I.J. Intelligent Systems and Applications, **2015, 01, 42-48**
Published Online December 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2015.01.04

Modeling the Scheduling Problem of Identical Parallel Machines with Load Balancing by Time Petri Nets

Sekhri Larbi
Industrial Computing and Networking Laboratory, Computer Science Department, University of Oran, BP 1524 Oran, Algeria
E-mail: larbi.sekhri@univ-oran.dz

Slimane Mohamed
Industrial Computing and Networking Laboratory, Computer Science Department, University of Oran, BP 1524 Oran, Algeria
E-mail: Slimanemd@hotmail.com

Abstract—The optimal resources allocation to tasks was the primary objective of the research dealing with scheduling problems. These problems are characterized by their complexity, known as NP-hard in most cases. Currently with the evolution of technology, classical methods are inadequate because they degrade system performance (inflexibility, inefficient resources using policy, etc.). In the context of parallel and distributed systems, several computing units process multitasking applications in concurrent way. Main goal of such process is to schedule tasks and map them on the appropriate machines to achieve the optimal overall system performance (Minimize the Make-span and balance the load among the machines). In this paper we present a Time Petri Net (TPN) based approach to solve the scheduling problem by mapping each entity (tasks, resources and constraints) to correspondent one in the TPN. In this case, the scheduling problem can be reduced to finding an optimal sequence of transitions leading from an initial marking to a final one. Our approach improves the classical mapping algorithms by introducing a control over resources allocation and by taking into consideration the resource balancing aspect leading to an acceptable state of the system. The approach is applied to a specific class of problems where the machines are parallel and identical. This class is analyzed by using the TiNA (Time Net Analyzer) tool software developed in the LAAS laboratory (Toulouse, France).

Index Terms— Scheduling Problem, Load Balancing, Time Petri Net, Identical Parallel Machines

I. INTRODUCTION

Recent works in scheduling tend to classify problems and to find techniques to reduce their complexity by developing optimal and efficient algorithms. Among all classifications existing in the literature, Convay's and Graham's classifications are considered as the most used to encode such problems [1] and [2]. In [2], a clear separation is made between the three parts of the problem: task, resource and the objective to optimize. In these works, a task is an important element and it doesn't carry about resources just using them without conflicts [3]. Nowadays with the technology advances, these traditional methods have proved to be inadequate because degrading system's performances (inflexibility, lack of strategy in the adopted solutions for use resources, etc.) .

Petri nets are a formal tool for modeling and analyzing systems that exhibit behaviors such as competition, conflicts and the causal dependencies between events. They are appropriate to model and validate discrete event systems (*DES*). They are used for modeling concurrent systems such as Occam programs [4], the design of an automated production system [5] and recently in the protocol modeling for wireless sensor network [6].

Petri nets based approaches address the issue of flexibility of tasks and resources. The modeling power and analysis power of Petri nets theory can explore easily different aspects of the scheduling problem (behavioral and structural properties) and integrate factors ignored in previous works [7].

In scheduling related literature, Petri nets theory is applied in several fields. Ramamoorthy used graphical techniques to analyze the cyclical flow in automated production systems in order to represent the scheduling problem by cyclic Petri Nets [8]. Chrétienne provides a timed Petri nets based approach for cyclic generic scheduling problem [9]. Van Der Aalst exploits Petri nets to minimize the cycle time for planning repetitive problems [10]. Korbaa proposes a colored Petri nets based approach to model cyclic scheduling and to reduce the complexity analysis [11]. In [12], a Petri net-based model for scheduling problems with a branch and bound algorithm is proposed using dynamic branching strategy and resource-based lower bound. Other works on scheduling flexible manufacturing systems using Petri-nets and genetic algorithms are proposed in [13]; and colored Petri nets [14].

In these works, performance evaluation of the system is defined only in term of the task execution ordering without conflicts and resources sharing. This fact leads to a bad system performance.

In this paper, we extend classical methods by taking into account the load balancing aspect in the system, and ensuring an acceptable operating level. The method improves the conventional Petri nets based approaches by ensuring an economic resources using and achieving an acceptable state of the system. The modeled system can be analyzed to verify some properties like: finding a feasible scheduling, detecting conflict, eliminating redundancy, finding the lower/upper limit for the total execution time.

This work focuses on the formal modeling of identical parallel machines scheduling problem by using time Petri net and the formal validation of its properties by using TiNA tool. The preliminary results reveal that the obtained model load is bounded and adjusted among the resources.

The paper is structured as follows. Section 2 introduces some definitions on scheduling problem area with an illustrative example. In section 3, we introduce briefly the time Petri nets model. In section 4, we detail our method by explaining different mapping steps translating a scheduling problem into a time Petri net. The method is applied to an illustrative example and analyzed by *TiNA* tool. Finally we conclude with a conclusion and future prospects.

II. Scheduling Problem (SP)

Scheduling can be defined as the allocation of scares resources over time to perform a collection of tasks. It is concerned with finding the optimal one.

Scheduling techniques are used to answer questions that arise in production planning, project planning, computer control, manpower planning, cloud tasks scheduling, distributed simulations, etc. [15] and [16]. Many techniques have been developed for a variety of problems and classified by categories (single machine, parallel machine, flow job, job shop, etc.). To fix the terminology, we begin by defining the general scheduling problem.

Many problems fit into this definition. In essence, scheduling boils down to the allocation of resources to tasks over time. Some authors refer to resources as 'machines' or 'processors' and tasks are called 'operations' or 'steps of a job'. Resources are used to process tasks. However, it is possible that the execution of a task requires more than one resource, i.e. a task is processed by a resource set. Moreover, there may be multiple resource sets that are able of processing a specific task. The processing time of a task is the time required to execute the task given a specific resource set. By adding precedence constraints it is possible to formulate requirements about the order in which the tasks have to be processed. We will assume that resources are always available, but we shall not necessarily assume the same for tasks. Each task may have a release time, i.e. the time at which the task becomes available for processing. A schedule is an allocation of resources to tasks over time and for each task it is specified when it is processed and which resources process it.

A. Definitions

Formally, an *SP* is defined as a triplet ($G_T(T, C_<)$, R, C_R) where $G_T(T, C_<)$ is the tasks graph with:

- *T: finite set of tasks* $T = \{T_i\}_{i..n}$ with :
- s_i*: starting time of the task execution* T_i
- p_i*: processing time of task* T_i
- c_i*: completion time of task* T_i
- r_i*: release time of the task* T_i
- d_i*: due time*

$C_<$*: precedence constraint (partial order relation on* $T \times T$*).*

T and $C_<$ *make* G_T *task-Graph.*

Feasible scheduling: A scheduling is feasible if it satisfies the precedence constraints and resource availability constraint.

Objective function: The objective function is a metric that describes the quality of the solution based on specific objectives.

$$\phi : \Pi \rightarrow \mathbb{R} \tag{1}$$

$$\theta \mapsto q = \phi(\theta) \tag{2}$$

Optimal scheduling: The optimal schedule is a feasible solution that optimizes the objective function on the state space.

$$\theta_* = \mathrm{Min}_{\theta\in\Pi}\phi(\theta \tag{3}$$

R: finite set of resources $R = \{R_i\}_{i.m}$ with the maximum available amount m_i

C_R: resources constraint. It defines for each task, resource requirements with a_{ij} the necessary amount of resources R_j for the execution of T_i.

Scheduling: The scheduling of an instance of an *SP* is an application that associates to each task T_i and its resource requirements, start time of its execution during p_i.

$$\theta : T \times R \rightarrow \mathbb{R} \tag{4}$$

$$(T_i \times R_j) \mapsto t_{ij} \tag{5}$$

State space (Π): The state space is the set of all possible scheduling on $T \times R$.

In the sequel we adopt the Graham's notation [4] $<\alpha|\beta|\gamma>$ for *SP* where:

- α represents the machine environment,
- ß represents the characteristics of tasks,
- γ is the objective function.

B. Illustrative Example.

The example illustrated by the Fig. 1 will be used in the sequel as a reference for our mapping method.

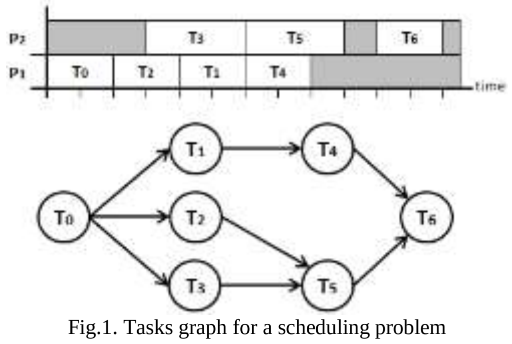


Fig.1. Tasks graph for a scheduling problem

According to Graham notation, Fig. 1 represents an instance of the *SP*, where the environment machine contains two identical parallel machines with precedence constraints and optimizing the total execution time as objective. The different elements of the *SP* are:

$T = \{T_0, T_1, T_2, T_3, T_4, T_5, T_6\}$

$P = \{2, 2, 2, 3, 2, 3, 2\}$

$< \, = \{(T_0, T_1), (T_0, T_2), (T_0, T_3), (T_1, T_4), (T_2, T_5), (T_3, T_5), (T_4, T_6), (T_5, T_6)\}$

$R = \{R_1, R_2\}$ two instances of this kind of resources are available (processors).

C_R: each task uses a unit of this resource for its execution.

Fig. 1 gives a possible solution to this instance, its configuration is:

Scheduling $= \{(T_0, R_1, 0), (T_1, R_1, 4), (T_2, R_1, 2), (T_3, R_2, 3), (T_4, R_1, 6), (T_5, R_2, 6), (T_6, R_2, 10)\}$.

C. Working hypotheses

In this study, a number of assumptions on the *SP* model are to consider:

- Simultaneous use of a resource by tasks is prohibited.
- Once launched, a transaction must complete without interruption.
- Lack of communication between resources and tasks.
- No duplication of tasks.
- Processing times are fixed and known in advance.
- Model used is parallel identical machines *SP* where resources are independent and having the same features. This model is classified by Graham as $<P|\beta|\gamma>$.

III. Time Petri Nets

Time Petri nets (*or TPN for short*) [17] and [18] are a convenient model for real time systems and communication protocols. TPN extend Petri nets by associating two values (*min, max*) of time (*temporal interval*) to each transition. The value *min* $(min \geq 0)$, is the minimal time that must elapse, starting from the time at which transition *t* is enabled until this transition can fire and *max* $(0 \leq max < \infty)$, denotes the maximum time during which transition *t* can be enabled without being fired. Time *min* and *max*, for transition *t*, are relative to the moment at which *t* is enabled. If transition *t* has been enabled at time α, then *t* cannot fire before $\alpha+min$ and must fire before or at time $\alpha+max$, unless it is disabled before its firing of another transition.

We assume that the reader is familiar with the definitions and basic concepts of Petri nets [19].

IV. Modeling SP by Time Petri Nets

This section presents main steps of modeling SP by Petri nets. Entities of SP (task, resource, constraint) are associated to elements of Petri net model (place, transition, edge). Two main known translating methods are referenced in literature. These methods enable to map a SP into a Petri net model. The first method is due to Chrétienne and the second one is due to Van der Aalst. In both methods, translating process is made up of six steps:

- Modeling tasks,
- Modeling precedence constraints,
- Modeling resources,
- Modeling constraints of available resources,
- Initial marking of time Petri net
- Synchronization mechanism.

Unfortunately in these methods, there is no way to control the policy of the resources used by the system. This fact is considered as the main drawback of such methods. For this reason, in this paper, we propose an extension of these traditional approaches by taking into account this fact.

A. Scheduling Problem Modeling with Load Balancing

In previous methods there is no way to know how resources were used; also the interest is oriented only to tasks availability not for their loading. Our approach tries to combine the best of both previous methods by incorporating load balancing of resources. The main idea is to model separately each constraint, then separate the Petri net corresponding to the tasks graph from those corresponding to resources. Both Petri nets are connected in specific points through which communication is done. This approach allows us to adjust the load on resources resulting from the execution of tasks and therefore achieve a steady state system with a feasible scheduling. The method will be applied to the case study formed by identical parallel machines. Our mapping method proceeds in three main steps: mapping tasks graph into time Petri net task, mapping resources into time Petri net resources and finally connect these two generated nets [20] and [21].

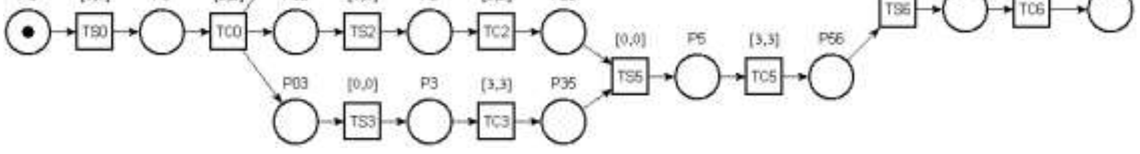


Fig.2. Tasks Time Petri net

Let $SP = (G_T\ (T, C_<), R, C_R)$ be an identical parallel machine scheduling problem.

- **Mapping tasks graph**

  $G_T\ (T, C_<)$ is transformed into tasks time Petri net

***Step1**. Modeling tasks Ti (see Fig. 3 (a)).*
- *T_{Si} start transition to launch T_i*
- *P_i associated place for T_i*
- *T_{Ci} completion transition to end T_i.*
- *Create an arc from T_{Si} to P_i and another from P_i to T_{CI}*

***Step2**. Modeling the precedence constraint $T_i < T_j$ by adding (see Fig. 3 (b)):*
- *P_{ij} place associated with this constraint*
- *Create an arc from P_{ij} to T_{SJ} and another from T_{CI} to P_{ij}*

***Step3**. Defining the initial marking*
- *All places are empty (initially no task is in execution).*

***Step4**. Modeling the synchronization mechanism*
- *The time is associated with completion transitions T_{Ci} and equal to p_i*

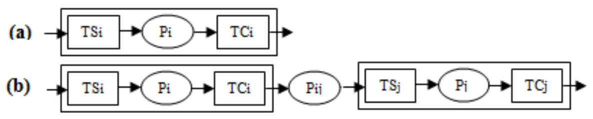


Fig.3. Tasks Time Petri net

In the resulting net, two specific places are added:

P_S: Start place to start the process. P_S is initialized with a token that passes through the rest of the net.

P_C: Completion place to end the process. The process stops when the token reaches the place P_C.

By applying the mapping process to the *SP* illustrated by the Fig. 1, we obtain the tasks time Petri net represented by the Fig. 2. This part of solution illustrates a task graph side as a time Petri net node starting by task start transition. The place having a token represents the task processing and the task end transition represents the task delay. When the task finishes then delivers its outputs to the next step. The overall process starts with token on its first place *PS* and ends when the last transition *PC* expires it's waiting time as execution of the last task in the task graph.

- **Mapping resources**

Resources are translated into Resources Petri Net.

***Step1**. Modeling Input / Output Resources:*
- *R_{IP}, R_{OP}, P_{SW}, Input/ Output / Switch place.*

***Step2**. Modeling resource R_j is achieved by introducing:*
- *P_{RJ} place*
- *Q_{IRj}, Q_{ORj} place points of entry / exit non-reentrancy be associated with Rj*
- *T_{IRj}, T_{ORj} transitions input / output of R_j*
- *(P_{IR}, T_{IRj}) (T_{ORj}, P_{OR}) arcs are connected to the outside*
- *(T_{ORj-1} Q_{ORj}) (T_{IRj-1} Q_{IRj}) arcs connecting the preceding and following resources*

***Step3**. Defining the initial marking*
- *P_{RJ} contains m tokens.*

By applying these steps to *SP* of Fig. 1, we obtain the resource time Petri net illustrated by the Fig. 4.

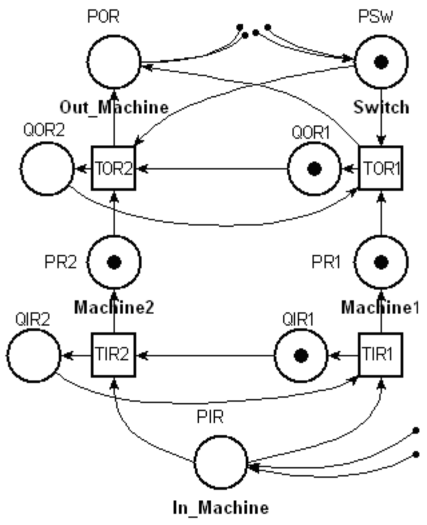


Fig. 4. Resources Time Petri net

- **Connection of Tasks time Petri Net and Resources time Petri Net**

Before the connection we model resource constraints and allocation of resources to tasks.
- *(P_{OR}, T_{Si}): Resource application.*
- *(T_{Si}, P_{SW}): Notification of use of the resource.*
- *(T_{CI} P_{IR}): Release of the resource.*

Connection of tasks time Petri net (Fig. 2) and resources Petri net (Fig. 4) forms a time Petri net illustrated by the Fig. 5.

When the task is ready to run it consumes available resources modeled by the place *P_{OR}*. Then, this task generates a token on the place *P_{SW}* meaning that the next resource is ready to be consumed. The load distribution is ensured by the fact that each resource can be fairly used. When the place P_{SW} will be marked up it goes out to the next resource that will put the token in the place *P_{or}* (see Fig. 5). This process is repeated while exist enough resources to satisfy requests made by tasks. In other words, we expect the release of used resources. When a task completes then it releases the possessed resource. By recovering this resource, the place *P_{IR}* gives the control to the next resource.

This solution respects two principles; the first one using the resources in round robin fashion and the second concerns the availability of the resource for task execution. In this situation, the execution is blocked until a resources is released (become available) then it will be allocated to one of the waiting tasks. After this we will pass turn token to the next resources as marked for allocating next time.

B. Properties Analysis

The analysis of time Petri net illustrated by the Fig. 5 is performed using the *TiNA* tool software version 8.0 (*www.laas.fr/bernard/tina.htm*). The analysis has shown that the network has good properties. Fig. 6 summarizes some analytical results given by *Net Control Draw of TiNA* tool. It is possible to analyze all firing sequences of a time Petri net corresponding to a scheduling problem. Each firing transitions sequence starting from initial marking to final marking corresponds to a feasible scheduling. Transition sequences belonging to time Petri net given by Fig. 5 define the using of resources by the tasks. A transition sequence from the initial place *PS* to final place *PC* models ordered execution of tasks and ordered use of resources.

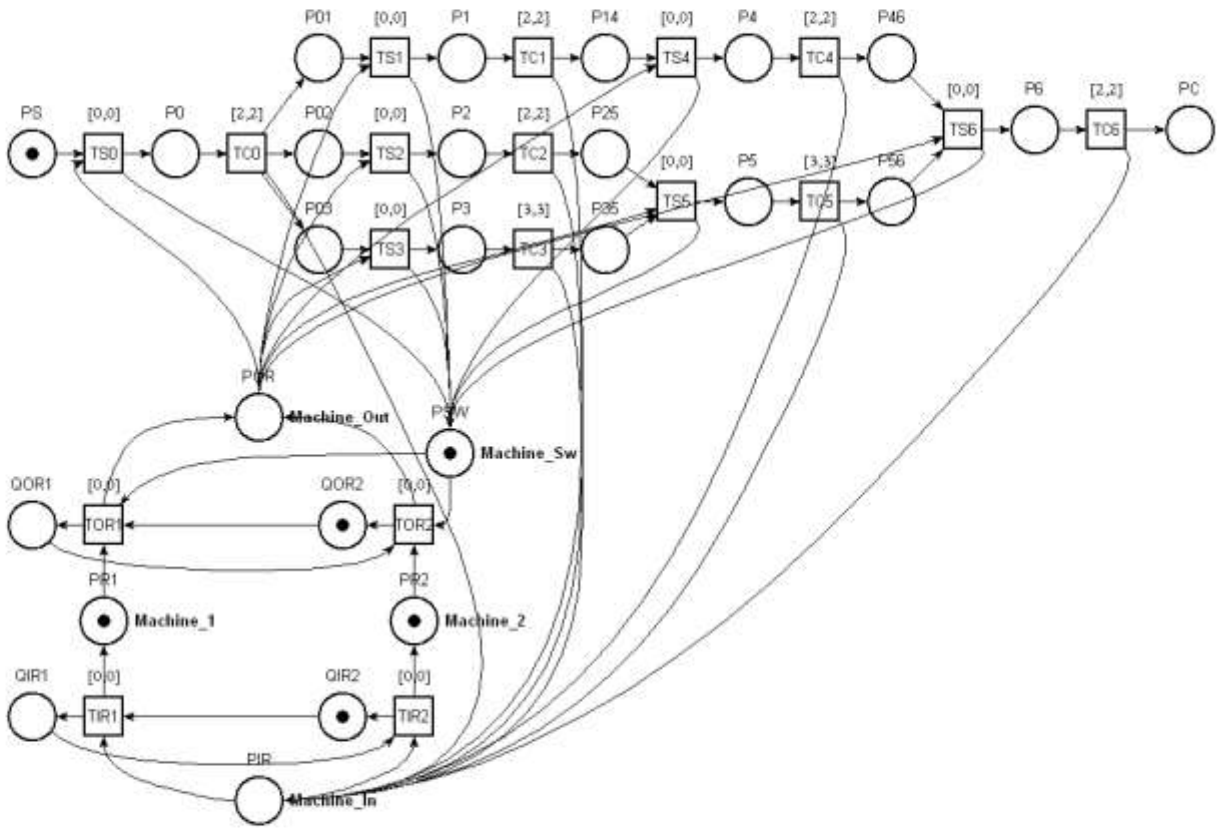
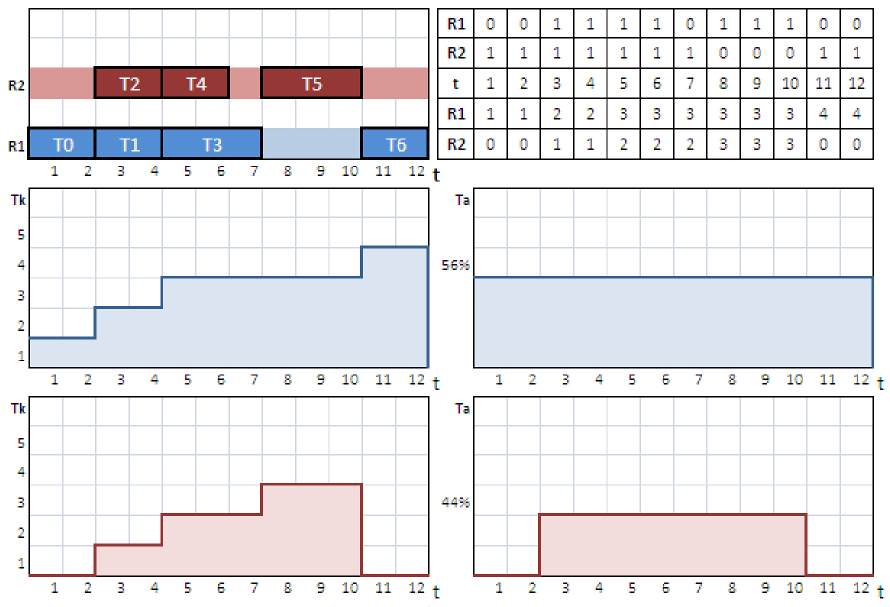


Fig. 5. Time Petri net of SP

In the time Petri net of Fig. 5, a transition sequence generated by using *TiNA* tool is given as scheduling solution such a 5-tuple (Task, t_{begin}, Used-Ressource, Execution-time, t_{end}) :



R1	0	0	1	1	1	1	0	1	1	1	0	0
R2	1	1	1	1	1	1	1	0	0	0	1	1
t	1	2	3	4	5	6	7	8	9	10	11	12
R1	1	1	2	2	3	3	3	3	3	3	4	4
R2	0	0	1	1	2	2	2	3	3	3	0	0

Fig. 6. Gant chart for resources utilization

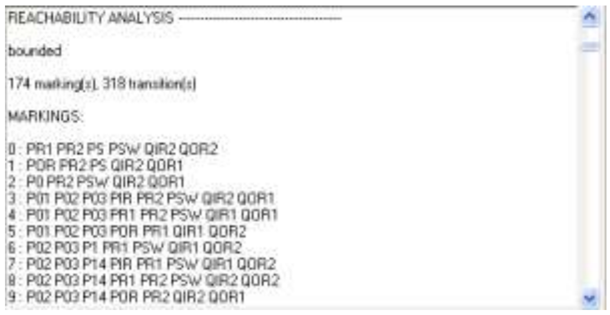
TRACE = TOR1@0.0 TS0@0.0 TOR2@0.0 TC0@2.0 TS2@0.0 TIR1@0.0 TOR1@0.0 TS1@0.0 TC2@2.0 TIR2@0.0 TC1@0.0 TOR2@0.0 TS4@0.0 TIR1@0.0 TOR1@0.0 TS3@0.0 TC4@2.0 TIR2@0.0 TOR2@0.0 TC3@1.0 TS5@0.0 TIR1@0.0 TOR1@0.0 TC5@3.0 TS6@0.0 TIR2@0.0 TOR2@0.0 TC6@2.0 TIR1@0.0

SOLUTION = {(T0, 0, R1, 2, 2), (T1, 2, R1, 2, 4), (T2, 2, R2, 2, 4), (T3, 4, R1, 3, 7), (T4, 4, R2, 2, 6), (T5, 6, R2, 3, 10), (T6, 10, R1, 2, 12)}

Resources utilization of *R1* and *R2* are *75%* and *58%* respectively with a load average of *56%* to *R1* and *44%* to *R2* (number of assigned tasks), maximal difference between loading of *R1* and loading of *R2* is equal to one task. For our example above, the obtained solution has an acceptable balanced load among the whole system. This situation allows us to get good level of performance using the available resources. The classical methods do not permit to know such information because there is no way to save which resource executes which task. Fig. 6 shows the solution of the previous example, the load of each resource and the average load of using them.

Many of the scheduling problem solution can be modeled as reachability problems. In [22], the authors propose a constraint programming based approach to address the reachability problem. The authors use an incremental methodology based on linearization techniques to translate the reachability graph of a timed Petri net into a mathematical programming model.

An enumerative method exists in order to exhaustively validate the behavior of time Petri net. This technique is related to the reachability analysis method for usual Petri nets and allows one to formally verify time dependent systems. The *TiNA* tool uses the enumerative method and is applied to the verification of time Petri net model of *SP* illustrated by the Fig. 5. The analyze reveals that our SP has good properties as summarized in Fig. 7.



```
REACHABILITY ANALYSIS --------------------------------

bounded

174 marking(s), 318 transition(s)

MARKINGS:

0 : PR1 PR2 PS PSW QIR2 QOR2
1 : POR PR2 PS QIR2 QOR1
2 : P0 PR2 PSW QIR2 QOR1
3 : P01 P02 P03 PIR PR2 PSW QIR2 QOR1
4 : P01 P02 P03 PR1 PR2 PSW QIR1 QOR1
5 : P01 P02 P03 POR PR1 QIR1 QOR2
6 : P02 P03 P1 PR1 PSW QIR1 QOR2
7 : P02 P03 P14 PIR PR1 PSW QIR1 QOR2
8 : P02 P03 P14 PR1 PR2 PSW QIR2 QOR2
9 : P02 P03 P14 POR PR2 QIR2 QOR1
```

Fig. 7. Analysis Results given by TiNA

The reachability graph shows all accessible states. In the context of our case study by time Petri nets, the reachability graph represents a spanning tree of all possible paths for task execution. Based on the properties of such graph, we can determine not accessible states, all terminal states (leaves of the tree), etc. If one of these leaves matches a final task in the task graph then we find a critical path that shows the longest sequence of tasks execution in this graph.

After launching the simulation, we will save its trace. By analyzing it, we will depict the most important features of this modeling method. Fig. 7 depicts that time Petri net of Fig. 5 is bounded and the reachability graph (state classes' graph) contains 174 markings (state classes) and 318 transitions. The trace generated by *TiNA* simulator shows that the obtained solution has 12 units length, this means that the optimal solution has a Makespan C_{max} <= 12 units for this instance of problem. So, based on this result, we can fix the parameters of other algorithms to get this optimal solution.

V. Conclusion and Future Perspectives

In this paper we have proposed a new approach to mapping a scheduling problem into a time Petri net. This work extends previous works by taking into consideration the load balancing factor. Our mapping method is applied to specific class of *SP* (*identical parallel machines*). The method is based in a clear separation between main parts of a *SP*: tasks graph and resource constraints. Both parts are translated independently into a time Petri net. The resources time Petri net can be apportioned equally (resource use policy) serving requests from the first time Petri net (*tasks Petri net*). The transformation of a *SP* into a time Petri net allows us to analyze properties as: detecting resources conflicts, search of feasible scheduling, determining bound inf/sup of global run time, load balancing, etc. This validation can be done by software *TiNA* tool. Actually, as future prospects, we plan to extend our method to other classes of SP as no identical heterogeneous parallel machines.

Acknowledgment

This work was supported in part by the Industrial Computing and Networking Laboratory of Oran University.

References

[1] R.W. Conway, W.L. Maxwell and L.W. Miller. Theory of Scheduling. Addison Wesley, Reading, Mass. USA, 1967.

[2] R. L. Graham, E. L. Lawler, J. K. Lenstra and A. H. G. Rinnooy Kan. Optimization and Approximation in Deterministic Sequencing and Scheduling Theory: a survey, Discrete Math. 5, 1979, 287-326.

[3] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys. Sequencing and Scheduling: Algorithms and Complexity, vol. 4, Handbook in OR and Management Science. North-Holland, Amsterdam, 1993.

[4] Y. Slimani, A. Imine and L. Sekhri, Modelling Parallel Programs: A Formal Approach. *International Eurosim Conference. HPCN Challenges in Telecomputing and Telecommunication: Parallel Simulation of Complex Systems and Large Scale Applications*, 10-12 June 1996, Delft, the Netherlands.

[5] L. Sekhri, A.K.A Toguyeni and E. Craye. Diagnosability of Automated Production Systems Using Petri Net Based Models, *IEEE SMC' 2004, International Conference on Systems, Man and Cybernetics*. October 10-13, 2004, Hague, Netherlands.

[6] B. Kechar, L. Sekhri and M. K. Rahmouni, CL-MAC: Energy Efficient and Low Latency Cross-Layer MAC Protocol for Delay Sensitive Wireless Sensor Network Applications. *The Mediterranean Journal of Computers and Networks*, ISSN:1744-2397,vol.6, No.1, 2010.

[7] K. Hsing-Pei, B. Hsieh and Y. Yingchieh. A Petri Net Based Approach for Scheduling Resource Constrained Multiple Project, *JCIIE*, vol. 23, No. 6, pp. 468-477, 2006.

[8] C V. Ramamoorthy and M. J. Gonzalez, A Survey of Techniques for Recognizing Parallel Processable Streams in Computer Programs, *AFIPS Conf. Proceedings*, 1980, pp. 1-15.
[9] J. Carlier and P. Chretienne. Timed Petri Net Schedules, *Advances in Petri Nets. Springer*, vol. 62/84. 1989.
[10] W.M.P. Van Der Aalst. Petri Net Based Scheduling, *Operation Research Spectrum*, vol.18, No 4, 1995.
[11] O. Korbaa, A. Benasser and P. Yim. Two FMS Scheduling Methods Based on Petri Net: A Global and a Local Approach, *IJPR*, vol.41, No.7, pp1349-1371, 2002.
[12] J. K. Hyun, L. Jun-Ho and L.Tae-Eog, A Petri Net-based Modeling and Scheduling with a Branch and Bound Algorithm, *IEEE International Conference on Systems, Man and Cybernetics*, (*SMC'02*), October 6-9, 2002, Hammamet, Tunisia.
[13] D. J. Sampathi, Scheduling FMS Using Petri-nets and Genetic Algorithms, *IISST*, July 2012.
[14] I. S. Pashazadeh and N.E. Abdolrahimi, Modeling Entreprise Architecture Using Timed Colored Petri Nets, Single Processor Scheduling, vol. l5, No.1, 2014.
[15] A. E. Keshk, A. B. El-Sissi and M. A. Tewfeek, Cloud Task Scheduling for Load Balancing based on Intelligent Strategy, *International Journal of Intelligent Systems and Applications (IJISA)*, 2014, 05, 25-36.
[16] R. E. De Grande, M. Almulla and A. Boukerche, Autonomous Configuration Scheme in a Distributed Load Balancing System for HLA-based Simulations, *17th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications*, Delft, The Netherlands, 30 October-1 November, pp. 169-176, 2013.
[17] B. Berthomieu. and M. Menasche. An Enumerative Approach for Analyzing Time Petri Nets. *IFIP Congress Series*, vol. 9, 1983, pp. 41-46, North Holland.
[18] P. M. Merlin, A Study of the Recoverability Systems of Computing. Irvine Univ. California, PhD thesis, 1974.
[19] T. Murata, Petri Nets: Properties, Analysis and Applications, *Proc. of IEEE*, vol.77, No. 4, pp. 541-580, 1989.
[20] M. Slimane and L. Sekhri, Modeling the Scheduling Problem of Identical Parallel Machines with Load Balancing by Time Petri Nets. *International Symposium on Modeling and Implementation of Complex Systems*, **(***MISC 2010*), Constantine, Algeria, May 30-31, 2010.
[21] F. Abdiche, K. Mesghouni and L. Sekhri, Meta-Heuristic Optimization Based on P-Time Petri Nets Model for Deadlock Avoidance in Flexible Manufacturing Systems. 21th International Conference on Production Research. *(ICPR 21*), July 31-August 4, 2011, Stuttgart, Germany.
[22] Y. Huang. T. Bourdeaud, P. A. Yvars and A. Toguyeni, A Constraint Programming Model for Solving Reachability Problem in Timed Petri nets. *ICMOS*, 2012 Bordeaux,France.

Authors' Profiles

Sekhri Larbi is an Associate Professor at the Computer Science Department of Oran University. His current research area of interests include formal modeling in distributed and mobile systems, wireless ad-hoc and sensor networks, systems modeling using Petri nets, diagnosability and monitoring of automated production systems. He is member of the Industrial Computing and Networking Laboratory at Oran University. He has been a visiting professor at Cedric-CNAM research laboratory, in Paris, France, and Ecole Centrale de Lille (LAGIS) where he worked in Diagnosis of Industrial Systems; and LIUPA Laboratory at the University of Pau, France and distinguished lecturer at University of Ottawa, Canada.

Slimane Mohamed is a Ph. D. candidate in Department of Computer Science; University of Oran, Algeria. His current research area includes Flexible Manufacturing Systems Modeling, Scheduling Problems, distributed and parallel programming, simulation and information security. He received his Ms.Bc degree in scheduling and mapping distributed applications from Le Havre University, France. He is an assitant professor in Computer Science Departement at University of Chlef, Algeia.

How to cite this paper: Sekhri Larbi, Slimane Mohamed,"Modeling the Scheduling Problem of Identical Parallel Machines with Load Balancing by Time Petri Nets", International Journal of Intelligent Systems and Applications (IJISA), vol.7, no.1, pp.42-28, 2015. DOI: 10.5815/ijisa.2015.01.04