OPTIMIZING TIME PERFORMANCE IN REACHABILITY TREE- BASED SIMULATION

Miguel Mujica[(a)], Miquel Angel Piera[(b)], Mercedes Narciso[(c)]

Autonomous University of Barcelona, Department of Telecommunications and Systems Engineering.
Barcelona, Spain.

[(a)]MiguelAntonio.Mujica@uab.es, [(b)]MiquelAngel.Piera@uab.cat, [(c)]Mercedes.Narciso@uab.es,

ABSTRACT
This paper presents the highlights of a two-step algorithm based on the reachability tree for optimizing industrial system models coded with the Coloured Petri Net formalism. The first step of the algorithm consists in generating all the possible states that can be reached by the system, and the second step of the algorithm focuses on updating the time stamp values of the most promising state subspaces of the system. Some tests with benchmarking problems have been made to evaluate the efficiency of the second part of the algorithm, and the results obtained are analyzed in order to identify the algorithm processes that need improvements. Based on the results, an heuristic has been developed to improve the performance with a better choice of states to be expanded during the generation of the reachability tree.

Keywords: Coloured Petri nets, reachability tree, simulation, decision support systems, optimization time.

1. INTRODUCTION

Due to changes in the market rules (i.e. changes from high volume production to high diversity), during the last decade, companies shifted their methods for producing goods to more flexible schemas of production, distribution and logistics. With this transformation, companies expected to improve their productivity as more flexibility was gained by its value chain. Soon it became clear that the more flexible the processes were, the more difficult became it to coordinate all the elements of the chain (suppliers, production elements, distribution entities, etc).

Therefore, it became important to tune decision support tools that could help decision makers to coordinate in the most efficient way the different processes and elements of the system.

Several approaches from diverse knowledge areas have been developed to face some aspects of these problems. In later years, with the improvement of digital computers, digital simulation has emerged as a tool capable to give support in the decision making process of industrial systems. Petri Nets (PN) is a modeling formalism that has been applied successfully to develop accurate discrete event system models of industrial and logistic systems. It presents certain modeling characteristics which make it appropriate to be applied in these fields; in particular it supports models for concurrency, parallelism and conflicting situations. Coloured Petri Nets (CPN) is a high abstraction level modeling formalism that allows modeling very complex industrial systems in a more simplified representation (which result easier to maintain) than the model that could be obtained with Petri Nets (Jensen 1997). Both CPN and PN have been used for modeling the structural behavior of systems but they can also be employed to analyze system performance. In order to achieve this objective the formalism has been extended with time representations (time stamps, global clock) (Jensen 1997).

The reachability tree is a quantitative analysis tool used in PN and CPN which allows storing all the different states of a system in a tree-based structure. In general it is employed to verify Petri nets behavior through the exploration of the state space (Jensen, Christensen, Hard, Holzman). In the work presented in this article, the reachability tree approach has been used to develop a two step efficient algorithm with the objective to improve industrial systems performance.

1.1. The Reachability Tree

The reachability tree is a directed graph in which the nodes represent the set of reachable states of the system and the arcs correspond to the state changes. Such a graph represents all the possible system's scenarios and can be used to verify and analyze some properties of the system, but it shows some disadvantages when it is used for optimizing purposes:

- In order to detect previously appeared states, all the generated nodes must be stored in computer memory causing most of the times memory saturation due to the state explosion problem.
- The simulation performance is reduced as the memory burden grows when the storage of states takes place.
- In order to simulate the system some nodes need to be reevaluated, increasing the computational time.

- When evaluating old nodes time characteristics, an extra computational effort appears because some branches must be updated according to the combination of time stamps.

There have been some attempts to overcome the problem of memory saturation due to the exponential growth when new states are generated. Jard (1991) and Christensen (2002) developed respectively an approach that does not store all the states of the model and throws away states on the fly, minimizing the amount of computer memory used to explore the state space.

These approaches have the advantage that reduce the run-time of the state space exploration and result very efficient for analyzing and verifying system properties such as dead locks and safety properties, but the old node analysis made is not as efficient as it could be done with a different approach.

Holzmann (1998) developed an algorithm that is able to maintain all the states based in the hashing principle. This approach uses an efficient way of storing the different states of the system; however a main disadvantage appears when two different states are mapped to the same hash value (collision).

The state analysis that has been implemented in the algorithm presented in this article was based on the principles reported by Narciso and Piera (2005). It stores all state space information and evaluates only the most promising branches in the tree. This approach which does not overcome completely the problems of speed and memory, is capable to generate results analyzing the most promising branches of the tree and updating the time stamp values based on the decision rules reported in the same work. The state analysis has been combined with the fast search and efficient transition evaluation algorithms developed by Mujica and Piera (2006, 2007) resulting in an efficient tool for exploring the state space with optimization purposes, and a very promising one to solve real life industrial problems.

2. REACHABILITY TREE-BASED SIMULATOR

A simulator to optimize industrial system performance based on time analysis of old nodes stored in the reachability tree has been coded.

2.1. Time representation for the Reachability Tree

To represent time in the reachability tree, it is necessary to attach a global clock to each state besides the correspondent time stamps of the tokens that belong to the marking. To understand how time flows between states when using the time extension it is proposed to take any node in the state space (one suggestion is to take the root node), and follow one of the directed arcs to its successors (child nodes); the difference between the clocks represents the amount of time that has past when going from one node to the next one, i.e. the time that takes the system to change from one state to the other. If the same process is repeated with the subsequent nodes until getting the objective state it will be possible to obtain the complete flow time for the process. A representation of a reachability tree with its global clocks is presented in Figure 1.

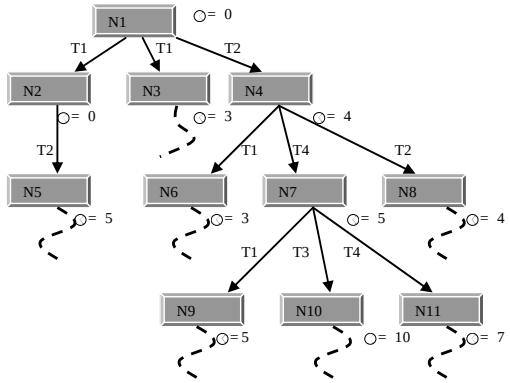


Figure 1: Time Extension for Reachability Tree

In this example the reachability tree is generated under the CPN rules (color enabling rules). Time advances during the generation of children nodes based on the rule that the global clock must take the minimal time between all time stamps of the enabling tokens (Narciso and Piera 2005). Using these rules to explore the reachability tree, it will be possible to analyze the state evolution behavior using time stamps and the global clock.

2.2. Updating Time in the State Space

After exploring the reachability tree, all the possible system states are generated (i.e. a bounded state space) and stored in the tree structure. The old nodes are stored altogether with their time characteristics in a separate list for later analysis.

When time is involved in the analysis, it is common to find old nodes which could differ one from the other in their time extension characteristics (time stamps and global clock). Since the same state can be reached from a different node, the time extension characteristics of old nodes should be properly updated in order to deal with the timed state space. However when dealing with optimization problems instead of system verification properties, it is possible to save computational time avoiding time evaluation for the old nodes which will not lead to the solution of the optimization problem.

The optimization of the paths in the state space is carried out making a time analysis for the old nodes. The algorithm compares the time stamps and global clock, and updates the time values of the best path that leads to the objective state. Narciso and Piera (2005) proposed an approach which analyses the old nodes in a way that the algorithm avoids the complete exploration of branches in the reachability tree. It updates only the ones that improve the paths which lead to the objective states (optimization). The authors proposed a rule for deciding between two old node states basing the

decision only on the time stamps of the markings. One example of how to use the mentioned rule is shown in Figure 2.

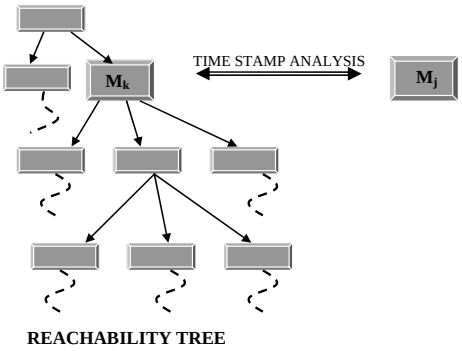


Figure 2: The time analysis of an Old Node

There are two markings representing the same state. One state M_k which has been already stored during the generation of the reachability tree is compared with another marking M_j which represents the same state but it has different time stamps values. In this case three possible results can be obtained from the evaluation:

- The time stamp values of the marking M_j are less than the time stamp values of the original marking M_k. Therefore it can be concluded that the marking M_j will produce states with better time values than the ones that are actually stored in the reachability tree. In this case the original marking M_k is replaced by the marking M_j, and the time values are updated for the best branch that goes from the original marking to the final one.
- Dealing with time minimization, the time stamp values of the marking M_j are greater than the values of the marking M_k. In this case it can be concluded that the M_k marking is better than the marking M_j and the original marking M_k will be kept without making any value updating.
- Some time stamp values in the marking M_j are better than the correspondent ones in the marking M_k and some are worse. For this outcome nothing can be concluded about the two markings. In this case a depth search must be done using the M_j time stamps values in order to get to the final state and then a decision based on the final time must be taken. If the final time resulting from using the M_j time values is better than the stored one, the marking M_j will replace the original marking M_k and the time stamps of the branch that goes from M_j to the final state.

2.3. A Two- Step Reachability Tree Algorithm

In the first step of the algorithm, the CPN rules are used to generate the complete state space. All the possible states of the system are generated together with their time values which depend mainly on the exploring sequence of the algorithm. The repeated states that appear during the exploration (old nodes) are stored in a separate list with their correspondent time stamps and global clock for a later analysis.

In the second step of the algorithm (when all the states of the system have been already generated) the stored old nodes are analyzed comparing their time stamp values with the correspondent values in the nodes stored in the reachability tree. This procedure is carried out with the purpose of finding the time values that reduce the most the final time of the sequence of states that go from the initial state to the final one. Therefore the time span for the whole procedure is optimized. The old node analysis is done following a simple order in the old node list (from the first to the last element).

The time analysis of tokens is based on the algorithm and updating rules proposed by Narciso and Piera (2005) which avoid the exhaustive analysis of all branches at lower levels in the state space when a better node is selected and the time stamps of the offspring nodes are updated.

The two steps of the algorithm are outlined in Figure 3

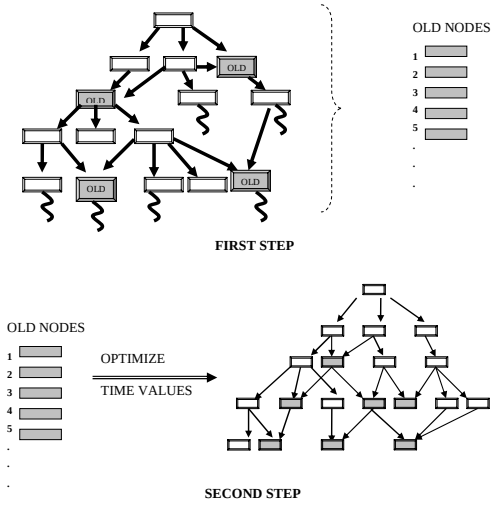


Figure 3: The Two-Step Algorithm.

In the first step the shaded nodes correspond to the old nodes that appear during the state generation, and are stored in the old node list. In the second step of the algorithm the list is evaluated focusing on the time stamp values of the nodes comparing them with the ones stored in the reachability tree, applying the evaluation algorithm previously mentioned.

3. TESTING THE SIMULATOR/OPTIMIZER

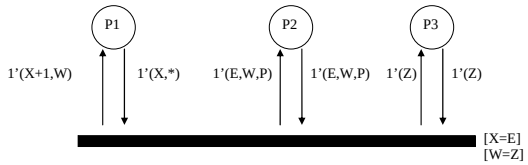
The simulator performance was tested by modeling some job-shops which appear to be well-known benchmarks (Dauzére et al 1994). Such job-shops are the 3x3, 5x5 and the 6x6 job-shops. The objective of the evaluation was to establish the efficiency of the second step of the algorithm throughout the comparison of time

stamps between repeated states when the depth exploration is done.

3.1. The Job-Shops

The job-shop in its different modalities is a benchmarking problem which consists in a certain number of jobs that must follow a specified sequence of tasks through different machines. In the 5x5 job-shop five jobs must go through processes in five different machines, the goal of these benchmarks is to obtain the sequence that minimizes the time span of the whole procedure. The different job-shop models were coded with the CPN formalism.

A typical model for a job-shop is presented in figure 4.



COLOUR	DEFINITION	DESCRIPTION
J	Int 11..33	Job and Task Identifier
W	Int 1..3	Machine Needed for the Next Job
X	Int 11..33	Job and Task in progress
E	Int 11..33	Job and Task Identifier
P	INTEGER	Time Spent for each Job
Z	Int. 1..3	Available Machines

Figure 4: The CPN Model of the 3x3 Job-Shop

The place node P1 stores the tokens which represent the information of the job and task numbers (color J) and the one about the machine needed to complete the correspondent task (color W). The place node P2 stores the tokens with the logical sequence information about the jobs in place P1 (colors E and W) and the time consumption when the task is done (color P). The last place node P3 represents the availability of the different machines (color Z). In the initial state of the system all jobs are waiting for the first task to be done, and the machines are available and ready to be used. The global clock and the time stamps are equal to zero in this state.

3.2. Second Step: Performance evaluation

The aim of the data collected is to evaluate the efficiency of the optimization process while the time stamp analysis is carried out. During the simulation, data of the following events was collected:

- The amount of repeated states (old nodes) that appeared during the exploration.
- The number of old nodes that were updated due to the comparison of their time stamp combination.
- Number of not-updated nodes (rejected nodes) due to the comparison of time stamp values.
- Number of old nodes for which it was not possible to decide about updating based on time stamp comparison. In order to come to a decision, a depth exploration was done for these cases.
- The number of nodes that needed to be updated after the depth exploration was done (because the final time was less than the original one).

The results obtained from the evaluation of the different benchmarks are presented in Table 1.

Table 1: The Simulator Performance

Job-Shop type	J-S 3x3	J-S 5x5	J-S 6x6
No. of different States	693	7,776	117,650
Old Nodes Found	2,032	24,625	487,408
Updated Nodes at Initial Evaluation	59	1,893	26,555
Discarded nodes after Initial Evaluation	485	5,829	128,387
Nodes Unable to Decide	1,488	16,903	332,475
Updated Nodes after depth explorations	11	10	35

The above data was collected during the optimization of the different benchmarks, maintaining in memory all the information for the complete state space and choosing the first available node as a depth-first search logic.

3.3. Analysis of the Experimental Results

It can be seen that in the case of the job-shops, the percentage of updated nodes after the depth exploration is 0.7%, 0.06% and 0.01% for the 3x3, 5x5 and 6x6 job-shop respectively. The previous results are important since the optimization phase takes a lot of CPU time, which can be reduced by minimizing the number of depth explorations.

It also can be seen that the time analysis becomes more complicated and the number of states which are unfeasible to decide, increase dramatically as the model becomes more complex (from around 1,500 to 16,000 and 300,000 for each of the benchmarks). Based on the result interpretation, it can be concluded that the simulator performance could be improved if the amount of irresolute nodes was reduced. The reduction was achieved through an heuristic developed for making a better selection of child nodes during the exploration of the state space.

4. IMPROVING THE EXPLORATIONS

The results presented in Table 1 show that the time analysis performance is not as efficient as it could be (it only updates very few states after making the depth exploration). A better way of selecting the nodes for the exploration is proposed. The selection rule aims to choose the best candidate among child states when the next depth level is to be explored. Selecting a better node will increase the number of rejected nodes, and the number of depth explorations will be reduced when the evaluation of old nodes takes place. In addition, it is considered that choosing good nodes during the exploration will lead to the best path in a faster way and therefore the optimization time spent in the second phase of the algorithm is reduced.

4.1. An Heuristic for Selecting the Best Nodes

The heuristic implemented was coded with the purpose of observing the improvement that can be reached if a better selection of states to be evaluated is done. A utility function that allows assigning a value for each one of the states was constructed. The function was developed on the basis that the less value the time stamps have, the less probable incrementing the clock is for the subsequent evaluations of any marking.

That is, given a M_j state of the reachability tree, let F be the function that sums the time stamps Ts_i where the index i goes from token 1 to token N_j of the marking.

$$F(M_j) = \sum_{i=1}^{N_j} Ts_i \qquad (1)$$

During the exploration of the state space, the selection of a node among child nodes is done based on the value calculated by the function (1) and the node which has the lowest value is the one to be selected. In other words, given a set of k child nodes, the next node among them to be evaluated is selected under the next criteria:

$$Min\left\{F(M_j) \ni j = 1,...,k \right\} \qquad (2)$$

The index j represents the child nodes of a given state.

An example of how to use the utility function to select the best child is presented in Figure 5.

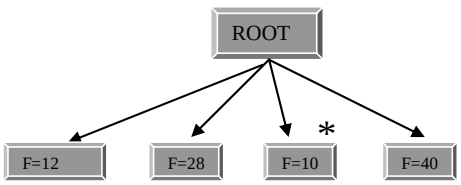


Figure 5: Using the Utility function to select the Best Node

In this figure, there is a root node with four different child nodes. If there were no utility function implemented, the first child to be evaluated would be the node on the left hand side (F=12). Using the utility function, a value assignment to each of the different child nodes is made; the child to be evaluated is the one with the lowest utility value. In this example the node with the value of 10 units is the chosen one. Therefore the utility function is used to guide the depth-first exploration algorithm for the generation of the reachability tree.

The heuristic was implemented in the algorithm, the models were run again and the data was collected to evaluate the new performance. The results obtained using this algorithm are presented in Table 2.

Table 2: Simulating with the heuristic

Job-Shop type	J-S 3x3	J-S 5x5	J-S 6x6
No. of different States	693	7,776	117,650
Old Nodes Found	2,032	24,625	487,408
Updated Nodes at Initial Evaluation	0	1,330	18,725
Discarded nodes after Initial Evaluation	2,032	10,506	213,966
Nodes Unable to Decide	0	12,789	254,717
Updated Nodes after depth explorations	0	22	34

4.2. Results

From the results presented in table 1 and 2, it can be seen that using the utility function, the selection rule is improved and as a result the performance of the exploration task is drastically enhanced. It can be seen that for the 3x3 job-shop the selection of the best path is achieved at the first step of the algorithm. This conclusion takes place since during the second phase of the algorithm all stored nodes are discarded in the old node list. Therefore it is not necessary to make a depth exploration to decide whether a time stamp updating is needed or not. In this particular case, the chosen states for opening the reachability tree are the best ones. From the results of the rest of the models (5x5 and 6x6 job-shop), it can be seen that the number of updatings practically remains the same. The analysis of the number of discarded nodes, allows concluding that the utility function improves the simulation performance.

As a result of using the heuristic, the discarded nodes increase by an amount of 5,000 for the 5x5 job-shop and by around 100,000 for the case of the 6x6 job-shop. The avoidance of the same quantity of depth explorations during the analysis of old nodes gives as a result a better performance of the simulator. The same conclusion is drawn by the analysis of the irresolute

nodes. In the case of the 5x5 and 6x6 job-shops the numbers of depth exploration is drastically reduced and the performance of the simulator is improved since less depth explorations are necessary.

4.3. Conclusions and Future Work

The reachability tree-based simulator presented in this article is a promising approach for solving real life industrial problems. It has been introduced an heuristic implemented for the first step of the algorithm which leads to exploring the reachability tree in a better way by reducing the quantity of evaluations needed in the second step. In this way time optimization is achieved. It has been also demonstrated that focusing on the best selection of nodes during the exploration of the reachability tree improves drastically the simulator performance. Due to the results presented, the development of an heuristic focusing on the selection of old nodes to be evaluated during the second step of the algorithm is proposed as a future work. It is expected that implementing such an heuristic in combination with the one introduced in this article will further increase the performance of the whole algorithm.

REFERENCES

Christensen, S., Kristensen, L.M., Mailund, T., 2001"A Sweep-Line Method for State Space Exploration"in *TACAS*, Springer-Verglag, Berlin-Heidelberg

Christensen, S, Mailund T., 2002, "A Generalized Sweep-Line Method for Safety Properties", in FME, Springer- Verlag, Berlin- Heidelberg

Dauzére-Peres, S., Lasserre, J.B., 1994, "An Integrated Approach in Production Planning and Scheduling", *in Lecture Notes and Mathematical Systems*, Springer-Verlag, Berlin.

Holzmann, G.J., 1998, "An analysis of Bitstate Hashing", in *Formal Methods in System Design*, V13 (3), p.p.301-314, November.

Jard C., Jeron, T., 1991"Bounded –memory algorithms for Verification On-the Fly", in *Proceedings of CAV 1991*, vol575 of *Lecture Notes in Computer Science*.Springer-Verlag.

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

Jensen, K, 1997, "Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use", Vol 2, Springer-Verlag. Berlin

Mujica, M., Piera, M.A., 2006 "Building an Efficient Coloured Petri Net Simulator", in *Proceedings of the International Mediterranean Modeling Multiconference*, p.p.153-158, October 4-6, Barcelona, Spain

Mujica, M., Piera, M.A, 2007, "Data Management to Improve CPN Simulation Performance", in *Proceedings of the International Mediterranean Modeling Multiconference*,p.p.53–58, 4-6 October, Bergeggi, Italy

Mujica, M., Piera, M.A., 2007,"Improvements for a Coloured Petri Net Simulator", in *Proceedings of the 6th EUROSIM Congress on Modeling and Simulation*, p.p. 237, September 9-13, Ljubljana, Slovenia

Narciso, M., Piera, M.A., y Figueras J., 2005, "Optimización de Sistemas Logísticos Mediante Simulación: Una Metodología Basada en Redes de Petri Coloreadas", Revista Iberoamericana de Automática e Informática Industrial, vol 2(4), p.p. 54-65IAII, Valencia, España.