Open Access Library Journal
2024, Volume 11, e11499
ISSN Online: 2333-9721
ISSN Print: 2333-9705

Implementation of an Iterative Algorithm for the Construction of Incidence Matrices for Generalized Petri Nets

Emile Niyongabo[1], Jérémie Ndikumagenge[2], Zihindula Mushengezi Elie[3], Hilaire Nkunzimana[4]

[1]Doctoral School, University of Burundi, Bujumbura, Burundi
[2]University of Burundi, Bujumbura, Burundi
[3]Institut Superieur Pédagogique de Bukavu, Bukavu, Congo
[4]Center for Research in Infrastructure, Environment and Technologie "CRIET", University of Burundi, Bujumbura, Burundi
Email: niyemi2014@gmail.com, jeremie.ndikumagenge@ub.edu.bi, eliezihindula@yahoo.fr, hilaire.nkunzimana@ub.edu.bi

How to cite this paper: Niyongabo, E., Ndikumagenge, J. and Elie, Z.M., Nkunzimana, H. (2024) Implementation of an Iterative Algorithm for the Construction of Incidence Matrices for Generalized Petri Nets. *Open Access Library Journal*, **11**: e11499.
https://doi.org/10.4236/oalib.1111499

Received: March 28, 2024
Accepted: December 27, 2024
Published: December 30, 2024

Abstract

The implementation of automated systems that meet the desired functional specifications requires scientifically proven modelling and design tools. The achievement of an automated computing system that meets the set criteria and technical specifications depends on several factors. These include the accuracy and the accuracy of the choice of modelling and design tools, the degree to which they are appropriate and adaptable to the application domain, the nature and setting of the application domain, the functional requirements and the available resources, to name but a few. This paper aims to extend the ontological basis of Petri nets by proposing an iterative algorithm for incidence matrix construction. The paper concludes with a discussion of the results focusing on the accuracy of the designed algorithm and its usability in the implementation of functional, reliable and robust automated systems.

Subject Areas

Modelling, Analysis, Processing and Automated Management of Information

Keywords

Petri Nets, Modelling, Iterative Algorithm, Incidence Matrix, Process

1. Introduction

The implementation of automated systems with specific functionalities and meeting certain reliability criteria requires modelling and simulation work at the pre-

design stage [1].

In terms of mathematical modelling of automated systems, including discrete event systems, engineers use a range of tools, some of which are centuries old and others recent. Examples include Markov chains, mathematical interpolation, dynamic programming, approximation, linear regressions, relational and/or linear algebra, graph theory, Petri nets, etc.

Among the formal mathematical modelling tools, Petri nets are recent. On the other hand, they have already proved to be adequate formal tools in the design and implementation of automated management systems.

Indeed, since their invention in the 1960s, Petri nets, rich in a vast and simple ontological base, offer developers of automated systems a range of methods for the formal study of the properties and thus functional specifications of the future system at the modelling stage [2].

The works of scholars and researchers such as Bouali Mohamed [3], have highlighted the importance of an a priori study of the behaviour of the future system at the pre-design stage and, therefore, of the technical and functional specifications of the future system before its actual implementation. Carried out upstream, this work aims to minimize design and implementation costs.

The modelling, design and synthesis activities of the future system are carried out by means of analysis methods that emanate from its ontological base, including coverage graphs, marking graphs, place and/or transition invariants through the use of the Gauss pivot method, etc. [3] [4].

In the present work, an iterative algorithm for the construction of an incidence matrix will be proposed. The last will support the study and analysis of most of the behavioural properties of the future system.

Finally, we develop an iterative algorithm for constructing the incidence matrix. We demonstrate a proof of correctness for algorithms with two nested loops, using mathematical induction.

For the algorithm developed, we demonstrated proof of correctness for algorithms with two nested loops using mathematical induction. This method of proving the correctness of an algorithm with two nested loops is new, as in other works, they were limited to a single loop.

2. Subject Matter, Tools, Materials and Methods

The work in this paper is based on a model of an automated random discrete event system represented as a generalized Petri net.

The methods of study and analysis of Petri nets, such as coverage and/or marking graphs and place and/or transition invariants, allow the properties or behaviours of the future automated system to be identified.

Further work will focus on a concrete case study in order to demonstrate the applicability of the results obtained and confirm the pragmatic aspect of this approach.

We intend to use some of these tools, means and methods applicable to Petri nets to formulate, state and develop an iterative algorithm for constructing an

incidence matrix from a Petri net.

The representation of a Petri net structure by means of the incidence matrix aims to shed light on the behavioural properties or specifications of the future system. The design of an iterative algorithm for the construction of an incidence matrix whose correctness is confirmed, and which requires an acceptable execution time leads us to resort to the methods of analysis of the algorithmic complexity as well as the techniques of proof of correctness of iterative algorithms including the invariants of loop which recalls the notion of the mathematical induction.

3. Results

3.1. Iterative Algorithm for the Construction of an Incidence Matrix of a Petri Net

To build this algorithm for the construction of an incidence matrix, we rely on a computational approach based on matrices called respectively before and after according to Equation (1) [3] [5].

$$W = W^{+} - W^{-} \tag{1}$$

This incidence matrix C is also called the action vector. Now, the forward incidence matrix, W^{-}, associated with a transition T_j indicating the conditions that react to the crossing of this transition depends on the current marking of M(p).

$$C^{-} = \begin{pmatrix} \text{For } i \leftarrow 1 \text{ to } n \\ \quad \text{For } j \leftarrow 1 \text{ to } m \\ \quad\quad \text{weight\_bow}\left(p_i, t_j\right) \\ \quad \text{End For } j \\ \text{End For } i \end{pmatrix} \tag{2}$$

$$C^{-} = \begin{pmatrix} e_{11} & \cdots & e_{1m} \\ \vdots & \ddots & \vdots \\ e_{n1} & \cdots & e_{nm} \end{pmatrix} \tag{3}$$

$$e_{ij} = \begin{cases} \omega\left(p_i, t_j\right) & \text{if the arc exists} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$w\left(p_i, t_j\right)$ is a specific value assigned to the arc that leaves p_i to t_j

The value of the weight of the arc is equal to 1 if there is no value assigned to this arc.

In other words, $w\left(p_i, t_j\right)$ defines the minimum number of tokens for the transition t_j to be traversable.

Incidence matrix after:

$$W^{+} = \begin{pmatrix} \text{For } i \leftarrow 1 \text{ to } n \\ \quad \text{For } j \leftarrow 1 \text{ to } m \\ \quad\quad \text{weight\_bow}\left(t_j, p_i\right) \\ \quad \text{End For } j \\ \text{End For } i \end{pmatrix} \tag{5}$$

$$W^{+} = \begin{pmatrix} e_{11} & \cdots & e_{1m} \\ \vdots & \ddots & \vdots \\ e_{n1} & \cdots & e_{nm} \end{pmatrix} \tag{6}$$

$$e_{ij} = \begin{cases} \omega\left(t_j, p_i\right) & \text{if the arc exists} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

The action vector W is based on the elements that effectively define a Petri net.

The determination of the incidence matrix before W^{-}, after W^{+} and of incidence W assumes that

$\forall n \in \mathbb{N}$ and $\forall m \in \mathbb{N}$, $\mathbb{N}$ being the set of natural numbers.

Where m is the number of transitions and n is the number of places.

$\forall i \in \{1, \cdots, n\}$ and $\forall j \in \{1, \cdots, m\}$.

Listing 1 below gives the sequence of instructions for the iterative algorithm for constructing or determining an incidence matrix associated with any Petri net.

```
1   Algorithm Calculating_Incidence_Matrix
2
3       Number i=1, j=1 ;
4       Number n ; // total number of places
5       Number m ; // total number of transitions
6       Number W[n][m] ;
7       Begin
8        repeat input(m); until (m>0);
9        repeat input(n); until (n>0);
10       while (i<=n) do
11               while (j<=m) do
12                   if(Arc from Pi to Tj does not exist)
13                       bow_weight(Pi, Tj)=0 ;
14                   Endif
15
16                   if(Arc from Tj to Pi does not exist)
17                       bow_weight(Tj, Pi)=0 ;
18                   Endif
19
20                   W[i][j]= bow_weight(Tj, Pi) - bow_weight(Pi, Tj) ;
21                   j=j+1 ;
22
23               EndWhileJ
24               i=i+1 ;
25       EndWhileI
26  EndAlgorithm
```

bow_weight$\left(P_i, T_j\right)$: This is the evaluation of the arc that connects the i^{th} place to the j^{th} transition.

bow_weight$\left(T_j, P_i\right)$: This is the evaluation of the arc that connects the j^{th} transition to the i^{th} place.

3.2. Accuracy and Correctness of the Algorithm

Correctness of the algorithm carries the idea of being correct and not that of correcting, which would imply identifying errors [6]. The process of checking the correctness of an algorithm is based on the concept of loop invariance [7]. The latter is a property composed of three essential elements—initialization, conservation and termination. Initialization defines a condition (precondition) that ensures that the algorithm is correct when entering the loop. Retention is a condition that ensures that the algorithm is correct when starting the "next" iteration; that is, that the operations performed during the previous iteration are correct in the sense of this algorithm. Finally, the termination is the condition (post condition) that ensures that the algorithm is correct at the end of the last iteration of the loop,

and at the exit of the loop.

The loop invariant is thus a concept of verification of algorithms that exploits the notions of mathematical induction.

3.2.1. Variant and Loop Invariant

By definition, a variant of a loop is an expression whose value varies at each iteration of the loop. A loop variant is used to prove that a "***While***" loop ends. In other words, it stops after a finite number of steps [7].

3.2.2. Accuracy of the Iterative Algorithm for Constructing the Incidence Matrix

Assertion "The iterative algorithm for constructing the incidence matrix of a generalized Petri net is correct, exact."

Let us prove this assertion using the method of mathematical induction, which is a method by recurrence.

- **Initialization**: the values of the variables n and m correspond to the total number of places and transitions respectively. A priori, $n \in \mathbb{N}_0 \cup \{\infty\}$ and $m \in \mathbb{N}_0 \cup \{\infty\}$. Before starting each of these two loops, i and j each contain 1. We then have values of the weights of the arcs that connect the transition to the square or square to the transition. The difference between the weight of the arc that exits the transition T_j to the square P_i and the weight of the arc that exits the same square to the same transition is noted as follows: $(T_j, P_i) - (P_i, T_j)$.
- **Conservation**: Let $i \in [1, n]$ and $j \in [1, m]$, assume $P([i], ([j], [j+1], \cdots, [j+m-1]))$. This means that we perform the first iteration of i and then successively iterations of $[j]$, $[j+1]$, $\cdots$, $[j+(m-1)]$, the elements of the first row of the matrix C will be positive or negative integers which are respectively the differences of the weights of the arcs (T_j, P_i) and (P_i, T_j), (T_{j+1}, P_i) and (P_i, T_{j+1}), $\cdots$, $(T_{j+(m-1)}, P_i)$ and $(P_i, T_{j+(m-1)})$. Thus we have: $W_{ij} = (T_j, P_i) - (P_i, T_j)$, $W_{i(j+1)} = (T_{j+1}, P_i) - (P_i, T_{j+1})$, $\cdots$, $W_{i(j+(m-1))} = (T_{j+(m-1)}, P_i) - (P_i, T_{j+(m-1)})$. With iteration $i+1$ and then successively iterations $j, j+1, \cdots, j+(m-1)$, the elements of the second row of the matrix C will have the following values respectively: $W_{(i+1)j} = (T_j, P_{(i+1)}) - (P_{(i+1)}, T_j)$, $W_{(i+1)(j+1)} = (T_{j+1}, P_{(i+1)}) - (P_{(i+1)}, T_{j+1})$, $\cdots$, $W_{(i+1)(j+(m-1))} = (T_{j+(m-1)}, P_{(i+1)}) - (P_{(i+1)}, T_{j+(m-1)})$. Hence $([(i+1)], ([j], [j+1], \cdots, [j+(m-1)]))$. We can continue the operations in the same way until $i = n$ or $i+(n-1)$. On exiting the loop, the values taken by the last line are respectively: $W_{(i+(n-1))j} = (T_j, P_{(i+(n-1))}) - (P_{(i+(n-1))}, T_j)$, $W_{(i+(n-1))(j+1)} = (T_{j+1}, P_{(i+(n-1))}) - (P_{(i+(n-1))}, T_{j+1})$, $\cdots$, $W_{(i+(n-1))(j+(m-1))} = (T_{j+(m-1)}, P_{(i+(n-1))}) - (P_{(i+(n-1))}, T_{j+(m-1)})$. Hence

$P\left(\left(\left[i+(n-1)\right]\right),\left(\left[j\right],\left[j+1\right],\cdots,\left[j+(m-1)\right]\right)\right)$. Hence the expected result for the last row of the matrix. So, the algorithm is correct.

• **Termination**: Indeed, thanks to the proof of the loop variant we can prove termination. Knowing that *i* starts at 1, $\forall n \in \mathbb{N}_0$ and $\forall i \in [1,n]$, $i \le n$. Knowing that *j* starts at 1, $\forall m \in \mathbb{N}_0$ and $\forall j \in [1,m]$, $j \le m$.

Thus, we have:

$$\sum_{i=1}^{n} i = n \quad \text{and} \quad \sum_{j=1}^{m} j = m \tag{8}$$

We have found a suitable loop variant, so according to the stopping theorem of each of two nested conditional loops, each loop ends. The integer quantities *i* and *j* are integer, positive, and strictly increasing by 1 up to *n* and *m* respectively. When the two quantities *i* and *j* reach *n* and *m* respectively, the two loops (***while*** ($\boldsymbol{i \le n}$) and ***while*** ($\boldsymbol{j \le m}$)) finish the job. By the time *i* becomes greater than *n* and *j* greater than *m*, the *n* places and *m* transitions have been effectively processed.

3.2.3. Complexity of the Developed Algorithm

Each algorithm has two main resources associated with it—execution time and memory space [8]. The study of algorithmic complexity aims at determining the execution time and memory space required for an algorithm to deliver the expected result. In this paper, we will deal with the concept of time complexity.

Time complexity is the number of elementary operations that an algorithm must perform to complete the job and provide the expected result when the size of the elements to be processed is very large, tends to infinity, and, therefore, in the worst case. The establishment of the order of complexity makes it possible to determine the behaviour of the algorithm when *n* tends to infinity [9].

For the developed algorithm, as shown in Listing 1, let $T(n)$ be the execution time, a function with one argument called *n*, *n* being the data size and c_i the time cost of line *i*. The worst-case complexity calculation is the time required to execute all lines and exit the loop (end of outer loop).

As shown in Listing 1, the start of the nested loop starts from line 10.

The cost of instructions 1 to 9 is a constant value, which depends on the characteristics of the computer. We will denote this constant as r.

The time complexity of the set of instructions executed in the nested loop is given by the expression or Equation (9)

$$T(n) = r + \sum_{i=1}^{n}\left(C_{10} + \sum_{j=1}^{m}\left(C_{11}+C_{13}+C_{14}+C_{16}+C_{17}+C_{18}+C_{20}+C_{21}\right) + C_{23} + C_{24} \right) \tag{9}$$

$$T(n) = r + \sum_{i=1}^{n}\left(C_{10}\right) + \sum_{i=1}^{n}\sum_{j=1}^{m}\left(C_{11}+C_{13}+C_{14}+C_{16}+C_{17}+C_{18}+C_{20}+C_{21}\right) + \sum_{i=1}^{n}\left(C_{23}+C_{24}\right) \tag{10}$$

$$T(n) = r + \sum_{i=1}^{n}\left(C_{10}+C_{23}+C_{24}\right) + \sum_{i=1}^{n}\sum_{j=1}^{m}\left(C_{11}+C_{13}+C_{14}+C_{16}+C_{17}+C_{18}+C_{20}+C_{21}\right) \tag{11}$$

$$T(n) = r + \left(C_{10}+C_{23}+C_{24}\right)\sum_{i=1}^{n}1 + \left(C_{11}+C_{13}+C_{14}+C_{16}+C_{17}+C_{18}+C_{20}+C_{21}\right)\sum_{i=1}^{n}\sum_{j=1}^{m}1 \tag{12}$$

$$T(n) = r + (C_{10} + C_{23} + C_{24}) \times n + (C_{11} + C_{13} + C_{14} + C_{16} + C_{17} + C_{18} + C_{20} + C_{21}) \sum_{i=1}^{n} m \quad (13)$$

$$T(n) = r + (C_{10} + C_{23} + C_{24}) \times n + (C_{11} + C_{13} + C_{14} + C_{16} + C_{17} + C_{18} + C_{20} + C_{21}) \times n \times m \quad (14)$$

Assuming that m equals n, then equation number (14) will become

$$T(n) = r + (C_{10} + C_{23} + C_{24}) \times n + (C_{11} + C_{13} + C_{14} + C_{16} + C_{17} + C_{18} + C_{20} + C_{21}) \times n^2 \quad (15)$$

Let $(C_{11} + C_{13} + C_{14} + C_{16} + C_{17} + C_{18} + C_{20} + C_{21}) = p$, $(C_{10} + C_{23} + C_{24}) = q$

Then

$$T(n) = pn^2 + qn + r \quad (16)$$

$$T(n) \in \theta(n^2) \quad (17)$$

Indeed, algorithms with two nested asynchronous loops, each ranging from 1 to n respectively and with the inner loop body being constant, have, in general, a quadratic complexity. Algorithms with polynomial complexity can also be considered efficient, provided that the power of the general term is not too large [10].

3.2.4. Example of the Use of the Algorithm on an Anonymous Generalized Petri Net

Let us consider an anonymous Petri net, as shown in **Figure 1**.

We can summarize in a two-dimensional table what the algorithm does for each iteration of the outer loop of these two nested "***While***" loops.

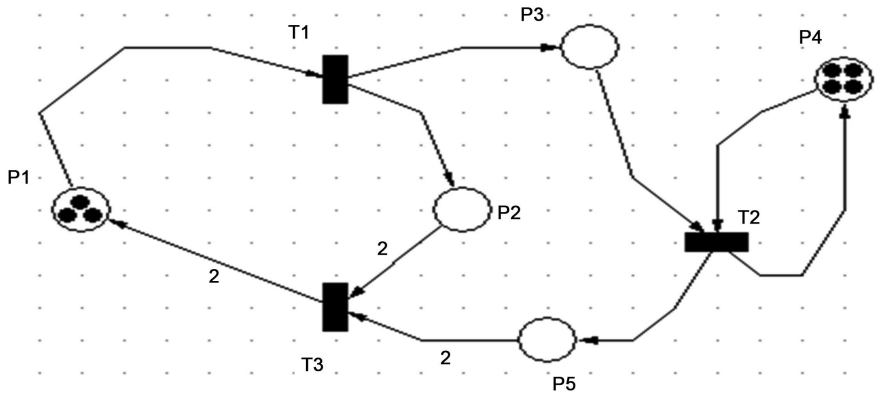

Figure 1. Example of a Petri net.

In the figure above, we have:

- Number of transitions $n = 3$;
- Number of places $m = 5$.
- The weights of the arcs:

- $(T_1, P_1) - (P_1, T_1) = 0 - 1 = -1$;
- $(T_2, P_1) - (P_1, T_2) = 0 - 0 = 0$;
- $(T_3, P_1) - (P_1, T_3) = 2 - 0 = 2$;
- $(T_1, P_2) - (P_2, T_1) = 1 - 0 = 1$;
- $(T_2, P_2) - (P_2, T_2) = 0 - 0 = 0$;

- $(T_3,P_2)-(P_2,T_3)=0-2=-2$;
- $(T_1,P_3)-(P_3,T_1)=1-0=1$;
- $(T_2,P_3)-(P_3,T_2)=0-1=-1$;
- $(T_3,P_3)-(P_3,T_3)=0-0=0$;
- $(T_1,P_4)-(P_4,T_1)=0-0=0$;
- $(T_2,P_4)-(P_4,T_2)=1-1=0$;
- $(T_3,P_4)-(P_4,T_3)=0-0=0$;
- $(T_1,P_5)-(P_5,T_1)=0-0=0$;
- $(T_2,P_5)-(P_5,T_2)=1-0=1$;
- $(T_3,P_5)-(P_5,T_3)=0-2=-2$;

With this example, we will use the concept of loop invariant to prove that the algorithm developed is accurate and correct.

We will describe the evolution of the values of the variables *i*—external loop sentinel that indicates places and *j*—internal loop sentinel that indicates transitions by applying it to the Petri net of Figure 1. Table 1 provides the execution protocol of the algorithm.

Table 1. Protocol for running the algorithm.

i \ *j*	1	2	3	4
1	$W_{11}=(T_1,P_1)-(P_1,T_1)=0-1=-1$	$W_{12}=(T_2,P_1)-(P_1,T_2)=0-0=0$	$W_{13}=(T_3,P_1)-(P_1,T_3)=2-0=2$	No value
2	$W_{21}=(T_1,P_2)-(P_2,T_1)=1-0=1$	$W_{22}=(T_2,P_2)-(P_2,T_2)=0-0=0$	$W_{23}=(T_3,P_2)-(P_2,T_3)=0-2=-2$	No value
3	$W_{31}=(T_1,P_3)-(P_3,T_1)=1-0=1$	$W_{32}=(T_2,P_3)-(P_3,T_2)=0-1=-1$	$W_{33}=(T_3,P_3)-(P_3,T_3)=0-0=0$	No value
4	$W_{41}=(T_1,P_4)-(P_4,T_1)=0-0=0$	$W_{42}=(T_2,P_4)-(P_4,T_2)=1-1=0$	$W_{43}=(T_3,P_4)-(P_4,T_3)=0-0=0$	No value
5	$W_{51}=(T_1,P_5)-(P_5,T_1)=0-0=0$	$W_{52}=(T_2,P_5)-(P_5,T_2)=1-0=1$	$W_{53}=(T_3,P_5)-(P_5,T_3)=0-2=-2$	No value
6	No value	No value	No value	No value

Finally, we have the following incidence matrix: $W=\begin{pmatrix}-1 & +0 & +2\\ +1 & +0 & -2\\ +1 & -1 & +0\\ +0 & +0 & +0\\ +0 & +1 & -2\end{pmatrix}$

4. Discussions

The correctness of the iterative incidence matrix construction algorithm guarantees that the algorithm is mathematically correct and scientifically valid for use. Furthermore, most of the algorithms with two nested loops whose complexity is a quadratic function are simple and easy to use. The effectiveness and efficiency of this algorithm offer multiple advantages.

Indeed, the optimal use of the processor's time attested by the smallness of the polynomial complexity found and its efficiency in the worst case, guarantees the achievement of the expected result within the limits of the available resources. Moreover, the simplicity of searching for an element by the indexed access approach

and the ease of implementation reinforce the applicability of the result obtained in the modelling and design of automated systems.

Ongoing work on the possibility of representing Petri nets as linear linked lists will reaffirm the pragmatic aspect and usability of the latter.

5. Conclusions

The iterative algorithm for constructing the incidence matrix of a Petri net simplifies and facilitates the modelling and synthesis of an automated system.

Contrary to sequential access, indexed access to the elements entering into the composition of a Petri net constitutes an important methodological contribution to the analysis and synthesis of discrete event automated systems. As it is not greedy, and therefore requires reasonable and acceptable spatial and especially temporal resources, its adoption and implementation in formal Petri net-based modelling tools and platforms would bring added value.

The algorithm developed for the construction of the incidence matrix is accurate and correct for use. In future work, we will extend the ontological basis of Petri nets by introducing the tools needed to traverse the different states of the Petri net modelled system and reconfigure the existing Petri net modelled system by modifying or adding a node.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] Rivière, N. (2003) Modélisation et analyse temporelle par réseaux de Petri et logique linéaire. Doctoral Dissertation, INSA de Toulouse.

[2] Peterson, J.L. (1981) Petri net theory and the modeling of systems. Prentice Hall PTR.

[3] Bouali, M. (2009) Contributions à l'analyse formelle et au diagnostic à partir de réseaux de Petri colorés avec l'accessibilité arrière. Doctoral Dissertation, Université de Technologie de Compiègne.

[4] Chen, H. (2018) Block Decompositions and Applications of Generalized Reflexive Matrices. *Advances in Linear Algebra & Matrix Theory*, **8**, 122-133. https://doi.org/10.4236/alamt.2018.83011

[5] Sun, Y., Zhang, H. and Li, C. (2018) Generalized Irreducible A-Matrices and Its Applications. *Advances in Linear Algebra & Matrix Theory*, **8**, 111-121. https://doi.org/10.4236/alamt.2018.83010

[6] Defour, D. (2003) Fonctions élémentaires: Algorithmes et implémentations efficaces pour l'arrondi correct en double precision. Doctoral Dissertation, Ecole normale supérieure de lyon.

[7] Parreaux, J. (2018-2019) Leçon 927: Exemples de preuves d'algorithmes: correction et terminaison.

[8] Laskri, M.T. and Boudour, R. (2007) Outil de partitionnement hw/sw basé sur l'algorithme Kernighan/Lin amélioré. *Revue Africaine de la Recherche en Informatique et Mathématiques Appliquées*, **7**, 20-40.

[9] Van de Wiele, J.P. (1976) La complexité du calcul des polynômes.

[10] Jguirim, W., Naanaa, W. and Cooper, M.C. (2015) Sur une classe polynomiale relationnelle pour les CSP binaires.

List of Variables Used

W : incidence matrix

W^- : incidence matrix before

W^+ : incidence matrix after

i: variable for scrolling through the rows of the incidence matrix

j: variable for scrolling through the columns of the incidence matrix

e_{ij} : element of the incidence matrix

P: places

T: transitions

$\omega\left(p_i, t_j\right)$: weight of the arc of place p_i to the transition t_j

n: is the number of places.

m: is the number of places

c_i : cost of the ith line

$W[n][m]$: incidence matrix with variable n as the total number of rows and variable m as the total number of columns