IMPROVEMENTS FOR A COLOURED PETRI NET SIMULATOR

Miguel Mujica [1], Miquel Angel Piera [1]

[1]Autonomous University of Barcelona, Faculty of Telecommunications and Systems Engineering,
Barcelona, Spain

miguelantonio.mujica@uab.es (Miguel Mujica)

Abstract

This paper presents recent developments of a Coloured Petri Net Simulator tool. This tool integrates some principles taken from constraint logic programming as well as some search algorithms in order to make the performance of the tool as efficient as possible, some improvements have been obtained for the reduction of time for the transition evaluation task, as well as the search method trough the states space. The tool has been coded in an interface which makes the Coloured Petri Net easy to implement, as well as the results generated during the exploration easy to being evaluated and presented.

Keywords: Coloured, Petri Nets, Search, CLP, Coverability Tree, Simulation, VBA, Excel.

Presenting Author's biography

Miguel A. Mújica Mota was born in Mexico City. He studied chemical engineering at Autonomous Metropolitan University in Mexico City, and a Master's in Operations Research in the National Autonomous University of México, actually he is a pH Student at Autonomous University of Barcelona. His professional activities are in manufacture and production planning fields. His research interest focus on optimization techniques for Coloured Petri Nets aimed to solve industrial problems.

1 Introduction

Since its development(1962), Petri Nets have been widely used as a formalism for modeling systems; they have some properties that make them very useful for modeling some characteristics of manufacture and logistics systems (i.e., concurrency, parallelism, time modeling) thus they have been widely used for modeling discrete event systems.

Some variations of Petri Nets have been developed for simulation purposes or systems analysis and some fit for one purpose or another; the Coloured Ones (CPN) are the kind of Petri Nets that have the feature that attributes can be attached to tokens (colors), and with this characteristic the models developed are more compact than if they were developed with the classical ones making them very useful for the analysis of cause and effect relationship, as well as all the dependence between events [1].

Due to that, the Coloured Petri Net formalism is considered as a good one to code a CPN simulator which enables the analysis of CPN models as well as the evaluation of the different states of the system in study.

One key factor in the development of an efficient CPN simulator is the transition evaluation [2, 3]. Some authors have faced the problem of improving the speed of evaluation testing some data structures or modifying the selection algorithm for the tokens which enable the transition, generally they use the restrictions in a passive way, that is choosing the combinations of tokens which enable the transition, and after that checking whether the combination violate the restrictions or not [2,3,4]. The approach presented in this paper differs from the ones that are reported since it focuses in the best way of integrating the logic of the CPN (input nodes, arcs expressions, transitions, etc) with the variable restrictions imposed by the Guards. It has been found that using some principles taken from CLP is possible to use the present restrictions in the Guard expressions during the transition evaluation phase to avoid the unfertile combination of tokens in the input places of a transition. The transition evaluation algorithm uses some principles taken from constraint logic programming (CLP) mainly constraint propagation and scenario reduction to avoid infertile combination of tokens and make the transition evaluation as efficient as possible, firing only the combinations which enable the transition due to the tokens colours without violating any of the restrictions imposed to the variables (GUARDS).

One of the main analysis tools available for CPN is the one known as Coverability Tree, which is a logistic tree of the different states that can appear in the CPN model. This tree stores the relationship between parent-children states (causal relationship), events fired, among others. Some problems arise with the opening of the tree; mainly the growth of the branches of the tree and time involved in the process, the time spent in searching for states in the tree that have already appeared during the evaluation (OLD NODES), as well as the amount of CPU memory required to store all the information that can appear in the state space.

Some search algorithms have been integrated with the CPN features in order to reduce the time spent to explore all the branches in the tree in order to find states which have already appeared or not, and avoiding to repeat information previously stored.

2 CPN Transition Evaluation

2.1 Transition Evaluation

It has been stated that one of the main processes to be optimized in order to have good performance of a CPN simulator is the transition evaluation [3]. The classical approach of making this evaluation is the "generate and test" type, which evaluates all possible combinations of tokens within input places in a transition, and *a posteriori* the evaluation of constraint violations is done. Using this approach, CPU time is wasted testing combinations of tokens which enable some arc expressions, but violates variable constraints (Guards).

The transition evaluation in this tool uses variable constraints in an active way in order to reduce the sets of tokens to be evaluated of the input places and thus reducing the time inherent in the evaluation of each transition.

2.2 Algorithm of Evaluation

The algorithm uses the elements of CPN (colours, Guards, Arc Expressions) in an active way in order to make the transition evaluation as fast as possible. It uses some principles taken from CLP mainly Constraint Propagation and scenario reduction rules [5]. The process of evaluation is done as follows:

The first task the algorithm makes is the construction of a subset for every input place, this is done in a passive way only comparing the arc expression versus the colours of the tokens present in the input place, during this task, it can be discarded the tokens which do not enable the transition, and in the case that none of the tokens fit the requirements of the arc expression in any place, then the transition is not enabled, and the dynamic evaluation of tokens is stopped. Figure 1 shows the first approach for constructing the subsets.

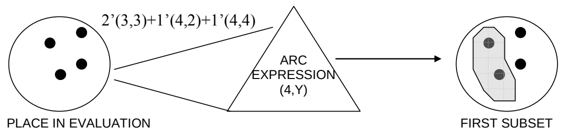


Fig. 1 The Creation of the first subset

The dynamic evaluation of tokens starts evaluating the left most place of the input places, evaluating each one of them in a sequential way, starting from the first one in the list of available tokens.

When one token is chosen, the variable allocation is made assigning the colour value to the corresponding variable of the arc expression and then the value propagation is done for every related variable within the arc expression using the restrictions in the Guards.

The value is propagated to the corresponding variable and input place using the relationship stated by the restrictions in the Guard. When the value propagation is made and the expression evaluated, the result is used to form a new subset in the correspondent input place, which satisfies the restriction and the arc expression. Fig.2 shows the procedure of assigning values and propagating the restrictions.

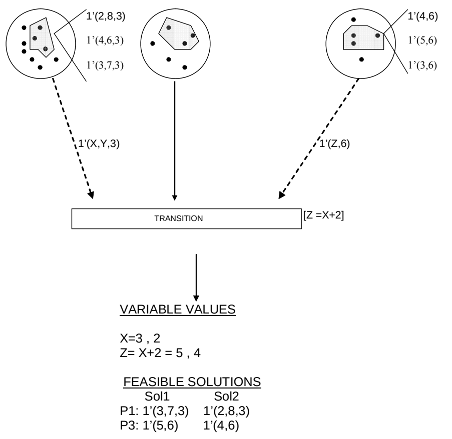


Fig. 2 The Constraint Propagation Process

When the algorithm finally reaches the last input place (the right most), all the arc expressions and restrictions must be satisfied using the token in the first place as a pivot.

After the procedure has reached the right most place, all the tokens which conform the subsets in the input places are the ones that are valid for enabling the transition and do not violate the restrictions imposed by the Guards. Then the transition fires with all the possible combinations between the tokens that form these subsets.

Fig 3 represents the direction of the propagation as well as the reduction obtained as evaluation of places advances during the transition evaluation, obtaining only the feasible combinations of tokens that enable the transition and satisfies the restrictions in the guards (shaded areas).

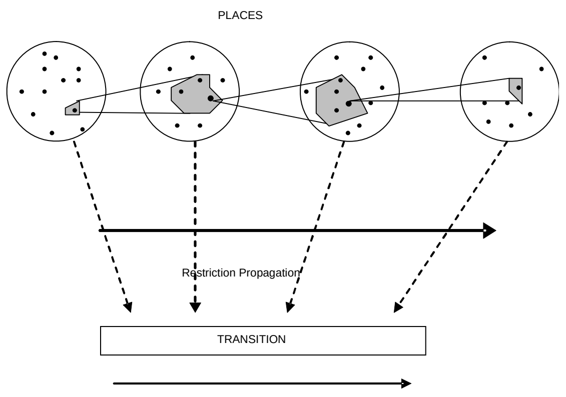


Fig. 3 The Feasible Combinations

This process of evaluation is made for every token present in the first subset in the first place.

An empirical measurement has been done for the performance evaluation of the simulator and to contrast the difference between the generate and test approach versus the one proposed here. The difference is outstanding as it can be seen in Table 1 where shows the time taken to open 200 nodes of the coverability tree.

TABLE. 1 Time performance with different transition evaluation methods

EVALUATION TYPE	NODES EXPLORED	TIME SPENT FOR EXPLORATION	TIME REDUCTION OBTAINED
Generate & Test	200	50.4 sec	---
CLP Algorithm	200	11.6 sec	77 %

3 Coverability Tree and Information Management

One problem that arises with the opening of the coverability tree is the memory overload due to exponential growth in most of the typical problems [6]. Therefore it is important to develop a special way of storing the information generated during the simulation run in order to maintain all the information of the tree (parent-children relationship, number of tokens in places, all the marks appeared, etc). The information of each mark in the coverability tree is decomposed in order to take advantage of some characteristics that can be helpful to avoid repeated information and therefore save memory and avoid the overload of memory. The next section explains how the information if managed.

3.1 Markings and CPU Memory Management

To tackle the problem of the tree growth is necessary to develop a special way of storing the marks in every part of the tree (place nodes, cardinalities, relations between nodes). This is done decomposing the marking in 2 blocks of information. The first level of information (*Colours Level*) groups the information of cardinalities that have the same colours in common, storing once the colors that appear in one place, and holding the cardinalities that appear with this colour in other block of information, maintaining always a relation between them. Managing the information of tokens this way, is possible to avoid the duplication of colour information, since it is common in any colored Petri net simulator that the colours of the tokens appear several times while the only characteristic that changes is the cardinality of the colour (number of tokens with the same colour). The figure 4 outlines the idea of grouping the information related to the token colours.

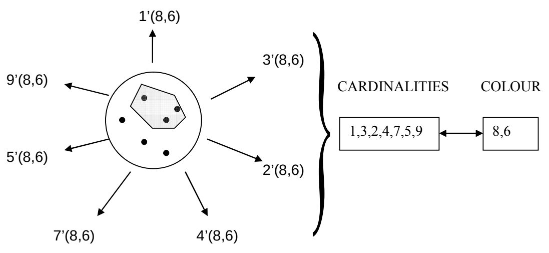


Fig.4 Grouping the token information

The second level of information (*Relations Level*) relates the information present at the colours level and the cardinalities that are in use for a specific marking thus conforming a state or mark of the CPN model with one relation for the different colours. Having this relationship done at the relations level, any mark that appears in the model can be represented with the tuple of relationship between blocks of information and the cardinalities used by the colours. It is important to remark that managing the information this way not only CPU resources are saved and the possibility of CPU memory overloaded reduced, but also the searching through the tree for the nodes is faster than if each mark would be stored in one block of information as it will be explained in the next section.

Figure 5 presents the relationship which defines a state combining the two levels of information.

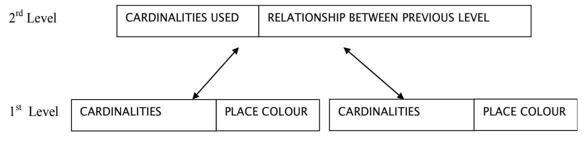


Fig. 5 The Decomposition of Information

3.2 Searching for Marks within the Tree

One key task for a simulator performance is the search of the states previously generated (OLD NODES). A typical coverability tree can have tens of thousands nodes, so as the simulator goes away opening the coverability tree, the search throughout the states already generated becomes more and more difficult to achieve, and the performance of the simulator decreases as time goes by.

This task has been faced taking advantage of the decomposition of markings. It has been integrated a dichotomical algorithm mixed with a sequential one for the search of every mark in the tree, one for the search through colours in places at colours level, and another for search through marks at the relations level. Using this approach the information about markings is stored in a static way using the unordered lists, and the ordered copies of these lists are used to accelerate the search through the state tree.

3.2.1 Search at Colours Level

At this level of information, all the different colours that appear in the places of the CPN model are stored in a list, and related to each colour there is a sequence of cardinalities that appear as simulation goes by.

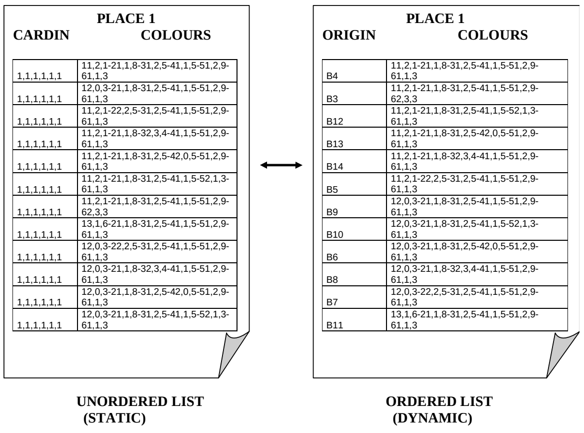
In order to find one colour in the fastest possible way, a dichotomical search has been implemented for the search of the colours in each place of the model. It is compulsory for the implementation of a dichotomical search that the information of the colours is ordered in certain way, therefore it has been created an ordered copy (which will be dynamic in nature) of the colours that appear in each place of the CPN model. The copy will store the colours that appear and with each element there is a reference to the original location in the unordered one. The creation of two copies is necessary since the ordered list will be reordering the elements as the simulation goes by and each time a new node is found.

Every time a token appears in a place, it will be used a dichotomical search through the ordered list to find the colours of the token and in the case that the token colours is found, the reference to the location in the unordered list is used to locate the sequence of cardinalities that have already appeared with that token colours, this second search is done in a sequential way. In the case that the token colours is not found, then it is stored in the last position of the unordered list with its corresponding cardinality, and its position is searched and inserted in the ordered list to keep it arranged for future searches.

At this level is possible again to take advantage of the decomposition of the mark, since once a colour is not found is possible to assure that this mark is a new one, and using a Boolean variable of the status of the mark (new or old) is possible sometimes to avoid making the search at the relations level and save some CPU time.

3.2.2 Search at Relations Level

At this level of information the same approach is implemented, one unordered copy for the relations between places of the colours level, and an ordered one for the dichotomical search of relations. At this level the Boolean factor obtained from the search at the colours level (whether the mark is new or not) is useful to decide if the search should be made or not. If the mark is considered as a new one, the resulting relation is added directly to the unordered and ordered lists without making any search. On the other hand if it is not known that the mark is new, the dichotomical search through the ordered list is done. If the mark is not found (it is a new mark), then it is inserted in the correspondent position in the ordered list, and it is stored in the last position of the unordered one. Figure 6 shows the relationship between the two lists.



PLACE 1

CARDIN	COLOURS
1,1,1,1,1,1	11,2,1-21,1,8-31,2,5-41,1,5-51,2,9-61,1,3
1,1,1,1,1,1	12,0,3-21,1,8-31,2,5-41,1,5-51,2,9-61,1,3
1,1,1,1,1,1	11,2,1-22,2,5-31,2,5-41,1,5-51,2,9-61,1,3
1,1,1,1,1,1	11,2,1-21,1,8-32,3,4-41,1,5-51,2,9-61,1,3
1,1,1,1,1,1	11,2,1-21,1,8-31,2,5-42,0,5-51,2,9-61,1,3
1,1,1,1,1,1	11,2,1-21,1,8-31,2,5-41,1,5-52,1,3-61,1,3
1,1,1,1,1,1	11,2,1-21,1,8-31,2,5-41,1,5-51,2,9-62,3,3
1,1,1,1,1,1	13,1,6-21,1,8-31,2,5-41,1,5-51,2,9-61,1,3
1,1,1,1,1,1	12,0,3-22,2,5-31,2,5-41,1,5-51,2,9-61,1,3
1,1,1,1,1,1	12,0,3-21,1,8-32,3,4-41,1,5-51,2,9-61,1,3
1,1,1,1,1,1	12,0,3-21,1,8-31,2,5-42,0,5-51,2,9-61,1,3
1,1,1,1,1,1	12,0,3-21,1,8-31,2,5-41,1,5-52,1,3-61,1,3

UNORDERED LIST
(STATIC)

PLACE 1

ORIGIN	COLOURS
B4	11,2,1-21,1,8-31,2,5-41,1,5-51,2,9-61,1,3
B3	11,2,1-21,1,8-31,2,5-41,1,5-51,2,9-62,3,3
B12	11,2,1-21,1,8-31,2,5-41,1,5-52,1,3-61,1,3
B13	11,2,1-21,1,8-31,2,5-42,0,5-51,2,9-61,1,3
B14	11,2,1-21,1,8-32,3,4-41,1,5-51,2,9-61,1,3
B5	11,2,1-22,2,5-31,2,5-41,1,5-51,2,9-61,1,3
B9	12,0,3-21,1,8-31,2,5-41,1,5-51,2,9-61,1,3
B10	12,0,3-21,1,8-31,2,5-41,1,5-52,1,3-61,1,3
B6	12,0,3-21,1,8-31,2,5-42,0,5-51,2,9-61,1,3
B8	12,0,3-21,1,8-32,3,4-41,1,5-51,2,9-61,1,3
B7	12,0,3-22,2,5-31,2,5-41,1,5-51,2,9-61,1,3
B11	13,1,6-21,1,8-31,2,5-41,1,5-51,2,9-61,1,3

ORDERED LIST
(DYNAMIC)

Fig. 6 An example of the information lists.

It is remarkable that using this strategy, an important reduction in the performance time is achieved. Since the number of operations for a dichotomical search take proportional time to the logarithm of elements in the list (for *n* elements, the operations run in O (log n) worst-case time) the total time spent in finding the colours and the relations between places is reduced drastically compared with a strategy which uses a sequential search (the same operations would take O (n) worst-case time to find an element) [7]. The table 2 presents the time reductions obtained to explore the whole tree of a CPN model for a Job-Shop 6x6.

TABLE. 2 Time performance with different searching methods.

SEARCH DONE	NODES EXPLORED	TIME SPENT FOR EXPLORATION	TIME REDUCTION OBTAINED
Sequential Search	117,600	5.5 hrs	---
Dichotomical Search	117,600	1.5 hrs	72 .7 %

4 User's Interface

The CPN simulator has been coded in VBA which allows using some Microsoft applications as a user's interface. In this particular case it has been implemented with EXCEL since at industry level is very common the use of this software and the coding of a CPN become more familiar than if it were implemented in a different interface.

4.1 Simulator Engine Implementation

The simulator engine is coded using the VBA feature available in EXCEL, and the tool is available for the user as a macro, which can be easily called from the tool bar.

The user must code a CPN model using four worksheets. The first two worksheets are used for defining the initial mark of the model (root node in the coverability tree). The third worksheet is used to code the structure of the CPN model, and the last one is for the definition of the relationship between variables (Guards in the CPN model).

All the operations performed by the simulator engine during a simulation run are invisible to the user and the results are displayed *a posteriori* in the two first worksheets.

With the state space generated is possible to analyze the results using a display interface also coded in VBA. Using this schema the End-User interacts with the simulator only at the definition of model and for the analyzing of the results as can be seen in the figure 7.

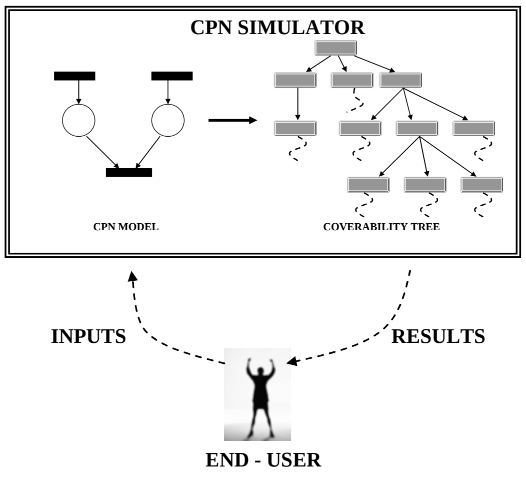
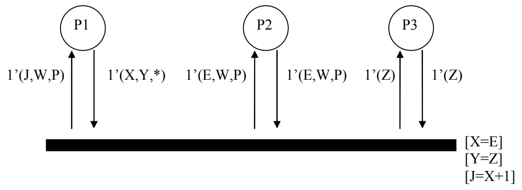


Fig.7 User Interaction with the Simulator.

4.2 Applicability for Real Problems

The simulator tool is for general purpose, and the type of problems that can be solved using this methodology are quite broad since it depend only that the CPN model is formulated in accordance to the syntax defined by the macro. Obviously the tool performance varies depending on the model size, number of transitions, number of tokens in places, etc. It was tested for the Job-Shop 6x6 [8] modeled with a CPN which is shown in figure 8.



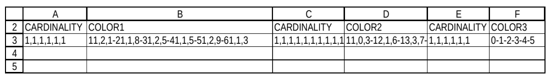
**Mo=[1'(11,2,1)+1'(21,1,8)+1'(31,2,5)+1'(41,1,5)+1'(51,2,9)+1'(61,1,3),
1'(11,0,3)+1'(12,1,6)+1'(13,3,7)+1'(14,5,3)+1'(15,4,6)+1'(16,8,1)+1'(21,2,5)+1'(22,4,10)+
1'(23,5,10)+1'(24,0,10)+1'(25,3,4)+1'(26,8,1)+1'(31,3,4)+1'(32,5,8)+1'(33,0,9)+1'(34,1,1)+
1'(35,4,7)1'(36,8,1)+1'(41,0,5)+1'(42,2,5)+1'(43,3,3)+1'(44,4,8)+1'(45,5,9)+1'(46,8,1)+
1'(51,1,3)+1'(52,4,5)+1'(53,5,4)+1'(54,0,3)+1'(55,3,1)+1'(56,8,1)+1'(61,3,3)+1'(62,5,9)+
1'(63,0,10)+1'(64,4,4)+1'(65,2,1)+1'(66,8,1) ,
1'(0)+1'(1)+1'(2)+1'(3)+1'(4)+1'(5)]**

Fig. 8 CPN model for the Job-Shop 6x6

This problem is modeled with three places and one transition, where the first place P1 holds the information about the status of each one of the jobs using three colours to hold these information, for example the first token with colours 11, 2, 1 means that the first job is doing the first task (11), using the machine number two (2), and the time spent doing that task lasts one time unit (1). The second place holds the tokens which carry logic information for the model in the colours, the case of the first token with colours 11,0,3 means that the job 1 doing the first task (11) needs the machine number zero (0), and the time of that task will use three time units (3). The place P3 hold the tokens which represent the availability of machines, the first token means that at the present state, the machine number zero is available to be used (0). Finally using the Guards, the relationships between variables are fixed, and the transition is completely specified.

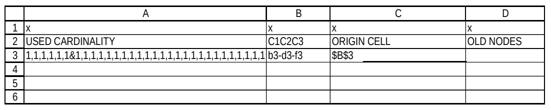
4.3 Simulator Interaction

As it has been previously mention, the interaction between the end-user and the simulator tool is via EXCEL worksheets, and the results presented by the tool uses also the worksheets. For any CPN model, the structure definition of the model and the initial mark must be done using the decomposition principles mentioned in section 3.Figure 9 shows how the initial mark is defined using the worksheets for the CPN model of figure 8.



	A	B	C	D	E	F
2	CARDINALITY	COLOR1	CARDINALITY	COLOR2	CARDINALITY	COLOR3
3	1,1,1,1,1,1	11,2,1-21,1,8-31,2,5-41,1,5-51,2,9-61,1,3	1,1,1,1,1,1,1,1,1,1	11,0,3-12,1,6-13,3,7-	1,1,1,1,1,1	0-1-2-3-4-5
4						
5						

WORKSHEET ONE



	A	B	C	D
1	x	x	x	x
2	USED CARDINALITY	C1C2C3	ORIGIN CELL	OLD NODES
3	1,1,1,1,1,1&1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1	b3-d3-f3	B3	
4				
5				
6				

WORKSHEET TWO

Fig. 9 The two levels of Information (Worksheets)

Therefore the end-user must make the decomposition of the first mark (root node) in order to start the simulation.

5 Results and Conclusions

In the case of transition evaluation it is clear that using the CLP principles in accordance with the CPN logic is possible to achieve important reductions in evaluation time as it has been shown in section 2.

In addition, the tool has been tested with the model for the Job-Shop 6x6 [8] and the time to explore the whole coverability tree were reduced in 72 % when using the new search technique with the dichotomical algorithm implemented. Therefore it is clear that using this approach for the searching through the colours and relations lists, the overall performance of the simulator reduces dramatically the time spent for the process of exploring and opening the tree. Is fair to mention that this kind of methodology for coding a CPN simulator has the advantage that it can be used as an optimization technique as well as only as a simulator, since it is possible to explore all the possible states of the model (coverability tree), while the ones that are available generally function as an evaluation technique where the transition evaluation chooses the tokens which enable a transition in a random way [2, 8].

6 References

[1] Jensen, K 1997. “Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use”. Vol. 1 Springer-Verlag. Berlin.

[2] Gaeta R. 1996, “Efficient Discrete-event Simulation of Coloured Petri Nets”. In Transactions on software Engineering vol 22

[3] Sanders M.J. 2000 “Efficient Computation of Enabled Transition Bindings in High-Level Petri Nets.” In Proceedings if the 2000 IEEE International Conference o Systems

[4] Bisgaard, Haag, T. , Rudmose T, “Optimizing a Coloured Petri Net Simulator”, University of Aarhus, 1994

[5] McAllester, David. 1992 “Constraint Satisfaction Search” in Artificial Intelligence Lecture Notes

[6] Gambin A, Piera M. A, Riera D., “A Petri Nets Based Object Oriented Tool for the Scheduling of Stochastic Flexible Manufacturing System” IEEE, 1999

[7] Mehlhorn, K, “Data Structures and Algorithms2,Graph algorithms and NP-completeness”, Berlin,Springer-Verlag 1984

[8] Dauzère-Pères, S., Lasserre, J.B, “An integrated approach in Production Plannig and Scheduling”, Lecture Notes and Mathematical Systems, Springer.Verlag, Berlin., 1994