state is done. This procedure allows the evaluation of the final state and consequently a decision about the original node can be made. A time-updating procedure is done in the feasible path for those states that surely produce better values in the objective state.

To exemplify the selection procedure that takes place in the second step, let us consider Figure 2; it illustrates how the feasible path is updated once the old nodes are analysed at the second step.

**Figure 2** The second phase of the optimising algorithm

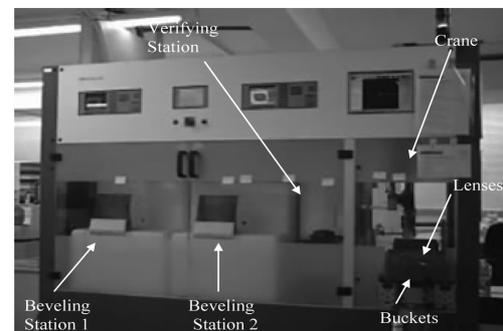
The  $M_4$  time values (first element in the old node list) are compared with the time values of the repeated old state that can be reached from  $M_3$  to check if it produces better time values than those of  $M_1$  (Figure 2, e.g., A). In the case that  $M_3$  produces better time values than  $M_1$ , the feasible path that goes to the objective state is updated with the time information generated from node  $M_3$  (Figure 2, e.g., B). The same analysis is performed for the remaining elements in the old node list ( $M_5$ – $M_{15}$ ). During this process, the time values of the paths that improve the objective state in the RT are updated with the best time values maintaining always the time consistency from the initial state to the objective state, i.e., there will be paths that are not time-consistent but those paths can be ignored in the time-updating procedure since they produce bad time values in the objective state.

When the main objective is the optimisation of the makespan, it is possible to improve the performance analysis of the two-step algorithm through heuristics in the first step. The idea behind the heuristics is to lead the DFS algorithm through the implementation of a cost function that assigns a value to each node in the successor nodes. The function analyses the time stamp values of a state and assigns a value to each state to differentiate which one will increase less the global clock in the subsequent evaluation. The DFS algorithm selects the states that reduce the most the global clock advance. In this work, the implemented algorithm makes the selection of the different nodes using a probability-based heuristic presented in Mujica and Piera (2009). Some authors have developed similar approaches for the kind of problems presented in this paper (Lee and DiCesare, 1994; Yu et al., 2003). The approach implemented in their works is different from the one used in this paper in several aspects; they only estimate the value of the objective node using heuristics such as the A\* algorithm. Although they do not say it explicitly, it is inferred from the algorithm description that they implemented a kind of RSS, which is completely time-driven; on the other hand, they also solve with re-exploration the problem of finding a repeated node, which is a very time-consuming task. The algorithm used

in the work presented here uses a more conservative approach but it is more reliable since it discards only those states that certainly (without estimation) produce worse time values, and when it is not possible to know *a priori* if the repeated node would produce a better or worse makespan then an exploration to the objective state is performed instead of updating the whole subbranch that hangs from the repeated node.

#### 4 System description

The presented case study is about the scheduling of an automated machine using the algorithm presented in the previous section. The machine that is the subject of this work has to verify and correct eyeglasses. Figure 3 illustrates the elements of the automated machine.

**Figure 3** The automated CNC machine

The machine has three workstations. One is for verification purposes (*Ver*); it checks whether the lenses have the correct dimension specification. The second and third workstations make some labour on the lenses so they can fit in the correspondent chassis model (*Beveling: B1 and B2*). One crane with two holders (*G1 and G2*) is used for the transportation of the lenses from the buckets with lenses to the workstations. Each eyeglass is composed by two lenses (right and left lenses). The glasses must follow a sequence through the whole process. First, they must be verified and then they must be bevelled. It is not mandatory that once one eyeglass has undergone one operation it must follow the remaining operations immediately. It is possible that the machine takes one lens for an operation, and thereafter takes a different one for another operation. The lenses can be selected for their operations without any particular order, but the operation sequence of each eyeglass must follow a specific order: a verifying operation followed by a beveling operation.

The latter must be done for each pair of lenses so if three glasses are to be processed, then six lenses must follow the operations in the workstations.

The problem consists to determine the best operations sequence (considering also the crane movements) to serve a certain lens beveling mix. Note that there are three different bevel operations, according to the glass support characteristics each bevel operation requires a different time, which complicates considerably the scheduling policy

to avoid idleness stats in the crane and in the verification machine.

### 4.1 Colour definition

The production system was modelled with the timed-CPN formalism. Information about processing times for the bevelling operations as well as the values of the operation sequences have been coded in the colours of each token. Table 1 shows the colours used to encode the information in the model. Column 1 defines the variable used for the colour, column 2 defines the domain values of the correspondent variables, and column 3 gives a brief description of the information stored in the colours.

**Table 1** Colour definition for the CPN model

Colour	Definition	Description
I	{0, 1}	This colour is used to allow only one lens movement through the guard in place <i>G</i>
Idc	Integer	This colour is used only as identifier
lid	Integer	This colour is used as the lens identifier in the VER place
lidl	Integer	This colour is used for the identifier of the left lens in the glass
lidr	Integer	This colour is used for the identifier of the right lens in the glass
ell	Integer	This colour identifies the number of operation already taken in the left lens
elr	Integer	This colour identifies the number of operation already taken in the right lens
tvisll	Integer	This colour specifies the time spent when the left lens is being bevelled
tvislr	Integer	This colour specifies the time spent when the right lens is being bevelled
posc	Integer	This colour is used as an identifier of the number of lenses of the same type
posg	Integer	This colour is used to specify the physical position of the crane that move the lenses

The place nodes in the CPN model use multisets composed with the colours previously defined; Table 2 shows the colour sets used for the different place nodes in the CPN model.

**Table 2** Colour set definition for the place nodes

Place	Colour set	Description
Buckets	$C = \text{Product}$ $\text{idc} * \text{posc} * \text{lidl} * \text{lidr} * \text{ell} * \text{elr} * \text{tvisll} * \text{tvislr}$	The tokens in this place hold the information of each pair of glasses, defined by the values of the colours
Crane	$G = \text{product}$ $\text{lidl} * \text{lidr} * \text{posg}$	The tokens in this place model the information of the crane that moves the lenses through the different stations in the process
G	$I$	This place is used only to limit the movement of the crane with the correspondent guards in the transitions that model the movement of the crane

**Table 2** Colour set definition for the place nodes (continued)

Place	Colour set	Description
Ver	$V = \text{lidl}$	The tokens in this place represent the verifying machine, since there is only one verifying station it only has one token
Bevelling	$V = \text{lidl}$	The tokens of this place model the bevelling machines, in this case the system has two machines therefore the model will have two tokens

### 4.2 The CPN transitions

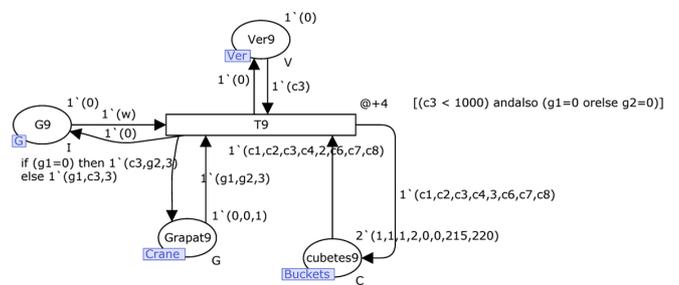
The model has been developed in groups of transitions; two of those groups are for the bevelling and verifying operations, and the two remaining groups are for modelling the transport of the lenses through the different stations in the system and for the movement of the crane itself.

#### 4.2.1 Operation transitions

The verifying operation is a group of four transitions, two for the left lenses, and another two for the right lenses. These transitions model the activities undergone by the lenses during the operation processes. The common input places to these transitions are *Ver*, *Crane*, *Buckets* and *b*. Figure 4 illustrates one of the transitions of this second group. Transition T9 will be enabled if some conditions are satisfied:

- there are buckets, which have already been verified once ( $ell = 2$ )
- the crane is in position number three ( $posg = 3$ ), i.e., one token of type  $1'(g1, g2, 3)$
- one of the holders of the crane is available ( $g1 = 0$  or else  $g2 = 0$ )
- the verifying machine is available ( $c3 < 1000$ )
- the *G* place does not constrain the firing of this transition but it helps to activate a crane movement.

**Figure 4** Example of verifying transitions (see online version for colours)

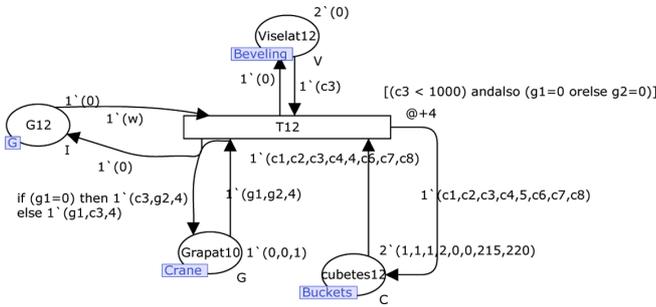


The bevelling operations are a group of four transitions, two for the right lenses and another two for the left lenses. These transitions also have four input places: *Bevelling*, *Crane*, *Buckets* and *G*. To fire the bevelling operation, the following conditions must be satisfied:

- the crane must have available holders ( $g1 = 0$  or else  $g2 = 0$ )
- the previous operations (verifying1, verifying2) have been already performed. This is verified with the identifiers  $ell$  or  $elr$
- the sequence of the bevelling operations is the correct one ( $lidr$  or  $lidr$  identifiers)
- the crane is in the correct position for the bevelling operations ( $posg = 4$ ).

Each time one of these transitions fires, the colour of the token in place  $G$  is turned into 0, allowing another movement of the crane. Figure 5 illustrates one transition of this bevelling group.

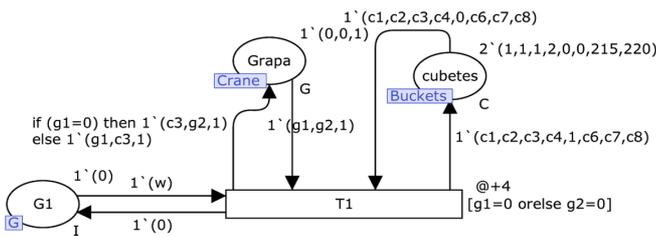
**Figure 5** Example of bevelling transitions (see online version for colours)



#### 4.2.2 Movement transitions

The modelling of the pick-up and delivery of lenses by the crane is modelled with four transitions, two transitions for the pick-up and another two for the delivery activity. Figure 6 presents one transition corresponding to the activity in which the crane takes a left lens from the bucket.

**Figure 6** The pick-up transitions (see online version for colours)



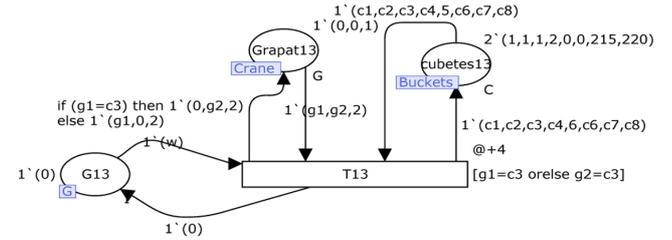
To pick up any lens, the following conditions must be satisfied:

- the crane must be in the correct position ( $posg = 1$ )
- the lenses to be picked up must be ready for the initial operation ( $ell = 0$ )
- any of the two holders of the crane must be empty ( $g1 = 0$  or else  $g2 = 0$ ).

Figure 7 presents an example of the transition that models the delivering of the left lens. In this case, the following conditions must hold to fire the transition:

- the crane must be in the delivery position for the correspondent operation (in this example position 2, stated by the value of colour  $posg$ )
- the operation number carried out in the left lens must correspond to the operation, expressed by the  $ell$  colour (value 5)
- the crane must carry the correspondent left lens, stated by the value of  $c3$ .

**Figure 7** An example of the delivery transitions (see online version for colours)



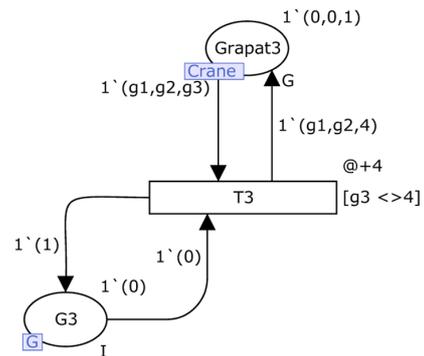
The crane movement is modelled with a group of four transitions. Figure 8 exemplifies one transition of the movement group.

The crane movement is modelled by the token in the place node  $Crane$ , the  $posg$  colour models the current position of the crane. Two conditions must be fulfilled to allow the movement:

- the correspondent position is not already occupied by the crane ( $g3 <> 4$ )
- last operation was not a crane movement, i.e., the token in the  $G$  place is  $1'(0)$ .

The transitions presented in this section are examples of the group transitions. The complete model can be constructed in a straightforward way following the examples given.

**Figure 8** Crane movement (see online version for colours)



## 5 Implemented scenarios

The eyeglasses types are differentiated by the identifiers and by the amount of work that must be done in the bevelling operation. On the basis of this information, three types of glasses are defined for the CPN model, which corresponds to tokens in the  $Buckets$  place node:

Type A glass: 1' (1,1,1,2,0,0,215,220)

Type B glass: 1' (2,1,3,4,0,0,120,120)

Type C glass: 1' (3,1,5,6,0,0,540,540)

To analyse the system, some workloads were evaluated. Table 3 shows the workload information. The first column shows the size and glasses type to be processed, and the

second presents their correspondent initial state of the timed CPN model.

The analysis of the RT started with small workloads for the same type of eyeglasses (i.e., two and three eyeglasses of type A). The performance was then tested increasing the workload variety. The model was finally tested using a real-size workload (22 lenses).

**Table 3** Workload scenarios

<i>Work load</i>	<i>Initial marking</i>
2 Glasses, 2 type A	Crane: 1' (0,0,1)@0 Buckets: 2' (1,1,1,2,0,0,215,220)@[0,0] G: 1' (0)@0 Ver: 1' (0)@0 Bevelling: 2' (0)@[0,0]
3 Glasses, 3 type A	Crane: 1' (0,0,1)@0 Buckets: 3' (1,1,1,2,0,0,215,220)@[0,0,0] G: 1' (0)@0 Ver: 1' (0)@0 Bevelling: 2' (0)@[0,0]
3 Glasses, 2 type A, 1 type B	Crane: 1' (0,0,1)@0 Buckets: 2' (1,1,1,2,0,0,215,220)@[0,0]+1' (2,1,3,4,0,0,120,120)@0 G: 1' (0)@0 Ver: 1' (0)@0 Bevelling: 2' (0)@[0,0]
4 Glasses, 2 type A, 1 type B, 1 type C	Crane: 1' (0,0,1)@0 Buckets: 2' (1,1,1,2,0,0,215,220)@[0,0]+1' (2,1,3,4,0,0,120,120)@0 +1' (3,1,5,6,0,0,540,540)@0 G: 1' (0)@0 Ver: 1' (0)@0 Bevelling: 2' (0)@[0,0]
22 Lenses (normal workload): 7 type A, 13 type B, 2 type C	Crane: 1' (0,0,1)@0 Buckets: 7' (1,1,1,2,0,0,215,220)[0,0,0,0,0,0,0,0]+13' (2,1,3,4,0,0,120,120)@[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]+2' (3,1,5,6,0,0,540,540)@[0,0] G: 1' (0)@[0] Ver: 1' (0)@[0] Bevelling: 2' (0)@[0,0]

The primary target of these tests was to evaluate the size of the TSS and the final objective was to obtain an optimal scheduling for real-size workloads or at least a schedule close to the optimal.

## 6 Results

The two-step algorithm was applied to solve the scheduling of the different workloads. The second phase analysed the repeated states on a sequential order without implementing any special selection rule. The resulting makespan for the different configurations is presented in Table 4. The first column shows the final state of the system along with its

time stamp values, the second column shows the makespan of the final state expressed in seconds, the third column shows if the RT was explored completely and the fourth column shows structural information of the RT generated.

The results of the first three scenarios were obtained exploring the complete RT. It can be seen that when the workload comprises more than three eyeglasses the analysed RT is bigger than 800,000 nodes. Because of computer resource limitations, the real workload for this machine could not be solved exploring the complete RT, due to that it was solved with a partial analysis. The analysis gave a makespan of 10,054 s when 800,000 nodes were analysed.

**Table 4** Scenario solution

<i>Final marking</i>	<i>Makespan</i>	<i>No. of different analysed nodes</i>	<i>Structural information of the TRT</i>	
Crane: 1' (0,0,1)@635	635 s	Complete RT	No. nodes:	19,232
Buckets: 2' (1,1,1,2,6,6,215,220)@[627,631]			No. arcs:	59,610
G: 1' (0)@635			No. old nodes:	15,431
Ver: 1' (0)@387			Levels:	53
Bevelling: 2' (0)@[590,619]				
Crane: 1' (0,0,1)@1343	1,343 s	Complete RT	No. nodes:	172,242
Buckets: 3' (1,1,1,2,6,6,215,220)@[853,1096,1339]			No. arcs:	765,177
G: 1' (0)@1343			No. old nodes:	145,911
Ver: 1' (0)@1104			Levels:	84
Bevelling: 2' (0)@[1088,1331]				
Crane: 1' (0,0,1)@958	958 s	Complete RT	No. nodes:	562,799
Buckets: 2' (1,1,1,2,6,6,215,220)@[917,950] +1' (2,1,3,4,6,6,120,120)@954			No. arcs:	1,800,951
G: 1' (0)@958			No. old nodes:	471,939
Ver: 1' (0)@715			Levels:	81
Bevelling: 2' (0)@[909,942]				
Crane: 1' (0,0,1)@1905	1913 s	500,000 nodes	–	
Buckets: 2' (1,1,1,2,6,6,215,220)@[1482,1881]+1' (2,1,3,4,6,6,120,120)@1897+1' (3,1,5,6,6,6,540,540)@1901	1905 s	800,000 nodes	No. nodes: —	
G: 1' (0)@1905			No. arcs: >2,541,330	
Ver: 1' (0)@1646			No. old nodes: >676,223	
Bevelling: 2' (0)@[1873,1889]			Levels: 105	
Crane: 1' (0,0,1)@10054	10,086 s	400,000 nodes	–	
Buckets: 7' (1,1,1,2,6,6,215,220)@[8287,8530,8773,9032,9239,9470,9482] +13' (2,1,3,4,6,6,120,120)@[5929,6093,6257,6421,6585,6749,6913,7077,7229,7377,7541,7653,7801]+2' (3,1,5,6,6,6,540,540)@[5765,10050]	10,054 s	800,000 nodes	No. nodes: —	
G: 1' (0)@10054			No. arcs: >3,911,731	
Ver: 1' (0)@9490			No. old nodes: >675,161	
Bevelling: 2' (0)@[9474,10042]			Levels: 614	

When the RT is partially explored, it is not possible to know how close to the optimal the solution is. To compare the two-step algorithm with another state space approach, the same model has been coded in CPNTools whose state space tool generates the complete RT without any simplification. Table 5 presents the state space comparison between these two approaches.

**Table 5** Comparison with the CPN SS algorithm

<i>Workload</i>	<i>CPN tools timed model</i>		<i>Two-step algorithm</i>	
2 Glasses, 2 type A	Nodes:	389,884	No. nodes:	19,232
	Arcs:	457,992	No. arcs:	59,610
	Secs:	8558	No. old nodes:	15,431
	Status:	Full	Levels:	53
3 Glasses, 3 type A	Unable to generate the SS		No. nodes:	172,242
			No. arcs:	765,177
			No. old nodes:	145,911
			Levels:	84

**Table 5** Comparison with the CPN SS algorithm (continued)

<i>Workload</i>	<i>CPN tools timed model</i>	<i>Two-step algorithm</i>	
3 Glasses, 2 type A, 1 type B	Unable to generate the SS	No. nodes:	562,799
		No. arcs:	1,800,951
		No. old nodes:	471,939
		Levels:	81

This table allows appreciating the amount of computational efforts avoided with the two-step algorithm. It can be seen that the size of the RT using an exhaustive search for timed models is considerably bigger than the one generated by the two-step algorithm. It can also be appreciated that for big workloads the state space algorithm in CPNTools fails generating the whole RT due to computer resource limitations. The saved information by the two-step algorithm allows exploring in the second step the branches that give better results in the objective state.

## 7 Conclusions and future work

A case study of an eyeglasses CNC machine has been presented. It can process a variety of eyeglasses with different types of lenses. Some operations must be carried out on the lenses, and they must follow a specified sequence to be completed. The problem of sequencing the operations in an optimal way is a difficult task and must be solved in a smart way due to the combinatorial nature of the problem. The scheduling problem was faced implementing a two-phase algorithm in combination with a CPN model. The algorithm was run for five workloads, but the complete information of the RT could be generated only in the first three cases, which appear to have a relative small RT.

The real-size workload problem could not be solved generating the complete information of the RT due to computer resource limitations, but the advantage of using the two-step approach (which uses a DFS algorithm in the generating phase and a particular way of storing the repeated states) allowed to generate a feasible solution. The DFS algorithm selects the nodes to be analysed based on a heuristic, which has given good results in other applications (Mujica and Piera, 2008), therefore it can be expected that the given solution is not far from the optimal scheduling. The optimal solution of the scheduling problem of the 22 lenses is left as an opened unsolved problem. On the basis of the size of the state space, it can be used as a benchmarking for optimisation algorithms.

As a future work, some implementations will be made in the two-step algorithm such as the one presented by Yu et al. (2003) for the P/T Petri Nets. Those heuristics can be implemented in some procedures of the two-step algorithm to improve the algorithm's performance. These implementations will be compared with a benchmark problem such as the one presented here to evaluate their efficiency.

## References

- Christensen, S. and Mailund, T. (2002) *A Generalized Sweep-Line Method for Safety Properties*, FME, Springer-Verlag, Berlin-Heidelberg.
- Jensen, K. (1997a) *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*, Vol. 1, Springer-Verlag, Berlin.
- Jensen, K. (1997b) *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*, Vol. 2, Springer-Verlag, Berlin.
- Jensen, K. and Kristensen, L.M. (2009) *Coloured Petri Nets: Modeling and Validation of Concurrent Systems*, Springer-Verlag, Heidelberg.
- Lee, D.Y. and DiCesare, F. (1994) 'Scheduling flexible manufacturing systems using Petri nets and heuristic search', *Transactions of Robotics and Automation*, Vol. 10, No. 2, pp.123–132.
- Mujica, M. and Piera, M.A. (2008) 'Optimizing time performance in reachability tree-based simulation', *Proceedings of the International Mediterranean Modeling Multiconference*, 17–19 September, Campora Sant Giovanni, Italy.
- Mujica, M. and Piera, M.A. (2009) 'A two step algorithm to improve systems optimization based on the state space exploration for timed coloured Petri nets', *Proceedings of the TiSto Workshop*, 22–26 June, Paris, France.
- Narciso, M., Piera, M.A. and Figueras, J. (2005) 'Optimización de Sistemas Logísticos Mediante Simulación: Una Metodología Basada en Redes de Petri Coloreadas', *Revista Iberoamericana de Automática e Informática Industrial*, Valencia, España, Vol. 2, No. 4, pp.54–65IAII.
- Piera, M.A. and Mušič, G. (2009) 'Coloured Petri nets: timed state space exploration examples and some simulation shortages', *Proceedings of the MathMod 09*, Vienna, Austria, pp.123–131.
- Valmari, A. (1998) 'The state explosion problem', *Lecture Notes in Computer Science*, Springer-Verlag, London, Vol. 1491, pp.429–528.
- Yu, H., Reyes, A., Cang, S. and Lloyd, S. (2003) 'Combined Petri net modelling and AI based heuristic hybrid search for flexible manufacturing systems – Part 1. Petri net modelling and heuristic search', *Computers and Industrial Engineering*, Vol. 44, pp.527–543.

## Website

[http://wiki.daimi.au.dk/cpntools/\\_home.wiki](http://wiki.daimi.au.dk/cpntools/_home.wiki)