LATVIAN JOURNAL OF PHYSICS
AND TECHNICAL SCIENCES
2025, N 6

DOI: 10.2478/lpts-2025-0043

MODELLING AND ANALYSIS OF SYSTEMS USING PETRI NETS: MANUFACTURING AND TASK MANAGEMENT CASES

S. Affi[1], I. Miraoui[2,3,*], A. Khedher[1]

[1] LATIS Laboratory,
National Engineering School of Sousse,
University of Sousse,
Route de Ceinture Sahloul, Sousse, 4054, TUNISIA
[2] ISSAT Gafsa, University of Gafsa,
Gafsa, 2112, TUNISIA
[3] TEMI Laboratory,
Faculty of Sciences,
University of Gafsa,
Gafsa, 2112, TUNISIA
*e-mail: aimed_mir@yahoo.fr

Reduction methods are widely used to simplify complex models, particularly for Petri nets, which model discrete event systems. Although effective in limiting combinatorial explosion, these methods have potentially critical flaws, especially by reducing the state space in a way that may obscure behaviours or states essential to comprehensive analysis. This paper proposes an approach to address these shortcomings by integrating reachability methods into the reduction process. By leveraging the ability of reachability methods to ensure the attainability of critical states while maintaining efficient state space reduction, this solution enhances the accuracy of complex system analysis while optimising computational resources. Two practical case studies of manufacturing system and task management system illustrate this approach and demonstrate its potential to improve the rigor of large-scale model analyses.

Keywords: *Computational optimisation, discrete event systems, Petri nets, reachability, reduction methods.*

1. INTRODUCTION

Discrete Event Systems (DESs) [1] are widely used in various fields to model processes where distinct and isolated events occur at specific moments, such as railway traffic control systems [2], computer networks [3], industrial production systems [4], and task scheduling in operating systems [5]. Before their final deployment, these systems follow a lifecycle in which each stage is crucial for validation. One of the most important stages is the abstraction of DES through modelling, a process that is particularly necessary due to the increasing complexity of these systems [6]. Discrete Event Modelling (DEM) plays a key role in control [7], diagnosis [8], and state estimation of DES [9]. Several modelling dimensions can be adopted depending on the study domain: structural, logical, and temporal modelling, which describes the organization of system components, functional behaviours, temporal modelling, and time constraints, respectively. These dimensions may overlap within the same DES, making it important to decide on the adequate modelling formalism due to the complexity and close interactions between these different aspects.

It is therefore essential to select a formalism that can capture these various aspects, especially for critical systems where temporal precision is paramount. Among the most used formalisms for modelling and analysing DES are automata [10] and Petri nets (PNs) [11], which offer high expressiveness through various extensions, such as Timed Petri Nets (TPNs) [12], Untimed Petri Nets (U-PNs) [13], and Time Interval Petri Nets (TIPNs) [14].

Modelling not only represents the structure and behaviour of systems but, when combined with analysis [15] and simulation methods [16], also enables the verification of essential properties, such as performance estimation, responsiveness, and fault detection. To this end, different methods exist for reachability analysis of critical states depending on modelling formalism chosen. However, the complexity of DESs lies in their evolving nature, where frequent state changes and the creation of new events lead to a combinatorial explosion of the state space, making analysis more challenging. Model reduction methods [17] are therefore essential to simplify modelling by reducing the state space, which is particularly relevant for large-scale complex systems. However, while these reduction methods are effective in managing the growing complexity of systems, they can sometimes introduce ambiguity or inaccuracy in reachability analysis [18]. This issue is particularly problematic in large systems, where reduction may obscure critical details necessary for precise analysis. It is essential, therefore, to strike a balance between simplification and accuracy to ensure that the critical properties of systems remain verifiable.

2. PNS FOR FORMAL MODELLING AND ANALYSIS OF DES

Formal modelling is a solution that helps address multiple problems at an early stage in the lifecycle of DES. It is based on a solid and powerful mathematical formalism that is useful from the specification phase through to the verification of relevant prop-

erties for DES analysis. Considerable attention is therefore given to formal modelling methods [19], specifically model checking [20], which is highly relevant for system analysis [21]. The principle involves applying tests to scenarios that are automatically generated from formal specifications.

In the literature, some of the most prominent and widely used formalisms for DES analysis based on model checking include automata [22] and PNs [23]. In the timed automata approach, all system executions and properties are represented using automata [22]. However, when dealing with complex systems, automaton-based approaches face an exponential growth of the state space. Moreover, they are unable to model a crucial aspect in this field: parallelism. Due to this lack of expressiveness, PNs have been adopted as a more powerful formalism for DES analysis. PNs [24] are widely used to model DES. They combine mathematical and graphical tools. The graphical representation of PNs is based on state and event modelling, represented respectively by places and transitions.

2.1. Definition of PN

$PN = (P,T,M_0,F)$; PN is a finite group of n places denoted P, where n > 0; a finite group of m transitions denoted T, where m > 0; the firing of transitions F; and each state is described by an initial marking M0.

M_0: P → N which defines the initial dispatching of tokens in the places.

The PN dynamics can be illustrated through the firing of transitions F(Firing). When a transition is enabled, the activation conditions are met. Transition firing helps simulating the system evolution, analysing system properties, designing control systems based on system objectives and constraints.

- Incidence Matrix (IM)

It indicates which arcs connect each place to each transition and conversely. The elements of the IM describe the weights of the arcs.

- State Equation (SE)

The SE is an equation that describes the state of the PN at any time. It can be written in matrix form $A * M = b$, where A is the IM, M is the marking, and b is a vector of boolean values indicating which transitions can be fired. The vector b is determined by comparing the current marking with the preconditions of each transition. If the conditions of firing of transition are verified, the corresponding entry in vector b is set to 1; otherwise, it is 0. The SE helps studying the system evolution based on the events that occur. This is done by updating the marking vector p each time a transition fires.

The simulation process involves the identification of the highest-priority transition that can fire, using strategies such as static priority or dynamic priority and then updating the marking vector accordingly.

After introducing the important concepts of PNs and defining the IM, it is crucial to focus on reachability analysis methods for DES.

Once a PN model is defined, it becomes essential to study the system's ability to reach certain states, especially critical states that ensure the system's proper operation or safety.

This type of analysis is based on the notion of reachability, which refers to the possibility of reaching a specific marking (a particular system state) from M0 by following a sequence of allowed transitions.

Reachability methods [25] play a key role in evaluating important system prop-

erties, such as liveness (ensuring that the system does not deadlock), safety (preventing the system from entering unwanted states), and deadlock-freeness (ensuring that no transition remains permanently disabled). Several formal methods [19] exist to analyse reachability in PNs, including graph theoretical method [26], linear algebraic techniques [27], and temporal logic framework [28]. Ensuring that the system behaves as expected according to the given specifications and application requirements, these methods allow for formal verification.

2.2. Methods of Reachability or Coverage of PNs

The problem of reachability in PNs is fundamental. It consists of determining whether it is possible to reach a marking M from M0.

Let us consider a PN (N, M0). We denote R(M0) as the set of reachable markings from M0.

Among the reachability methods, also known as analysis methods, the most common one is the tree traversal method [29], where search tree is constructed in which each node represents a marking, and each branch represents the crossing of a transition. This method is simple but can be computationally expensive for large PNs.

The graph coverage method [26], on the other hand, involves constructing a reachable marking graph starting from the initial marking. Each node represents a reachable marking, and each arc represents a transition traversal (reachable marking graph). For this, standard graph traversal algorithms, such as Dijkstra's algorithm [30] and Bellman-Ford's algorithm [31], are used. This method is more efficient than the tree traversal method for large-size PNs.

The linear algebraic method [32] involves representing the system in the form of matrices and vectors to solve the SE. By multiplying M0 by the IM, the resulted vector represents the next vector. By repeating this operation as needed, all reachable markings from the initial marking can be determined. This method can also be used to determine certain properties of the system through a direct analysis of the linear equations derived from the IM, such as liveness, safety, and capacity limitation.

Liveness is ensured by the existence of a marking from which all transitions can be fired. Safety is guaranteed if there is no deadlock. Capacity limitation is determined by the presence of a marking that exceeds a certain resource limit. The linear algebraic method offers a powerful analytical approach, especially for complex systems, by focusing on the linear relationships between states.

In contrast to the graph coverage or tree traversal methods, which explore the entire state space by constructing graphs or trees of all possible transitions, this method relies on a more compact matrix representation. It allows for the calculation of reachable states by solving the SE, without explicitly constructing the entire set of possible states. This IM method is particularly effective for analysing complex and large-scale systems. Thus, the matrix approach proves to be a more elegant and efficient method for analysing reachability in systems modelled by PNs, especially when the system's complexity makes other methods impractical.

The diagram in Figure 1 illustrates how a marking tree is built from an initial marking M0. It follows a recursive process to explore all markings reachable from M0 by activating transitions. This algorithm is essential for analysing Petri nets, particularly to (1) check if a net is bounded (avoiding infinite growth of the marking); (2) determine if a reachability condition is satisfied; (3) identify deadlock situations.

3.3. Methods for Reducing PNs

PN reduction [33] is an essential technique for simplifying a complex system by eliminating or merging redundant places, transitions, or arcs while following reduction rules [34]. This approach enables a reduction in the state space dimension.

- **Place reduction**

It involves removing redundant places or merging those with identical preconditions and postconditions. Transition reduction eliminates redundant transitions or merges those with similar effects (i.e., triggered in the same way and having equivalent impacts on the system). This significantly reduces the size of the state space, making analysis simpler and computationally less expensive.

- **Hierarchical reduction of subnetworks**

It involves replacing subsets of the network with a single place or transition [35], thereby greatly simplifying the overall model structure. In [36], hierarchical reduction of PN models is performed in two steps: first, encapsulating internal behaviour through external states for information exchange; second, constructing an improved reachability graph, where arcs between graph states are triggered by relevant transitions.

- **Causality reduction**

It removes redundant causal arcs that do not impact the system's behavioural properties, while conflict reduction eliminates unnecessary conflicts that do not alter the system's overall behaviour.

These various classical reduction techniques, often used in combination, help simplify the model. However, while they are effective in reducing the state space by merging or eliminating certain components, they may sometimes introduce approximations that degrade analysis accuracy. Reduction methods generally aim to minimise the model size at the cost of trading off certain essential information, which can lead to the loss of critical details about system behaviour especially in nonlinear systems or those with complex dependencies. Therefore, their effectiveness must be carefully weighed against the risk of inaccuracies in property verification.

3. COUPLING REDUCTION METHODS AND FORMAL METHODS TO ENSURE REACHABILITY

The behavioural models of DES capture the dynamic interactions between different parts of the system. By modelling possible behaviours using PN, we can better understand how the system reacts in various configurations or states. However, the state space expands significantly. PN reduction is a crucial technique to simplify the structure and reduce system complexity. However, this simplification can sometimes lead to the loss of vital information, particularly in systems with complex interactions, where certain subtle behaviours may disappear. This can impact essential behavioural properties such as liveness, conservation, or safety invariants. The primary challenge of reduction methods is that they may simplify the model to the extent that some key

behaviours or interactions become hidden.

Powerful reduction methods must therefore be used with extreme caution to preserve the integrity of the original model and ensure the analysis of relevant PN properties. It is crucial to verify, after each reduction step, that these properties are still maintained, especially in complex systems where even slight loss of information can obscure critical behaviours.

The linear algebraic technique stands out for its ability to maintain highly accurate representation of the system while avoiding the combinatorial explosion of the state space, without sacrificing precision [36]. This technique can be integrated with reachability methods, mitigating the limitations of reduction methods while preserving a rigorous analysis of essential properties such as liveness and conservation [37]. These aspects position the algebraic approach as a more reliable and comprehensive solution for systems where traditional reduction methods fail to capture the full complexity.

The proposed technique in this paper uses the linear algebraic technique. In fact, using reduction methods coupled with formal methods can help address the limitations posed by classical reduction techniques in the analysis of complex systems. Our approach enables the efficient management of complex systems even when a system is very large and intricate by leveraging formal methods [38] (such as Analytical Feasibility Tests [39], Deductive Methods [40], and Model Checking [15]). These provide a rigorous and systematic means to verify that the model behaves as expected at each stage. This allows for automatic verification that all essential properties are preserved after reduction.

Two levels of reduction are used:

- at the initial PN model level, to simplify the system while preserving key properties;
- after the construction of the reachability graph, to use successor constraints applied to the reduced PN [41].

A verification of the accuracy and consistency of the reductions is performed at each step to ensure that essential states for system behaviour are not lost during the reduction, thus maintaining precision. Systematic verifications are conducted to guarantee that certain properties (such as safety, liveness, or the reachability of critical states) are preserved and respected.

By coupling formal methods with reduction methods, we ensure that state-space exploration remains complete, preventing ambiguities and uncertainties introduced by simplification. This approach ensures model precision and reliability while meeting system requirements, from specification to development phases. It enables the construction of correct-by-construction systems [42].

4. CASE STUDY 1: MANUFACTURING SYSTEM

The system considered contains two machines, denoted **M1** and **M2**, and three tasks. Each task undergoes an operation stage, which is fulfilled on machine M1 or machine M2. A new task is only initiated after the previous task has been completed and discharged from the system. The elements of this model are provided in Table 1, which explains the data. The proposed system's PN model is shown in Fig. 2.

Table 1. Data Explanation

Places	Transitions
P1: Job 1	t1: M1 starts Proc. Job 1
P2: M1 available	t2: M1 starts Proc. Job 2
P3: Job 2	t3: M1 starts Proc. Job 3
P4: M2 available	t4: M2 starts Proc. Job 1
P5: Job 3	t5: M2 starts Proc. Job 2
P6: M1 Proc. Job 1	t6: M2 starts Proc. Job 3
P7: M1 Proc. Job 2	t7: M1 finishes Proc. Job 1
P8: M1 Proc. Job 3	t8: M1 finishes Proc. Job 2
P9: M2 Proc. Job 1	t9: M1 finishes Proc. Job 3
P10: M2 Proc. Job 2	t10: M2 finishes Proc. Job 1
P11: M2 Proc. Job 3	t11: M2 finishes Proc. Job 2
	t12: M2 finishes Proc. Job 3

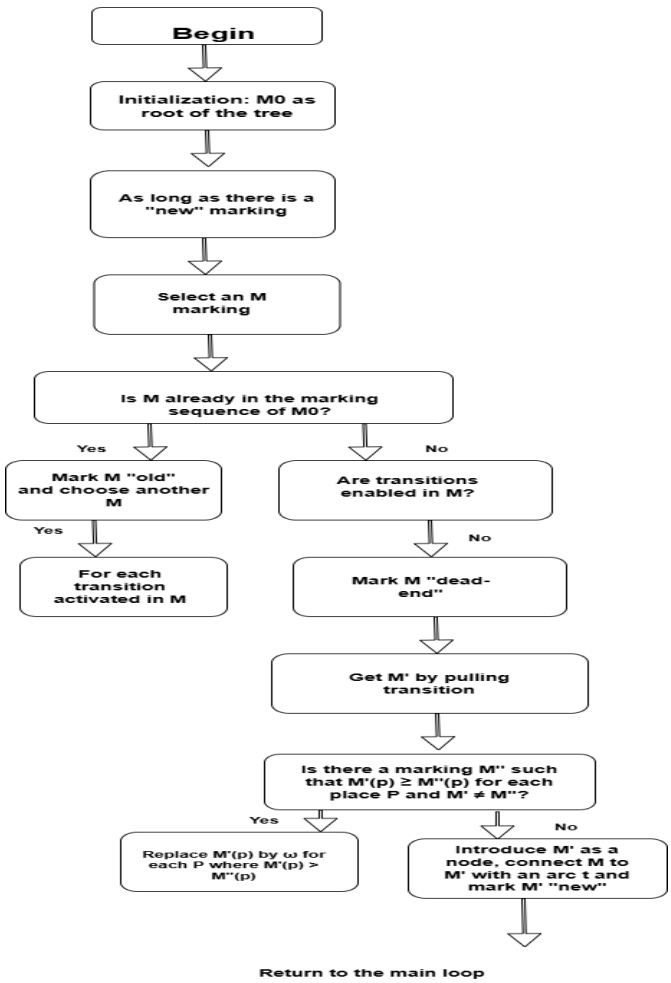


Fig. 1. Algorithm for the design of a covering tree for a PN.

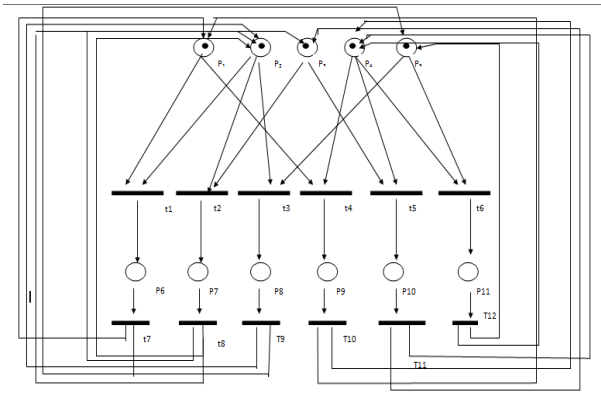


Fig. 2. PN with initial marking .

4.1. Formal Definition

The formal definition of this system is as follows:

$$\mathrm{PN} = (\mathrm{P}, \mathrm{T}, \mathrm{F}, \mathrm{M0}), \text{where}$$

$$F = \left\{ \begin{array}{c} (\mathrm{P1}, \mathrm{t1}), (\mathrm{P1}, \mathrm{t4}), (\mathrm{P2}, \mathrm{t1}), (\mathrm{P2}, \mathrm{t2}), (\mathrm{P2}, \mathrm{t3}), (\mathrm{P3}, \mathrm{t2}), (\mathrm{P3}, \mathrm{t5}), (\mathrm{P4}, \mathrm{t4}), (\mathrm{P4}, \mathrm{t5}), (\mathrm{P4}, \mathrm{t6}), \\ (P5, t3), (P5, t6), (P6, t7), (P7, t8), (P8, t9), (P1, t1), (P9, t10), (P10, t11), (P11, t12), (t1, P6) \\ (t2, P7), (t3, P8), (t4, P9), (t5, P10), (t6, P11), (t7, P1), (t7, P2), (t8, P2), (t8, P3), (t9, P2), \\ (t9, P5), (t10, P1), (t10, P4), (t11, P3), (t11, P4), (t12, P4), (t12, P5) \end{array} \right\}$$

$$\mathrm{P} = \{\mathrm{P1}, \mathrm{P2}, \mathrm{P3}, \mathrm{P4}, \mathrm{P5}, \mathrm{P6}, \mathrm{P7}, \mathrm{P8}, \mathrm{P9}, \mathrm{P10}, \mathrm{P11}$$
$$\mathrm{T} = \{\mathrm{t1}, \mathrm{t2}, \mathrm{t3}, \mathrm{t4}, \mathrm{t5}, \mathrm{t6}, \mathrm{t7}, \mathrm{t8}, \mathrm{t9}, \mathrm{t10}, \mathrm{t11}, \mathrm{t12}\}$$
$$\text{and } \mathrm{M0} = (1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0).$$

It corresponds to the availability of machines and the tasks waiting for processing. The input places of all the transitions in the PN illustrated in the figure are as follows:

$$\mathrm{IP(t1)} = \{\mathrm{P1}, \mathrm{P2}\}, \mathrm{IP(t2)} = \{\mathrm{P2}, \mathrm{P3}\},$$
$$\mathrm{IP(t3)} = \{\mathrm{P2}, \mathrm{P5}\}, \mathrm{IP(t4)} = \{\mathrm{P1}, \mathrm{P4}\},$$
$$\mathrm{IP(t5)} = \{\mathrm{P3}, \mathrm{P4}\}, \mathrm{IP(t6)} = \{\mathrm{P4}, \mathrm{P5}\},$$
$$\mathrm{IP(t7)} = \{\mathrm{P6}\}, \mathrm{IP(t8)} = \{\mathrm{P7}\},$$
$$\mathrm{IP(t9)} = \{\mathrm{P8}\}, \mathrm{IP(t10)} = \{\mathrm{P9}\},$$
$$\mathrm{IP(t11)} = \{\mathrm{P10}\}, \mathrm{IP(t12)} = \{\mathrm{P11}\}.$$

In a PN, concurrent and non-deterministic access activities are identified through simultaneous and conflicting transitions. This type of transition, derived from Fig. 2 of the proposed PN model, is presented in Table 2.

Table 2. Transitions

Conflicting transitions	Simultaneous transitions
(t_1, t_4),(t_1, t_2, t_3), (t_2, t_5), (t_4, t_5, t_6), (t_3, t_6).	(t_1, t_5),(t_1, t_6),(t_2, t_4). (t_2, t_6), (t_3, t_4), (t_3, t_5), (t_7, t_{11}), (t_7, t_{12}), (t_1, t_{11}).

4.2. Crossing

Several characteristics of a system (decision-making, simultaneity, synchronization, priorities, etc.) can be modelled using PNs. The relationship between the current marking M and M0 is erased based on the reachability properties.

4.3. Property Verification (Boundedness)

A PN is said bounded, if the number of token is always less than a finite number *m* for every reachable marking. The PN shown in Fig. 2 is bounded because *m* = 1. Its covering tree is shown in Fig. 3. The reachable markings of the PN is illustrated in Table 3.

- Liveness: If the PN has no deadlock, it is said to be live. The PN illustrated in Fig. 2 does not have a deadlock state, so it is live.

- Reversibility: The PN is reversible if M0 can be reached from all possible markings. The PN illustrated in Fig. 2 satisfies this property.

- Persistence: A PN is called persistent if each pair of activated transitions with independent triggering remains activated until it is triggered. The PN illustrated in Fig. 2 is not persistent.

4.4. IM and SE

The evolution of systems modelled by PNs is analysed by a SE. This method is applied only to pure PNs.

• **Definition 1:** The IM of a PN with *n* transitions and *m* places is an *n* × *m* matrix defined by:

$A = [aij]_{n\times m} = O - I$, where: $O = [a_{ij}^{+}]$ represents the number of arcs from transition *i* to its output place *j*; $I = [a_{ij}^{-}]$ represents the number of arcs from transition *i* to its input place *j*.

• **Definition 2:** The SE is $Mn = M_{n-1} + C_n A^T$, where *A* is the *IM* and *Mn* is an *m* × 1 vector, where the *i*-th entry represents the token number in place *i* immediately after the *n*-th firing in a firing sequence.

C_k is an *n* × 1 vector (control vector) with *n* − 10 and 1 non-zero entry, 1 in the *j*-th position, indicating that transition *j* is triggered at the *n*-th firing. The vector *Ck* cannot be chosen arbitrarily because only activated transitions can be triggered. For each *n*, C_n is constrained by $M_{n-1} + C_n A^T \geq 0$. The SE is:

$$\nabla M = ATX \text{ où } \nabla M = Md - M0;\ X = \sum ki = 1\ Ci.$$

X is an *n* × 1 vector of positive integers; it is called the firing count vector. The number of times transition *i* must fire to transform M0 into Md is the *i*-th entry of X. The IM A plays a crucial role in the analysis of the dynamic behaviour and the structure of a PN.

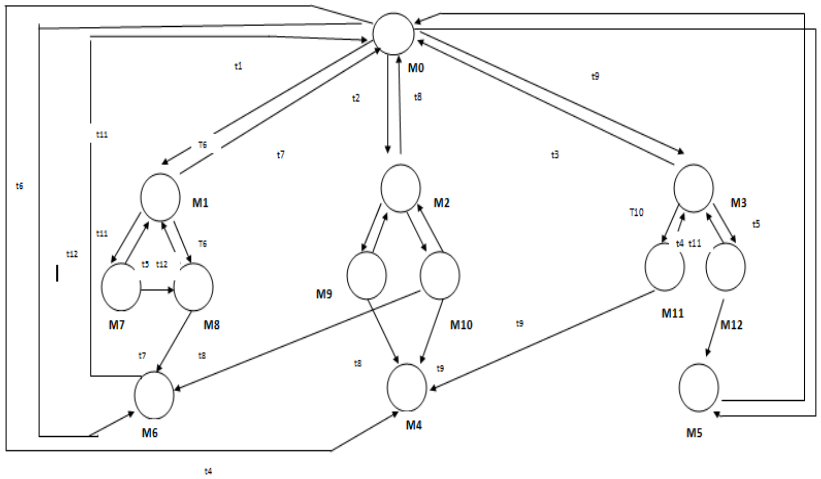


Fig. 3. Reachability graph of Fig. 2.

4.5. Linear Algebraic Technique

For the PN represented in Fig. 2, the SE is as follows:

$$M_k = M_{k-1} + C_k A^T.$$

Table 3. Reachable Markings of Figure 2

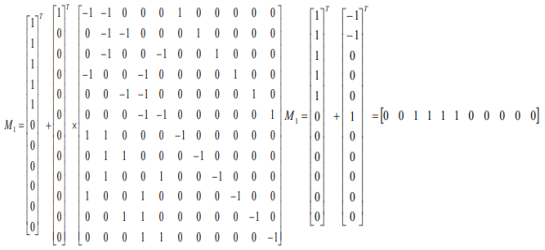
Marquage	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}
M_0	1	1	1	1	1	0	0	0	0	0	0
M_1	0	0	1	1	1	1	0	0	0	0	0
M_2	1	0	0	1	1	0	1	0	0	0	0
M_3	1	0	1	1	0	0	0	1	0	0	0
M_4	0	1	1	0	1	0	0	0	1	0	0
M_5	1	1	0	0	1	0	0	0	0	1	0
M_6	1	1	1	0	0	0	0	0	0	0	1
M_7	0	0	0	0	1	1	0	0	0	1	0
M_8	0	0	1	0	0	1	0	0	0	0	1
M_9	0	0	0	0	1	0	1	0	1	0	0
M_{10}	1	0	0	0	0	0	1	0	0	0	1
M_{11}	0	0	1	0	0	0	0	1	1	0	0
M_{12}	1	0	0	0	0	0	0	1	0	1	0

The IM A is:

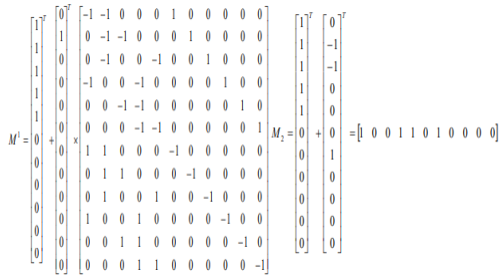
$$A = O - I = \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

$$I = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} O = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

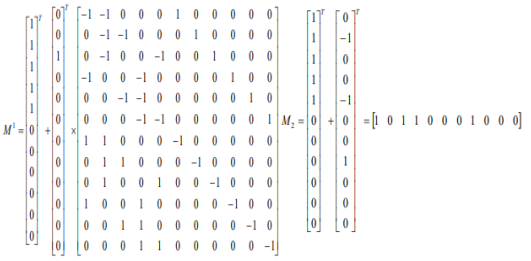
T_1 is fired:



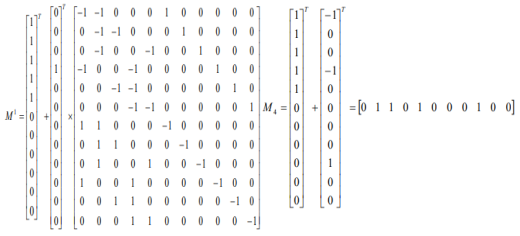
T_2 is fired:



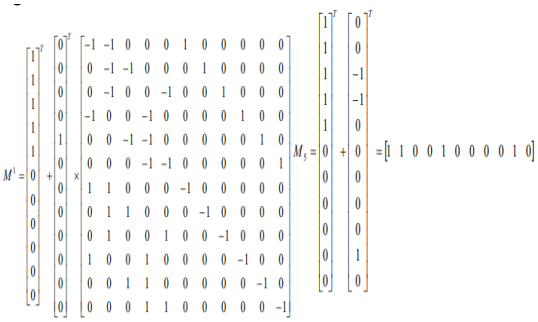
T_3 is fired:



T_4 is fired:



T_5 is fired:



In the same manner, we can find all the other operations and the following brands of the PN:

- **Transition invariant:** Y^TA (invariant) = 0,

where Y is a vector indicating how many times each transition in the network has been performed (the number of firing transitions drawn into Petri net).

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \\ y_{10} \\ y_{11} \\ y_{12} \end{bmatrix}^T \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} = 0$$

$$\begin{gathered} -y1 - y4 + y7 + y10 = 0; \\ -y1 - y2 - y3 + y7 + y8 + y9 = 0; \\ -y2 - y5 + y8 + y11 = 0; \\ -y4 - y5 - y6 + y10 + y11 + y12 = 0; \\ -y3 - y6 + y9 + y12 = 0; \\ y1 - y7 = 0; \\ y2 - y8 = 0; \\ y3 - y9 = 0; \\ y4 - y10 = 0; \\ y5 - y11 = 0; \\ y6 - y12 = 0 \end{gathered}$$

By solving the equations above, we obtain:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \\ y_{10} \\ y_{11} \\ y_{12} \end{bmatrix}^T = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}^T$$

Thus, the invariant $T-$ (from Fig. 2) is:

(0 0 0 0 0 0 0 0 0 0 0 0).

In the same manner, we can find the place invariant $P-$ (from Fig. 2) from the equation:

$AX^T = 0$, where X is the weight of each place and an n-vector as:

(0 0 0 0 0 0 0 0 0 0 0 0 0 0).

4.6. Reduction Technique

This method allows simplifying the study of large systems by reducing their correspondent PNs, while conserving their properties (such as safety and boundedness).

Table 4 presents various transformations applied to Petri nets, illustrated with representations. In Petri nets, reduction rules help simplify complex models while preserving essential properties. Key reduction techniques include:

Fusion of Series Places: If two places, *p1* and *p2*, are connected in sequence through a single transition ($p1 \rightarrow t \rightarrow p2$), and *p1* is the only input to *t* while *p2* is the only output, these places can be merged into a single place. This fusion reduces the net's complexity without altering its behaviour.

Fusion of Series Transitions: Similarly, if two transitions, *t1* and *t2*, are connected

in sequence through a single place ($t1 \rightarrow p \rightarrow t2$), and $t1$ is the only output of p while $t2$ is the only input, these transitions can be combined into a single transition. This simplification maintains the net's functionality.

Fusion of Parallel Places: When two places, $p1$ and $p2$, share identical input and output transitions, they can be fused into a single place. This reduction streamlines the net by eliminating redundant places.

Fusion of Parallel Transitions: If two transitions, $t1$ and $t2$, have the same input and output places, they can be merged into a single transition. This consolidation reduces redundancy and simplifies the net's structure.

These reduction techniques are valuable for managing the complexity of Petri nets, ensuring that analyses remain tractable while preserving the net's original properties.

A proposed reduction of the PN illustrated in Fig. 2 is shown in Fig. 4.

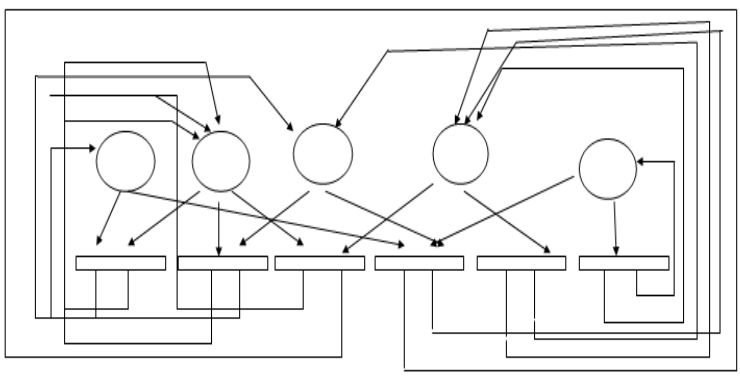
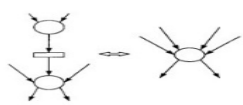
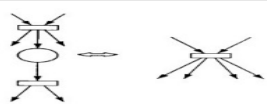
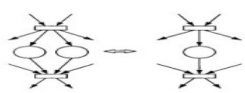
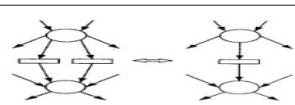


Fig. 4. Reduction of the PN from Fig. 2.

Table 4. Transformations

Transformation	Pictorial Representation
Fusion of Series Places (FSP)	
Fusion of Series Transitions (FST)	
Fusion of Parallel Places (FPP)	
Fusion of Parallel Transitions (FPT)	

5. CASE STUDY 2: TASK MANAGEMENT SYSTEM

Let us consider a task management system (Fig. 5) with:

- Two types of tasks: Urgent Tasks and Normal Tasks.
- A server that processes the tasks.
- Two states for the server: busy or available.

The PN will represent the stages of task processing, with transitions to model state changes (task arrival, processing, etc.).

5.1. Initial Model (Complete PN)

The formal model is:

A. Places:

- P_1: Queue of normal tasks
- P_2: Queue of urgent tasks.
- P_3: Available server
- P_4: Occupied server.
- P_5: Task being processed.

B. Transitions:

- T_1: Arrival of a normal task.
- T_2: Arrival of an urgent task.
- T_3: Start of processing a normal task.
- T_4: Start of processing an urgent task.
- T_5: End of processing.

C. Initial Marking (M_0):

$M_0(P_1)=0$, $M_0(P_2)=0$, $M_0(P_3)=1$,
$M_0(P_4)=0$, $M_0(P_5)=0$.

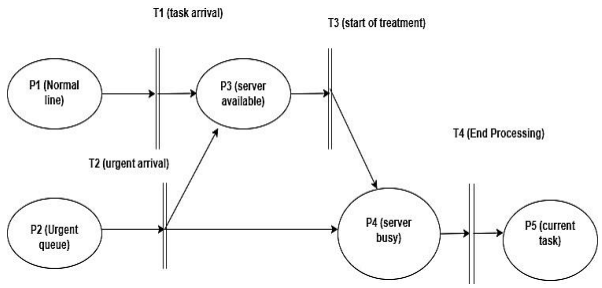


Fig. 5. Initial PN Model – Task System.

The IM is defined as follows:

P_i \ T_i	P_1	P_2	P_3	P_4	P_5
T_1	+1	0	0	0	0
T_2	0	+1	0	0	0
T_3	-1	0	-1	+1	+1
T_4	0	-1	-1	+1	+1
T_5	0	0	+1	-1	-1

D. Initial Transition Invariant

The transition invariant corresponds to a firing sequence that brings the network back to its initial or stable state. This means that each transition is executed a specific number of times to maintain the network's equilibrium.

The Conservation equations:

We ensure that tokens are conserved in the network by balancing the transitions. The invariant for the initial model is given by:

$$x1 * T1 + x2 * T2 + x3 * T3 + x4 * T4 = 0.$$

T_1 is fired once to move a token from P_1 to P_3.
T_1 is fired once to move a token from P_2 to P_3.
T_3 is fired once to move a token from P_3 to P_4.
T_3 is fired once to move a token from P_4 to P_5.

$$1 * T1 + 1 * T2 + 1 * T3 + 1 * T4 = 4.$$

This ensures that the system remains consistent.

5.2. Reduced Model

Reduction Technique Used: Place Fusion

We apply a fusion of similar places to simplify the processes related to task handling (Fig. 6). The queues for normal tasks (P_1) and urgent tasks (P_2) are merged into a single place (P_{queue}).

A. Places:

- P_{queue}: Single task queue
- P_3: Available server.
- P_4: Occupied server.
- P_5: Task being processed.

B. Transitions:

- T_1': Arrival of a task (normal or urgent).
- T_2': Start of task processing.
- T_3': End of task processing.

C. Initial Marking (M_0):

$M_0'(P_{queue})=0$, $M_0'(P_3)=1$, $M_0'(P_4)=0$, $M_0'(P_5)=0$

D. IM (Reduced Model)

The IM for the reduced model is defined as follows:

P_i / T_i	P_{queue}	P_3	P_4	P_5
T_1'	+1	0	0	0
T_2'	-1	-1	+	+1
T_3'	0	+	-1	-1

Transition Invariant (after)

In the reduced model, with a single queue (P_{queue}), only the transitions T_1', T_2', and T_3' are fired:

T_1' is fired to move a token from P_{queue} to P_3.
T_2' is fired to move a token from P_3 to P_4.
T_3' is fired to move a token from P_4 to P_5.

Reduced Transition Invariant:

With fewer transitions, the invariant becomes:
1 * T1' + 1 * T2' + 1 * T3' = 3.

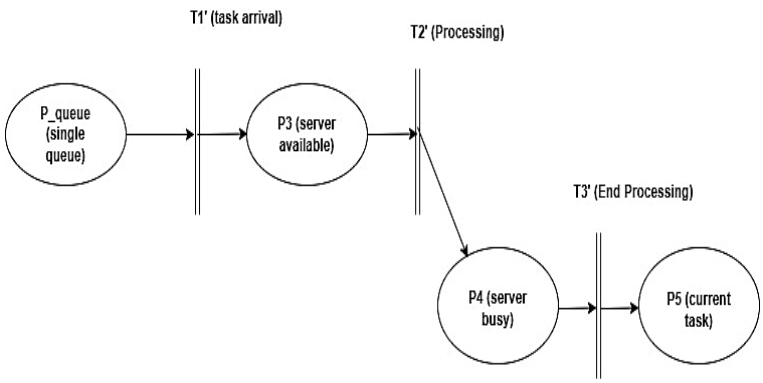


Fig. 6. Reduced PN model.

5.3. Comparison Before and After Reduction

From a model size perspective, the places number is optimised from 5 to 4, and the transitions number is optimised from 5 to 3. The token conservation properties are

satisfied in the network, and the reachable states in the initial model remain reachable in the reduced model. Moreover, the transitions remain traversable. Thus, the types of tasks (urgent/normal) are no longer differentiated in the reduced model, but this does not disrupt the overall functioning.

This model demonstrates how a reduction method (place fusion) can simplify a PN while preserving essential properties (conservation, liveness, and reachability). Formal methods ensure that information loss is minimised, which is crucial for modelling large-scale complex systems.

6. CONCLUSION

Through the case studies presented in this article, an innovative approach is introduced to overcome the limitations of classical reduction methods by integrating reachability methods into the model reduction process. Using the example presented, which involves a task management system modelled by a PN, we demonstrate how reachability methods enable the preservation of the reachability of critical states (such as states where the server is available or occupied) while achieving a significant simplification in the state space. In the initial model, although it was detailed and precise, the combinatorial explosion made analysis and verification resource-intensive. By applying reduction techniques based on reachability, we simplified the model while maintaining its essential properties of the system. For example, critical states related to task execution and queue management (both normal and urgent) were preserved in the reduced model, ensuring that key transitions (T_1', T_2', T_3') remained active and consistent with the expected behaviour. This integration of reachability methods offers two major advantages. First, it provides increased precision, ensuring that essential states for the system's analysis are not lost, ensuring consistency in the analysis. Second, optimisation of computational resources, where the reduction of the state space significantly reduces the computational load without compromising the accuracy the analytical results. Our approach demonstrates that it is possible to combine efficient reduction with the preservation of critical information, thereby opening promising perspectives for the study of large-scale complex systems. Through the case study presented in this article, we successfully modelled, using a PN, the dynamic allocation of three jobs across two machines. The created PN model is utilised to configure the SE and the mathematical model that describes the system behaviour. The linear algebraic technique, coupled with reduction methods, is employed to verify relevant properties. However, these approaches cannot be applied to all classes of PNs. The key contribution of this new approach lies in its ability to balance efficient reduction with analytical precision. Unlike traditional methods that sometimes sacrifice essential system properties, our method guarantees that all key transitions and states remain verifiable after reduction. Additionally, it optimises computational resources by reducing model complexity while maintaining comprehensive analysis. In future work, we plan to explore coloured PNs, which represent a natural evolution of ordinary PNs, in the field of optimising modelling tools for complex DES. Specifically, we aim to integrate both reduction aspects and greater expressiveness.

REFERENCES

1. Cassandras, C.G., & Lafortune, S. (2008). *Introduction to Discrete Event Systems*. Springer.
2. Kecir, K. (2014). *Contrôle optimal d'un système ferroviaire complet*. Université de Toulouse III-Paul Sabatier.
3. Boufaied, A. (2003). *Contribution* à la surveillance *distribuée des systèmes à* événements *discrets complexes*. Université Paul Sabatier-Toulouse III.
4. Cardin, O. (2016). *Contribution à la conception, l'évaluation et l'implémentation de systèmes de production cyber-physiques*. Université de Nantes.
5. Trinquet, Y., & Elloy, J-P. (2010). *Les systèmes d'exploitation temps réel: les principes. Techniques de L'ingénieur*.
6. Girault, C., & Valk, R. (2013). *Petri Nets for Systems Engineering: A Guide to Modeling, Verification, and Applications*. Springer Science & Business Media.
7. Seatzu, C., Silva, M., & Van Schuppen, J. H. (Eds.). (2013). *Control of Discrete-Event Systems*. (vol. 433). Springer.
8. Zaytoon, J., & Lafortune, S. (2013). Overview of Fault Diagnosis Methods for Discrete Event Systems. *Annual Reviews in Control, 37* (2), 308–320.
9. Ammour, R., Leclercq, E., Sanlaville, E., & Lefebvre, D. (2017). State Estimation of Discrete Event Systems for RUL Prediction Issue. *International Journal of Production Research, 55* (23), 7040–7057.
10. Norman, G., Parker, D., & Sproston, J. (2013). Model Checking for Probabilistic Timed Automata. *Formal Methods in System Design, 43*, 164-190.
11. Labadi, K. (2005). *Contribution à la modélisation et à l'évaluation de performances des systèmes logistiques à l'aide d'un nouveau modèle de réseaux de petri stochastiques*. Université de Technologie de Troyes.
12. Declerck, P., Chouchane, A., & Bonhomme, P. (2017). A strategy for estimation in timed Petri nets. In: *2017 4th International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE.
13. Declerck, P. (2021). Counter Approach for the Estimation of Optimal Sequences in Partially Observable Untimed Petri Nets. *Discrete Event Dynamic Systems, 31* (4), 489–512.
14. Khedher, A., & BenOthman, K. (2022). Estimation and Fault Detection of Discrete Event Systems Modeled by Time Interval Petri Nets. *Research Square*. https://doi.org/10.21203/rs.3.rs-1263538/v1
15. Gargantini, A., & Heitmeyer, C. (1999). Using Model Checking to Generate Tests from Requirements Specifications. *ACM SIGSOFT Software Engineering Notes, 24* (6), 146–162.
16. Faure, A., Poquet, M., & Richard, O. (2018). Évaluation *d'algorithmes d'ordonnancement par simulation réaliste*. hal-01779936.
17. Silva, M. (1980). *Simplification des réseaux de Petri par* élimination *de places implicites. Digital Processes, 6* (4), 245–256.
18. Liu, B., Ghazel, M., & Toguyeni, A. (2013). Edition spéciale MSR, Évaluation à la volée de la diagnosticabilité des systèmes a événements discrets temporisés. *Journal Européen des systèmes automatisés, 47*, 227–242.
19. Heitmeyer, C., & Mandrioli, D. (1996). *Formal Methods for Real-Time Computing*. John Wiley & Sons.
20. Alur, R., Courcoubetis, C., & Dill, D. (1990). Model-checking for real-time systems. In: *Fifth Annual IEEE Symposium on Logic in Computer Science*. IEEE.
21. Cheng, A.M. (2003). *Real-Time Systems: Scheduling, Analysis, and Verification*. John Wiley & Sons.
22. Norman, G., Parker, D., & Sproston, J. (2013). Model Checking for Probabilistic Timed Automata. *Formal Methods in System Design*, *43*, 164–190.

23. Kordon, F., Linard, A., Buchs, D., Colange, M., Evangelistika, S., Lampka, K., … & Wimmel, H. (2012). Report on the Model Checking Contest at Petri Nets 2011. *Transactions on Petri Nets and Other Models of Concurrency VI,* 169–196.
24. Reisig, W. (2012). *Petri Nets: An Introduction* (vol. 4). Springer Science & Business Media.
25. Benasser, A. (2000). *L'accessibilité dans les réseaux de Petri: une approche basée sur la programmation par contraintes*. Lille 1.
26. Barkaoui, K. (1988). *Contribution aux méthodes d'analyse des réseaux de Petri par la théorie des graphes*. Paris 6.
27. Girault, F. (1997). *Formalisation en logique linéaire du fonctionnement des réseaux de Petri*. Université Paul Sabatier-Toulouse III.
28. Popova-Zeugmann, L. (2013). *Time Petri Nets*. Springer.
29. Murata, T. (1989). Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE, 77* (4), 541–580.
30. Ogata, K. (2020). A Generic Approach on How to Formally Specify and Model Check Path Finding Algorithms: Dijkstra, A* and LPA. *International Journal of Software Engineering and Knowledge Engineering, **30*** (10), 1481–1523.
31. Kátai, Z., & Fülöp, P. I. (2010). *Modeling dynamic programming problems: Petri nets versus d-graphs*. In: *Proceedings of 8th International Conference on Applied Informatics (ICAI)*.
32. Augusto, V. (2012). Cours *Réseaux de Pétri*. École Nationale Supérieure des Mines de Saint-Etienne.
33. Lee-Kwang, H., Favrel, J., & Baptiste, P. (1987). Generalized Petri Net Reduction Method. *IEEE Transaction on Systems Man and Cybernetics, 17* (2), 297–303.
34. Sloan, R.H., & Buy, U. (1996). Reduction Rules for Time Petri Nets. *Acta Informatica*, *33*, 687–706.
35. Lee, K.-H., & Favrel, J. (1985). Hierarchical Reduction Method for Analysis and Decomposition of Petri Nets. *Transaction on Systems Man and Cybernetics*, *15* (2), 272–280.
36. Mahfoudhi, A., Hadj Kacem, Y., & Abid, M. (2011). Compositional Specification of Real Time Embedded Systems by Priority Time Petri Nets. *The Journal of Supercomputing, 59*, 1478–1503.
37. Karamti, W., & Mahfoudhi, A. (2014). Scheduling Analysis Based on Model Checking for Multiprocessor Real-Time Systems. *The Journal of Supercomputing*, *68*, 1604–1629.
38. Dutertre, B., & Stavridou, V. (2000). Formal analysis for real-time scheduling. In: *19th Digital Avionics Systems Conference. (Cat. No. 00CH37126)*. IEEE.
39. Ahmad, S., Malik, S., Ullah, I., Park, D.-H., Kim, K., & Kim, D. (2019). Towards the Design of a Formal Verification and Evaluation Tool of Real-Time Tasks Scheduling of IoT Applications. Sustainability, *11* (1), 204.
40. Souyris, J., Wiels, V., Delmas, D., & Delseny, H. (2009). Formal verification of avionics software products. In: *Formal Methods: Second World Congress* (pp. 532–546). 2–6 November 2009. Eindhoven, The Netherlands.
41. Choquet-Geniet, A., Geniet, D., & Cottet, F. (1996). Exhaustive computation of the scheduled task execution sequences of a real-time application. In: *Formal Techniques in Real-Time and Fault-Tolerant Systems: 4th International Symposium.* 9–13 September 1996. Uppsala, Sweden, Springer.
42. Hohmann, W. (2004). *Supporting Model-Based Development with Unambiguous Specifications, Formal Verification and Correct-By-Construction Embedded Software*. SAE Technical Paper.