# AN IMPROVED TIME-LINE SEARCH ALGORITHM TO OPTIMIZE INDUSTRIAL SYSTEMS

**Miguel Mujica[(a)], Miquel Angel Piera[(b)]**

[(a,b)]Autonomous University of Barcelona, Faculty of Telecommunications and Systems Engineering, 08193, Bellaterra, Barcelona

[(a)]miguelantonio.mujica@uab.es, [(b)]miquelangel.piera@uab.es

**ABSTRACT**
The coloured Petri net formalism has been used recently to analyze and optimize industrial systems making use of the state space analysis. This approach has great potential to give very good results when it is properly implemented. In this article an improved version of the algorithm known as the *time line search* for optimizing the makespan of manufacturing models is presented. The algorithm uses a compact state space of coloured Petri net models in order to analyze the highest possible number of configurations.

Keywords: timed Petri nets, state space, optimization, simulation, manufacturing.

## 1. INTRODUCTION

In this article an improved version of the time line search algorithm is presented. The initial version of the TLS (Mujica and Piera 2010) was implemented as a heuristic to generate the state space using an algorithm in two phases (Mujica et al. 2010). The performance of the implementation yielded very good results when it was implemented in models that had state spaces small enough so they could be stored in the computer memory. Based on those results an algorithm that analyses and generates the compact timed state space (CTSS) in a better way was devised. The original algorithm, (which will be called from now on as the *old* algorithm) presented some shortages that caused some time penalties due to the node evaluation activity. The new implementations are devised with the purpose of overcoming those drawbacks giving as a result a better version of the time line search algorithm. The new algorithm has been tested with the industrial model of a CNC eye-glass machine (Mujica and Piera 2009b) which generates big state spaces when small workloads are simulated.

## 2. TIMED COLOURED PETRI NETS

Coloured Petri Nets (CPN) is a simple yet powerful modelling formalism which allows to properly model discrete-event dynamic systems which present a concurrent, asynchronous and parallel behaviour (Moore et al. 1996, Jensen 1997). CPN is a bipartite graph which is composed of two types of nodes: the place nodes and the transition nodes. Place nodes are commonly used to model system resources or logic conditions, and transition nodes are associated to activities of the real system. The entities that flow in the model are known as tokens and they have attributes known as colours. The characteristics of the formalism allow modelling not only the dynamic behaviour of systems but also the information flow which is a key attribute in decision making.

In order to evaluate systems performance it is necessary to make an extension to the formalism attaching time stamps that determine the availability of tokens, a global clock that represents the model time and a time delay to transitions that model the time consumed by the activities. Formally they can be defined as follows.

Definition 1. Timed Coloured Petri Nets (TCPN)

$$TCPN = (P, T, A, \sum, V, C, G, E, D, I) \text{ where}$$

1. P is a finite set of places.

2. T is a finite set of transitions T such that $P \cap T = \varnothing$

3. $A \subseteq P \times T \cup T \times P$ is a set of directed arcs

4. $\sum$ is a finite set of non-empty colour sets.

5. V is a finite set of typed variables such that $Type[\upsilon] \in \sum$ for all variables $\upsilon \in V$.

6. $C: P \rightarrow \sum$ is a colour set function assigning a colour set to each place.

7. $G: T \rightarrow EXPR$ is a guard function assigning a guard to each transition T such that $Type[G(T)] = Boolean$.

8. $E: A \rightarrow EXPR$ is an arc expression function assigning an arc expression to each arc *a*, such that:
$$Type[E(a)] = C(p)$$
Where p is the place connected to the arc *a*

9. $D: T \rightarrow EXPR$ is a transition expression which assigns a delay to each transition (this delay is commonly represented with a '@' sign).

10. I is an initialization function assigning an initial timed marking to each place p such that:
$Type[I(p)] = C(p)$

EXPR denotes the expressions used by the inscription language, and TYPE[*e*] denotes the type of an

expression $e \in$ EXPR, i.e. the type of values obtained when evaluating *e*. The set of free variables in an expression *e* is denoted VAR[*e*] and the type of a variable *v* is denoted TYPE[*v*].
Type [I(p)] = C(p)

In TCPN context the state of every model is also called the *timed marking* which is composed by the expressions together with their time stamps associated to each place p and they must be closed expressions i.e. they cannot have any free variables.

The markings are defined as follows:

Definition 2. The *timed marking* of a TCPN is a function $M^T$:P $\rightarrow$ EXPR such that $M^T(p) \in C(p)$. It maps each place *p* into a multi set of values $M^T$(p) representing the timed marking of place *p*. The individual elements of the multi set are called *timed tokens* and the expressions contain also time stamps.

Definition 3. The *untimed marking* $M^U$ of a TCPN model is a function $M^U$ :P $\rightarrow$ EXPR that maps each place *p* into a multi set of values $M^U(p) \in C(p)$ representing the untimed marking of place *p* . In this case the expressions do not contain any time information.

In order to fire a transition, the number of tokens in the input place nodes (the directed arcs go from the places to the transition) must satisfy not only the arc inscriptions but also the restrictions imposed by the guard expressions. Only the tokens that have time stamp values less than or equal to the global clock participate in the transition enabling procedure.

When a transition occurs, the output tokens will have a time stamp $\Delta t$ time units larger than the current global clock Gc which simulates time delay due to the execution of an activity. The time stamp calculated by formula (1) represents the earliest model time when the output tokens can be used again for a new transition firing, i.e. the token will not be available for $\Delta t$ time units.

$$t_o = Gc + \Delta t \quad (1)$$

Where $t_o$ is the time stamp value that must be attached to the output tokens when the transition firing takes place, Gc is the global clock of the model when the firing occurs and $\Delta t$ is the time associated with the transition.

## 3. THE COMPACT TIMED STATE SPACE

The reachability graph is a directed graph which has been traditionally used by scientific community for the verification and analysis of behavioural properties of timed and non-timed CPN models (Christensen et al. 2001, Kristensen and Mailund 2002, Wolf 2007). The reachability graph is also known as the state space (SS) because it generates and stores all the different reachable states from an initial one. The SS analysis can be performed with timed or untimed models to evaluate properties, such as liveness, boundedness and reachability of states among others (Jensen et al. 2001) to determine the behaviour of the modelled system.

In timed models each node of the SS represents a timed marking of the TCPN model. Some authors have developed different ways of representing the timed state space (TSS) basing their representations on different structural characteristics of the model (Chiola et al. 1997, Jensen et al. 2001). Those representations have been developed in order to reduce or delay as much as possible the state explosion problem (Valmari 1998) without loosing the necessary analysis capabilities to verify model properties.

The following definitions are common to almost every state space representation.

Definition 4. Let $\mathfrak{M}^T$ be the set of timed markings of a state space, and $M_i^T, M_k^T \in \mathfrak{M}^T$ be timed markings. A state $M_k^T$ will be called *old node* if it is exactly the same (together with its time values) as one that have been previously generated in any other level of the SS, i.e. $M_k^T = M_i^T$

It is possible to reduce the amount of states to be analyzed when the symmetry of the colours in the tokens is exploited without taking into account the time stamps of the markings. Therefore it is possible to define a special kind of "repeated state", the symmetric old node or *S-old node*.

Definition 5. Let $\mathfrak{M}^T$ be the set of timed markings of a state space. Let $M_i^T$ and $M_k^T$ be timed markings with their correspondent untimed markings $M_i^U$ and $M_k^U$.

A marking $M_i^T$ is an S-*old node* to another $M_k^T$ marking when the following condition holds:

$$M_i^T, M_k^T \in \mathfrak{M}^T \wedge M_i^U = M_k^U$$

The representation using the S-old nodes is called the *compact timed state space* (CTSS) since it is possible to reduce the amount of space needed to store all the information generated in the state space (Mujica et al. 2010).

Other characteristics common to the CTSS and the TSS are:

- The root node represents the initial marking of the system.
- The successor or children nodes correspond to the new states or markings obtained once the enabled transitions have been fired.
- For each node in the tree as many successor nodes as enabling combination of tokens in the marking must be generated.

• Each node is connected with its successor nodes through directed arcs.

• The connecting arcs represent transition firings and they also contain the information concerning the transition fired and the tokens used in the firing.

## 4. NEW ALGORITHM TO GENERATE THE CTSS

The old *time line search* algorithm has been developed based on an algorithm in two steps (Mujica and Piera 2009a). It generates the CTSS making use of an incremental variable and the firing time of every marking is used as a key that matches the value of the incremental variable. The main elements needed for the algorithm are:

a) A key that is assigned to each node in the CTSS. It will be used to determine the sequence of evaluation. The key takes into account the global clock when the firing takes place and the time stamps of the marking using the following formula:

Being $M_i^T \in \mathfrak{M}^T$ a timed marking, $T_i$ its correspondent time stamp list and *Gc* the global clock of the timed marking. The function $tline_k$ assigns the key in the following way.

$$tline_k : \mathbb{N}^k \times \mathbb{N} \rightarrow \mathbb{N}$$
$$(T_i, Gc) \mapsto Max\{Min\{T_{i1},...,T_{ik}\}, Gc\} \quad (2)$$

b) An incremental variable which determines the group of nodes that must be evaluated next. The nodes whose keys match the value of the variable will be the ones to be evaluated next.

c) A list of node numbers with the sequence of evaluation based on the progress measure.

Making use of these elements the state space is generated and analyzed in two phases. The first phase generates the CTSS following the mentioned sequence and the second phase is used to improve the feasible path that has been found during the first phase.

Since the constructed CTSS is event-driven (Mujica and Piera 2009a), the markings to be evaluated are those that occur closest to the current global clock. Based upon this characteristic, the states with a firing time value greater than the current global value will not be evaluated until the progress value reaches the value of the firing time of the group. The latter situation caused in the old version of the TLS algorithm that the evaluation of some S-old nodes with good potential was delayed even if they would be fired with a global time earlier than the one from the already generated node. Figure 1 illustrates the situation when an S-old node with good potential was forced to a later evaluation.

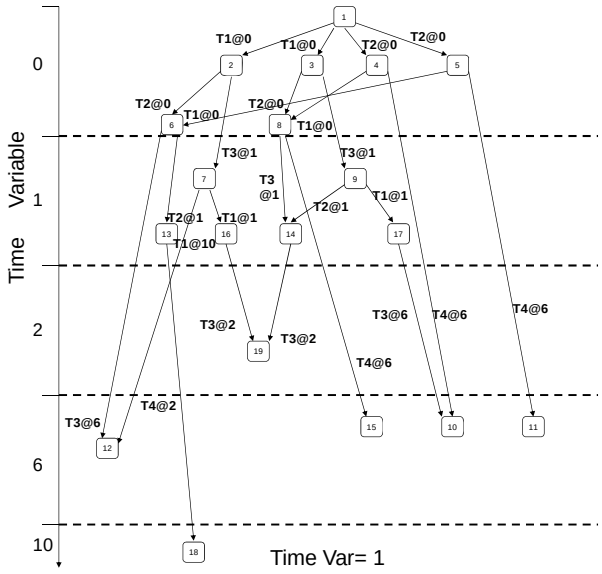


Figure 1: Generation of the SS (drawback)

This figure represents a compact timed state space. The nodes in the figure are being evaluated in order of appearance but the sequence of evaluation depends on the value of the time variable. The first group of nodes (nodes #1,#2,#3,#4,#5,#6 and #8) have been evaluated when the time variable had the value of 0 units. If we put focus on node #6, during its evaluation it generated nodes #12 and #13. In the case of node #12 the firing time was 6 time units while the node #13 was 1 time unit. In the case of node #2 it generated node #7 at a firing time of 1 unit. When all the nodes of the 0-time group were evaluated, the time variable (Time Var) headed to the next value (1 time unit). The second group of nodes was evaluated starting in node #7 which in this case generated the S-old node #12 and the node #16. If we put focus on the S-old node #12 it can be appreciated that it had been already generated by node #6 which produced it at a firing time of 6 units while the same S-old node can be generated by node #7 with a time value of 2 time units!.

The latter situation illustrates the shortage that was incurred by the old algorithm. In order to follow the principle of the time line, node #12 should have been evaluated when the leading variable reached 2 time units instead of 6 which happened with the old approach. The first improvement to the new algorithm aims to overcome this shortage, and it is explained in the following subsection.

### 4.1. IMPROVEMENT A: UPDATING THE TIME LINE

A procedure which analyzes on-the-fly whether a node is better or not has been developed in order to avoid the shortage discussed in the previous sub-section.

The improved algorithm will use the same elements of the old TLS algorithm but during the evaluation it will

verify that the firing time of the correspondent nodes is certainly the smallest one.

The verification is performed every time a group of nodes has been evaluated. Therefore it is assured that the firing times used so far have the earliest values.

Figure 2 illustrates graphically the procedure performed by the algorithm to overcome the shortage.

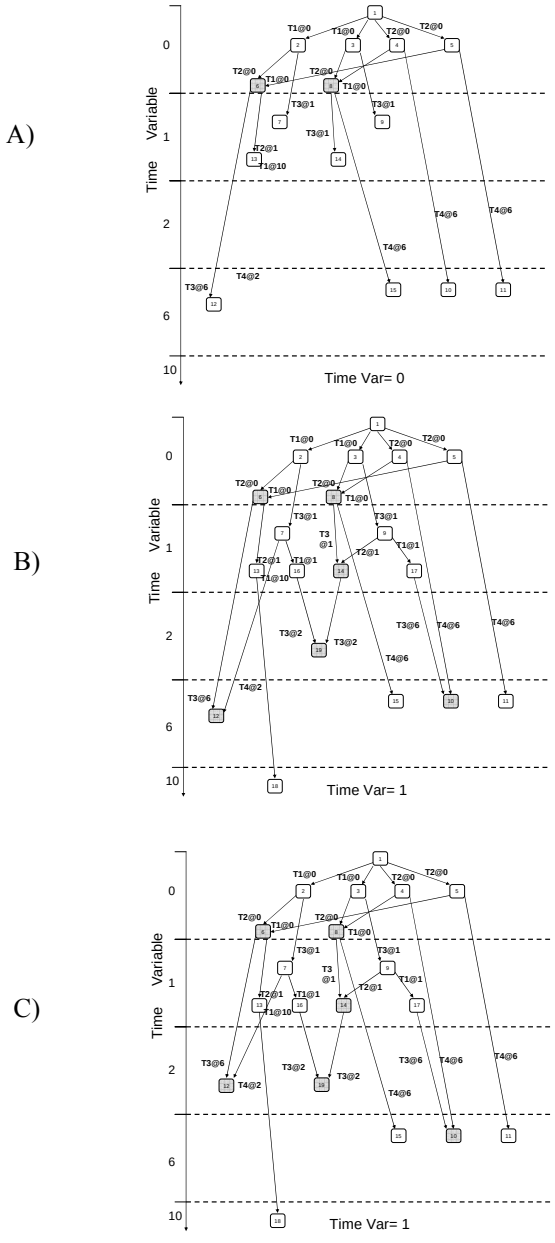


Figure 2: Generation of SS (overcoming the shortage)

Figure 2A represents the state space when the time variable has the value 0. During this instant of time the first nodes to be evaluated are nodes #1, #2, #3 #4 and #5 which generate nodes #6 to #15. During these evaluations nodes #6 and #8 are generated with a firing time of 0 units. The time variable value will not change until all the nodes that fall in the group of 0-firing-time value have been evaluated including nodes #6 and #8 (which were generated during the first evaluation). The rest of the nodes take their place in the list waiting for their evaluation time to come.

When all the nodes of the 0-firing-time value group have been evaluated, the S-old nodes generated so far (the gray-shaded nodes in the figure) must be analyzed in order to determine if they have the smallest firing-time value for the successive evaluations. In Figure 2A the S-old nodes #6 and #8 are generated with the 0-firing-time value from their father nodes.

Figure 2B represents the next step in the evaluation procedure, the time variable heads to the new value (1 time unit). The group that matches the new time value is evaluated following the sequence #7,#9,#13 and node #14. The resulting state space from that evaluation sequence is illustrated in Figure 2B. It can be appreciated that four new S-old nodes have been generated during this evaluation, nodes #10, #12, #14 and #19.

In this figure, node #12 represents the S-old node that has been generated when node #6 was evaluated.

When all the nodes from the group of 1-firing-time value have been evaluated, the firing times of the new S-old nodes are verified in order to ensure that the firing time value is the smallest possible:

- The algorithm takes the list of the S-old nodes and makes a comparison between the firing times of the nodes that generate an S-old node.
- If it is found a firing time that is less than the original firing time, it will update the time values of the found node (time stamps and global time) and the information related to the father node that generated the node will be replaced with the one that produce the best time values. If the node has not been evaluated yet it will be removed from the group that originally belonged to and it will take its position into the new group. In the case of node #12 its father node will be changed from node #6 to node #7, Figure 2C.
- If the node has been already processed then it will update its time values and afterwards it will update the time values of the branch that hangs from the node.

Implementing this analysis strategy, it is ensured that the evaluated nodes in the generation phase of the CTSS use the smallest time values.

It can be argued that the branch-updating operation would need a lot of operations in order to perform the updating; but it is expected that the number of nodes to be updated are few since they have been evaluated accordingly to the progress measure, therefore the branch is not big compared to the total amount of nodes to be generated.

### 4.2. IMPROVEMENT B: Differentiating similar markings

A problem arises when there are groups of nodes that result difficult to distinguish based upon their

potential to improve the final node. Due to this condition it is not an easy task to decide which father node must be maintained for the subsequent evaluations. The latter situation shows up when two states are fired at the same global time and the operations take similar time; in that case the resulting markings will represent different logistic states but the time stamps of both markings will have similar values. In order to establish a difference between the conflicting nodes the following implementation was developed:

- If two states have the same firing time, the one with the earliest time stamp will be selected. This action assumes that this token will be ready earlier than the one from the other state.
- If the two characteristics hold (the firing time is the same and the earliest time stamp is the same) then it will be selected the second earliest time stamp and so on until it is found one time stamp that differentiate both markings.

The previous implementation assumes that all the tokens that compose the markings have the same probability of enabling a transition of the model. Figure 3 gives an example of the developed procedure to distinguish between two nodes.

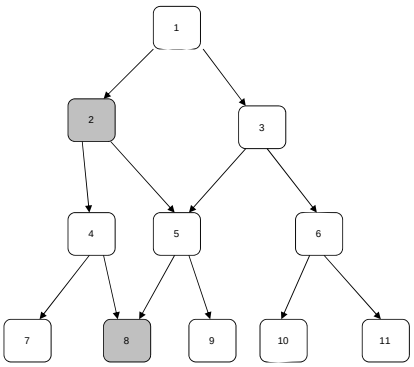


Figure 3: Differentiating two markings

In this example the S-old node #8 can be generated from nodes #4 and #5. Both nodes have the same firing time (3 time units) therefore they can only be differentiated evaluating their time stamps. Using the previous approach, the time stamp values that are taken into account for the decision have the values 3,4,5 for the S-old node that can be generated from node #4 and 3,5,6 for the S-old node that can be generated from node #5. The time stamp that differentiates both nodes is the second one thus in this example node #8 would be evaluated using the time values generated from node #4. If the S-old node #8 were originally generated from another node than node #4 then node #8 would be updated with the new time values. If the node has not been evaluated yet then the updating process ends in that node; if there were a sub branch that hanged from the node then the time values of the complete branch must have be updated.

### 4.3. Algorithm of the improved time line algorithm

In this section the flowchart of the new time line algorithm is presented.

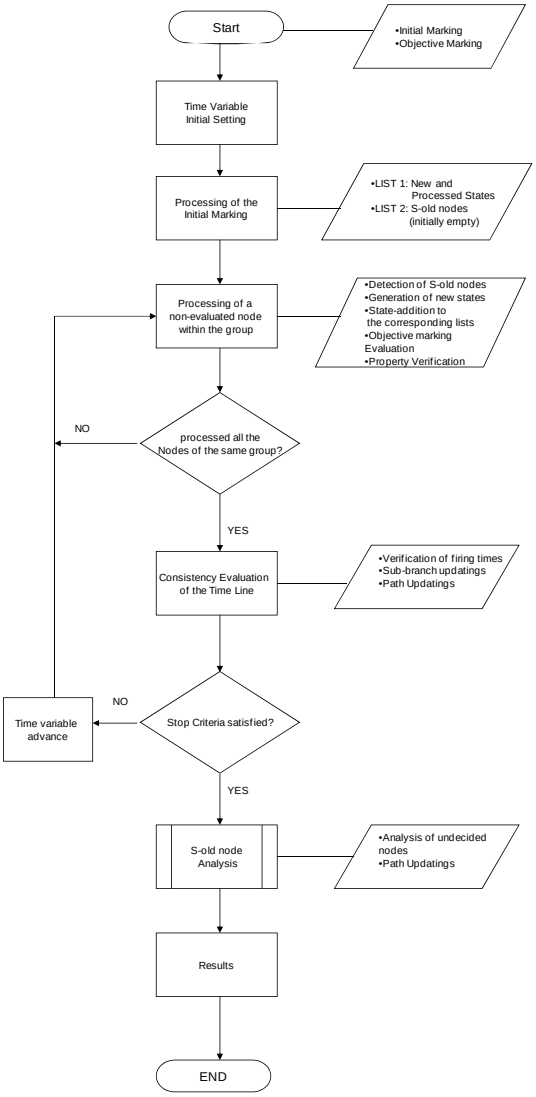


Figure 4: Improved time line search algorithm

This algorithm presents some advantages that come from the old algorithm and from the one in two-phases (Mujica and Piera 2009a):

- Good information management to store all the generated information
- Property verification can be performed prior to the optimization phase
- Good initial feasible path

The *consistency evaluation* step of the time line verifies if the S-old nodes to be evaluated certainly have the smallest possible firing time. If that is not the case the correspondent time value switching is performed and the sub-branches are updated if there are any.

The *S-old node analysis step* of Figure 4 evaluates in a second phase the generated S-old node list in order to optimize the feasible path whenever is needed. In this case it is fair to mention that the initial path can be used if a fast response of the algorithm is needed (Mujica and Piera 2010).

### 4.4. Experimental Results

In order to verify the improvements achieved with the new algorithm, it has been tested with small workloads of the CNC machine (Mujica and Piera 2009b). The proposed workloads are small enough to be stored in the computer memory therefore the achievements can be appreciated when a comparison is made testing three different algorithms: the two phase algorithm, the old TLS algorithm and the new one. Table 1 presents the results obtained from the performed optimizations.

Table 1: Obtained results from the benchmark

| Final Marking of the Buckets Place node | Obtained Makespan DFS Approach | Obtained Makespan *Old* TLS Approach | New TLS Implementation | Structural Information of the CTSS |
|---|---|---|---|---|
| Buckets: 2'(1,1,1,2,6,6,215,220) | 1,039 sec. | 635 sec. | 590 sec. | No. Nodes: 19,232 No. Arcs:59,610 No. OLD Nodes:15,431 Levels: 53 |
| Buckets: 3'(1,1,1,2,6,6,215,220) | 1,335 sec. | 960 sec. | 833 sec | No. Nodes: 172,242 No. Arcs:765,177 No. OLD Nodes:145,911 Levels: 84 |
| Buckets: 2'(1,1,1,2,6,6,215,220) +1'(2,1,3,4,6,6,120,120) | 955 sec. | 821 sec. | 720 sec | No. Nodes: 562,799 No. Arcs:1,800,951 No. OLD Nodes:471,939 Levels: 81 |
| Buckets: 2'(1,1,1,2,6,6,215,220) +1'(2,1,3,4,6,6,120,120) +1'(3,1,5,6,6,6,540,540) | 1,913 sec. (500,000nodes explored) | Unable to reach the Objective State | Unable to reach the Objective State | No. Nodes: --- No. Arcs:>2,541,330 No. OLD Nodes:>676,223 Levels: 105 |

It can be appreciated that making the implementations presented in this article the new TLS algorithm gives a better makespan. It can also be appreciated that in all the presented cases the new algorithm outperforms the previous ones.

## 5. CONCLUSIONS AND FUTURE WORK

The scheduling of industrial systems is a challenging problem due to the combinatorial nature present in most of them. The experiments with a timed coloured Petri net model of a CNC machine shows that the makespan is improved when the nodes are evaluated on a smallest-firing-time basis. The implementations presented in this article are crucial in order to develop an approach such as the time line search algorithm which uses the CTSS as the search space for performing the optimization of the makespan of industrial models.

In order to have the capacity to evaluate big state spaces a garbage collection algorithm for the CTSS is needed. This algorithm is being part of the current research of the authors.

## REFERENCES


Chiola, G., Dutheillet, C., Franceschinis, G., Haddad, S., 1997. A Symbolic Reachability Graph for Coloured Petri Nets. *Journal of Theoretical Computer Science*, vol.176 (1-2), pp. 39-65.

Christensen, S., Jensen, K., Mailund, T., Kristensen, L.M., 2001. State Space Methods for Timed Coloured Petri Nets. *Proc. of 2nd International Colloquium on Petri Net Technologies for Modelling Communication Based Systems*, pp. 33-42, Berlin.

Jensen, K., 1997. Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use. vol. 1 Springer-Verlag, Berlin.

Jensen K., T. Mailund, L.M. Kristensen,2001. State Space Methods for Timed Coloured Petri Nets. *Proceedings of 2nd Intenational Colloquium on Petri Net Technologies for Modelling Comunication Based Systems*, Berlin

Kristensen, L.M., Mailund, T., 2002. A Generalized Sweep-Line Method for Safety Properties. *FME*, Springer-Verlag, pp. 549-567, Berlin- Heidelberg.

Moore, K.E., Gupta, S.M., 1996. Petri Net Models of Flexible and Automated Manufacturing Systems: A Survey. *International Journal of Production Research*, Vol. 34(11), pp. 3001-3035.

Mujica, M.A., Piera M.A., (2009a). A Two Step Algorithm to Improve Systems Optimization based on the State Space Exploration for Timed Coloured Petri Net Models. *Proc. of the TiStoWorkshop,* Paris, France, pp.47-61.

Mujica, M.A., Piera M.A., (2009b). Performance Optimization of a CNC Machine through exploration of Timed State Space. *Proc. of the International Modelling Multiconference (I3M*), Tenerife, Spain, pp.20-25, 23-25 Sept.

Mujica M.A., Piera M.A., Narciso M., 2010. Revisiting state space exploration of timed coloured petri net models to optimize manufacturing system's performance. *Simulation Modelling Practice and Theory*, Vol.18, 9, p.p. 1225-1241.

Mujica M.A., Piera M.A., 2010. Time Line Search for the State Space-based Optimization Algorithm for Timed Coloured Petri Nets. *Proc. of MCPL IFAC'10*, Coimbra, Portugal, 8-10 Sep 2010.

Valmari, A.,1998. The State Explosion Problem. *Lecture Notes in Computer Science*, vol. 1491, Springer-Verlag, London, pp. 429-528.

Wolf, K., 2007. Generating Petri Net State Spaces. *Proc. of the 28th int. conf. on applications and theory of Petri nets and other models of concurrency*, pp.29-42, Springer.