ANALYSIS OF STATES IN PRODUCTION SYSTEMS MODELED BY PETRI NETS USING DIOPHANTINE EQUATIONS

Julián Gómez-Munilla[a], Emilio Jiménez-Macías [b], Juan-Ignacio Latorre-Biel [c], Mercedes Pérez de la Parte[a] , Jorge Luis García-Alcaraz[d]

[a] University of La Rioja. Industrial Engineering Technical School.
Department of Mechanical Engineering. Logroño, Spain
[b] University of La Rioja. Industrial Engineering Technical School.
Department of Electrical Engineering. Logroño, Spain
[c] Public University of Navarre. Department of Mechanical Engineering, Energetics and Materials.
Campus of Tudela, Spain
[d] Technology and Engineering Institute, Universidad Autónoma de Ciudad Juárez, Chihuahua (Mexico)

(a) julian.gomez@unirioja.es , mercedes.perez@unirioja.es , (b) emilio.jimenez@unirioja.es ,
(c) juanignacio.latorre@unavarra.es , (d) jorge.garcia@uacj.mx

ABSTRACT

This paper is devoted to study ways to determine the possible states of a production system, modeled by Petri nets (PN), making use of techniques for solving systems of linear Diophantine equations. For this porpoise, PNs have been divided into three possible types, and each provides a specific method so as to exploit its characteristics to optimize the computation time. These types are conservative systems, nonconservative systems with conservative components, and nonconservative systems without conservative components. These proposal methods have been compared with the determination of states from marking evolution, being clearly advantageous in computation time.

Keywords: Petri Net, Discrete Event Systems, Diophantine Equations, State equation

1. INTRODUCTION

Petri Nets constitutes a family of formalisms with mathematical-graph duality, which allows to efficiently representing discrete event systems, especially when they presents concurrency and shared resources. The study of the states of the systems being modeled is needed to understand its behavior and properties, so that knowing all states of the PN (the marking) is a critical task.

This paper aims to explore ways to determine the possible states of a production system (Jimenez et *al.*, 2006, 2009, 2012), modeled by PN (Latorre et *al.*, 2009, 2013a) and analyzing its states using resolution techniques (Latorre et *al.*, 2013b, 2013c) of linear Diophantine equations systems. PNs are classified into 3 types, and each provides a specific method so as to exploit its characteristics to optimize the computation time. These types are: conservative systems, but nonconservative systems with conservative components, and nonconservative systems without conservative components. These proposal methods have been compared with the determination of states from marking evolution, being clearly advantageous in computation time.

2. PETRI NETS

Petri nets are formed by a set of places (P), another of transitions (T), and arcs (F), each with a given weight (W), connecting places transitions and vice versa. The set of places, transitions, arcs and weights defines the structure of the system (Murata, 1989; Silva, 1993; Girault and Valk, 2001).
Graphically, places are represented by circles, transitions by bars or rectangles, and arcs with arrows. To indicate the weight of the arcs, when greater than one, numbers are typically placed beside the arrows. The marking of the net is represented by tokens (points) in the places, or with numbers if the number of tokens is high.

The PN structure is represented by the incidence matrix, where rows represent different places and transitions columns. The elements of the matrix indicate the number of tokens that appear (if positive) or disappear (if negative) in each place -row- when a transition-column- is fired. The PN marking is represented by a column vector that has so many components as places in the PN.

The state equation of a PN determines which is the marking (state) reached after firing a number of transitions from an initial marking.

$$m = m_0 + C \cdot \sigma \qquad (1)$$

Being:
- m: Final marking vector
- m_0: Initial marking vector
- C: Incidence Matrix
- σ: Firing verctor**.** Column vector with as many rows as transitions in the system. Their values represent the number of times that each transition is fired.

The state equation does not take into account whether the firing vector is really applicable, since it is not considered the order in the firings (possible states could be reached from the point of view of that equation, which could not be reached by a sequence firing of transitions).

The properties of the PN can be of two types: structural and dynamic. The main difference is that the first ones depend only on the structure of the system, while the latter also depends on the initial marking.

Structural properties, being only dependent on the structure, can be determined mathematically. Among these we highlight 2, which are important for this analysis:

Conservativeness: a PN is conservative if, for any marking, the sum of all their marking in the places, each multiplied by a factor, remains constant. Mathematically, if the vector of factors or weights:

$$W \cdot m = W \cdot (m_o + \mathrm{C} \cdot \sigma) = W \cdot m_o + W \cdot \mathrm{C} \cdot \sigma \qquad (2)$$

To satisfy the condition $W \cdot m = W \cdot m_o$, it is necessary that $W \cdot \mathrm{C} \cdot \sigma = \mathbf{0}$, and since the firing vector is indifferent because it is a structural property, observe that:

$$W \cdot \mathrm{C} = \mathbf{0} \qquad (3)$$

The vector or vectors W that satisfy the equation (3) are left cancellers of the incidence matrix, or labeled invariant or conservative components.

For the PN is conservative, all sites must be contained in (or "covered" by) any marking invariant component or conservative component. If there are places not covered, the PN is not conservative, although the covered places do present conservation propertires.

Repeatability: A PN is repetitive if there exists a sequence of firings of all transitions that returns the system to an initial marking. Mathematically, if σ_r is the firing vector that returns the PN to the initial marking:

$$m = m_o + \mathrm{C} \cdot \sigma_r \qquad (4)$$

To satisfy the condition $m = m_o$, it is necessary that:

$$\mathrm{C} \cdot \sigma_r = \mathbf{0} \qquad (5)$$

The vector or vectors σ_r satisfy the equation (5) are right cancellers of the incidence matrix, or firing invariant, or or repetitive components.

So that the PN be repetitive, all transitions must be contained in (or "covered" by) any firing invariant or repetitive component. If there are uncovered transitions, the PN is not repetitive, although the covered transitions are.

The repeatability property is not used as such in this work, although it has been considered for its close relationship with conservativeness (they are the same for the dual PN).

3. DIOPHANTINE EQUATIONS

3.1. Definition

A Diophantine equation is one that admits only as solution an integer. Specifically, diophantine linear equations have the form:

$$a_1 \cdot x_1 + a_2 \cdot x_2 + \ldots + a_n \cdot x_n = b \qquad (6)$$

Where $a_1, a_2, \ldots, a_n$, and b are known integers and $x_1, x_2, \ldots$, and x_n are unknown integers. If $b = \mathbf{0}$, it is said that the Diophantine equation is homogeneous. A Diophantine equations linear system is a system in which equations are of the form (6).

3.2. Euclidean algorithm and Bezout identity

The resolution of a linear Diophantine equation is based on the Euclidean algorithm, which is used to calculate the greatest common divisor of two integers (Ajili and Contejean, 1995, Bradley, 1971; Havas et *al*., 1998; Hemmecke, 2011; Lazebnik, 1996). There is the property that, if d is the greatest common divisor of a and b, then there are two integers x and y such that the reste of the division $\frac{a \cdot x + b \cdot y}{d}$ is zero.

Bézout's identity says that there are two integers, m and n such that:

$$d = a \cdot m + b \cdot n \qquad (7)$$

That is, the greatest common divisor of a and b can be represented as a linear combination of these two integers a and b.

3.3. Algorithms for solving Diophantine equations

There are, at present, several algorithms for solving systems of Diophantine equations. Some of them use the so-called Hermite normal form, and others are based on repeatedly test possible solutions. The algorithm used in this work to solve systems of equations is of the second type, because they do not want to find all solutions, but only those who are between zero and a limit (Clausen and Fortenbacher, 1989; Contejean and Devie, 1994), either because the system is conservative (and if so inherently limited), or because it is not but our intention is to determine progressively the states, by including limits in the not limited places and increasing these limits prograsively to build the reachability tree in an structured and organized way.

3.4. Application of Diophantine equations to PN

Diophantine equations appear many times in the mathematical resolution of PN. Both the firing vector as the marking vector are composed always by non-negative integers, and therefore any equation or system of equations in which one of the two vectors is the unknown is a diophantic equation.

4. PROPOSED METHODS

Following they are presented a series of methods that aim to solve the problem of identifying all possible state in a Petri net from an initial marking, depending on the type of PN concerned.

4.1. General methods

This case can be used in all types of PN, and is the system used to non-conservative systems without marking invariants; for those who do have conservative componentsn other methods will be proposed in the following sections.

4.1.1. Resolution of the state equation

This method is valid for conservative Petri nets and for non conservative. It consists of applying the state equation to all possible marking and selecting those for which there are non-negative integer solutions for the firing vector.

Let be a PN with n places, all of them with the marking limited between 0 and a maximum (*lim*) number of tokens. Therefore, there appear $(lim + 1)^n$ possible markings. This is equivalent to having a word of n letters, where each letter can take the values $\{0,1,\dots,lim\}$, what is known in combinatorics as variations with repetition of $lim + 1$ elements taken from n to n or $VR(lim + 1, n)$.

For each of the possible markings, it is necessary to verify the existence of a firing vector between the initial marking and studied marking. A number of $(lim + 1)^n$ linear Diophantine equations systems should be solved, with the form:

$$m = m_o + C \cdot \sigma \Rightarrow C \cdot \sigma = m_o - m \qquad (8)$$

In all of them, C and m_o are similar. The analysed marking m belongs to the system iff there exists a vector σ, whose components are non-negative integer and such that satisfies the equation (8).

The advantage of this method is that all checked possible markings within the limits are obtained, including those that are not achievable by evolution from the initial marking by firing successive transitions (spurious states). The main drawback is the time spent, because $(lim+1)^n$ different systems of linear Diophantine equations should solved. This makes it infeasible in many cases the use of this method.

4.1.2. Evolution of successive shots marked by transitions

This method is also applicable to conservative and non-conservative PN. It consists on firing every transition from the initial marking to determine the following markings. A marked with a negative number of marks somewhere means that bthe state is not possible.

Let be a PN with n places where each marking can be between 0 and a maximum (*lim*) of tokens, and m transitions; using the state equation:

$$m = m_o + C \cdot \sigma \quad \Rightarrow \qquad (9)$$

$$\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_{1,0} \\ \vdots \\ x_{n,0} \end{bmatrix} + \begin{bmatrix} c_{11} & \cdots & c_{1m} \\ \vdots & \ddots & \vdots \\ c_{n1} & \cdots & c_{nm} \end{bmatrix} \cdot \begin{bmatrix} t_1 \\ \vdots \\ t_m \end{bmatrix}$$

Applying only one firing from an only transition t_j, the firing vector presents this form:

$$\begin{bmatrix} t_1 \\ \vdots \\ t_j \\ \vdots \\ t_m \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \qquad (10)$$

Then, equation (9) is equivalent to (11):

$$\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_{1,0} \\ \vdots \\ x_{n,0} \end{bmatrix} + \begin{bmatrix} c_{11} & \cdots & c_{1m} \\ \vdots & \ddots & \vdots \\ c_{n1} & \cdots & c_{nm} \end{bmatrix} \cdot \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} x_{1,0} \\ \vdots \\ x_{n,0} \end{bmatrix} + \begin{bmatrix} c_{j1} \\ \vdots \\ x_{jn} \end{bmatrix} \qquad (11)$$

I.e., firing only once the transition t_j is equivalent to to adding the column j of the incidence matrix.

A firing of a new transition is applied to the markins so obtained, in order to obtain a new set of markings.

Naming m_1 the set of valid markings obtained after firing once every transition from m_o; m_2 the set of valid markings obtained after firing once every transition from m_1, etc.:

$$\begin{aligned} m_1 &= m_0 + C \cdot \sigma \\ m_2 &= m_1 + C \cdot \sigma \\ &\dots \\ m_n &= m_{n-1} + C \cdot \sigma \end{aligned} \qquad (12)$$

The process stops when no new markings are obtained. This situation is caused by any of the following reasons:

1) Because it is not possible to fire any transition (there are deadlocks).
2) Because it is possible to fire any transition but would exceed the limit marks (valid states but not desired).
3) Because it is possible to fire any transition but this would already drive to previous markings.

It is possible, however, did not obtain all valid markings (with a number of tokens between 0 and *lim*). For example, you can reach invalid marking as follows:

$$m_n = \begin{bmatrix} a \\ b \\ \lim + k \\ c \end{bmatrix}; k > 0 \qquad (13)$$

There may be a column in the incidence matrix C such that:

$$c_{1:n,j} = \begin{bmatrix} d \\ e \\ -k \\ f \end{bmatrix}; |d| \le a, |e| \le b, |f| \le c \qquad (14)$$

When firing the transition j from m_n, the following marking would be obtained:

$$m_{n+1} = \begin{bmatrix} a+d \\ b+e \\ \lim \\ c+f \end{bmatrix} \qquad (15)$$

is clearly valid, as the number of tokens in all places is between 0 and, *lim* and there is no reason for having being reached previously.

A tolerance to accept provisionally markings with a number of tokens bigger than the limit can be allowed, then removing them from the final results. The advantage of this method is that it is remarkably fast: it is only necessary to sum column vectors and discard the result if an element is negative; there is no need to solve any equation.

The main drawback is that it is impossible to determine spurious states, since they, by definition, are not achievable by successive firing of transitions from the initial marking.

4.1.3. Identification of spurious markings

The application of the two methods shown so far, permits us to identify which marked are spurious and which are not.

Calling M the set of marked obtained by solving the equation of state in the $(lim + 1)^n$ possible markings (or by other means, as discussed later), and M_t the set of marked not spurious (obtained by initial developments by firing transitions) is clear that:

$$M_t \subseteq M \tag{16}$$

Furthermore, if M_s is the set of labeled spurious:

$$M_t \cap M_s = \emptyset;\ M = M_t \cup M_s \rightarrow$$
$$M_s = \{ M \setminus M_t \} \tag{17}$$

That is, all the markings obtained by solving the state equation that have not been obtained by evolution by firing transitions are spurious markings.

4.2. Conservative PN

Conservative Petri Nets have marking invariants covering all places. So there are mathematical laws that apply to all possible markings, spurious or not. The conservative PN are also limited. It is convenient to take advantage of this fact to determine all possible markings.

Let be a conservative PN with n places, whose q conservative components or marking invariants are the rows of a matrix W. By definition:

$$W \cdot C \cdot \sigma = 0 \Leftrightarrow W \cdot m = W \cdot m_0 \tag{18}$$

Then:

$$\begin{bmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{q1} & \cdots & w_{qn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{q1} & \cdots & w_{qn} \end{bmatrix} \cdot \begin{bmatrix} x_{1,0} \\ \vdots \\ x_{n,0} \end{bmatrix} =$$
$$= \begin{bmatrix} w_{1,1:n} & \times & w_{1:n,0n} \\ \vdots & \vdots & \vdots \\ w_{q,1:n} & \times & w_{q:n,0n} \end{bmatrix} \tag{19}$$

The only unknown in (19) is the vector $m = [x_1,\ldots x_i,\ldots,x_n]^T$. It has to meet that $0 \le x_i \le lim\ \forall\ i \in \{1,\ldots,n\}$. Then, (19) is a system of linear Diophantine equations whose solutions are the possible markings of the system, including the spurious ones.

The advantages of this method are clear: after obtaining the conservative components, a system of linear Diophantine equations must be solved to determine all markings, including spurious.

The main disadvantage is precisely that it only applies to conservative systems. A modification of the method may, however, greatly simplify determining markings in nonconservative PN with marking invariants.

4.3. Nonconservative PN with marking invariants

A Petri net with marking invariants is not conservative if there are places "not covered", ie if the vectors indicating the conservative components have one or more columns whose value is null in all the elements. It can be taken advantage of the fact that the places are covered in the conservative components (and therefore the PN itself is limited in those places) to greatly simplify the search for markings.

Let be a Petri net with n places and m transitions, whose q conservative components or marking invariants are the rows of a matrix W. Let p be the number of locations covered by the conservative components, and $n\text{-}p$ the number of places not covered. Then, the set P of index of places can be divided into two subsets: P_{n-p} the set of indexes of places covered, and P_{n-p} with indexes of the places not covered. So:

$$P_n \cap P_{n-p} = \emptyset;\ P_p \cup P_{n-p} = P \tag{20}$$

Then eliminating the places not covered by the conservative components (or what is the same, eliminating the columns of zeros):

$$\begin{bmatrix} w_{1,1} & \cdots & w_{1,i} & \cdots & w_{1,n} \\ \vdots & & \vdots & & \vdots \\ w_{q,1} & \cdots & w_{q,i} & \cdots & w_{q,n} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{bmatrix} =$$
$$= \begin{bmatrix} w_{1,1} & \cdots & w_{1,i} & \cdots & w_{1,n} \\ \vdots & & \vdots & & \vdots \\ w_{q,1} & \cdots & w_{q,i} & \cdots & w_{q,n} \end{bmatrix} \cdot \begin{bmatrix} x_{1,0} \\ \vdots \\ x_{i,0} \\ \vdots \\ x_{n,0} \end{bmatrix} \forall i \in P_p \tag{21}$$

The only unknown in (21) is the vector $m = [x_1,\ldots x_i,\ldots,x_n]^T\ \forall\ i \in P_p$. That must be met $0 \le x_i \le lim$ $\forall\ i \in P_p$. Then (21) is a system of linear diophantine equations whose solutions are the set of possible markings of the places covered by the conservative components.

Let be M_p the set of possible markings of places covered by the conservative components. For each of them there is a firing vector such that, applied to the initial marking in the covered places, brings the system to a final marking $m \in M_p$.

To apply the state equation to the places of P_p, it is necessary to eliminate from the incidence matrix the rows corresponding to the places not covered by conservative components (22).

The only unknown in (22) is the firing vector $\sigma = [t_1,\ldots,t_m]^T$, whose values must be non-negative integers. Then (22) is a linear system of Diophantine equations to be applied to each marking obtained in (21).

Let be J the set of vectors $\sigma = [t_1,\ldots,t_m]^T$ that satisfy (22) for all $m \in M_p$, ie all possible firing vectors that, applied to the initial marking, lead to a marking $m \in M_p$.

It is necessary to check that the various firing vectors J are also applicable to places of the set P_{n-p} , ie, the *n-p* places not covered by the conservative components.

$$\begin{bmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_{1,0} \\ \vdots \\ x_{i,0} \\ \vdots \\ x_{n,0} \end{bmatrix} + \begin{bmatrix} c_{1,1} & \cdots & c_{1,m} \\ \vdots & & \vdots \\ c_{i,1} & \cdots & c_{i,m} \\ \vdots & & \vdots \\ c_{n,1} & \cdots & c_{n,m} \end{bmatrix} \cdot \begin{bmatrix} t_1 \\ \vdots \\ t_m \end{bmatrix}$$

$$\forall m = \begin{bmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{bmatrix} \in M_p, \forall i \in P_p \qquad (22)$$

Simply apply the state equation, this time to all places:

$$m = m_0 + C \cdot \sigma \quad \forall \ \sigma \in J \ \ => \qquad (23)$$

$$\begin{bmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_{1,0} \\ \vdots \\ x_{i,0} \\ \vdots \\ x_{n,0} \end{bmatrix} + \begin{bmatrix} c_{1,1} & \cdots & c_{1,m} \\ \vdots & & \vdots \\ c_{i,1} & \cdots & c_{i,m} \\ \vdots & & \vdots \\ c_{n,1} & \cdots & c_{n,m} \end{bmatrix} \cdot \begin{bmatrix} t_1 \\ \vdots \\ t_m \end{bmatrix}; \forall \sigma = \begin{bmatrix} t_1 \\ \vdots \\ t_m \end{bmatrix} \in J, \forall i \in P$$

Let M_t be the set of markings obtained after applying (23). So marked as many vectors are obtained from the above equation as firing vectors has been tested:

$$|M_t| = |J| \qquad 24)$$

It is necessary to verify that these markings are invalid. Clearly covered places will have a valid marking because the firing vectors $\sigma \in J$ were obtained from them. However, it is possible that the places not covered do not have a valid marking. It should be checked for all markings $m \in M_t$ such that $0 \le x_i \le lim$ $\forall\ i \in P_{n-p}$, and discard markings that does not fulfill this condition, as well as the firing vectors that have led the system that led to them.

The non-rejected firing vectors must be applied again to the not discarded markings until any one exceeds the defined limit.

The proposed method is summarized in the following steps:
1) Regardless of the places not covered by the conservative components, find all possible markings solving the system of linear Diophantine equations (21).
2) By applying the state equation for each possible marking, again disregarding the places not covered, draw the firing vector (22).
3) Taking into account all places, apply firing vectors obtained in the previous section to the initial marking.
4) Discard the markings obtained in (23) that are not possible for having a negative number somewhere or exceeding the bound, as well as the firing vectors that led to them.
5) Continue implementing the non rejected firing vectors to the valid markings and to those obtained from them, up to reach the limits defined or even not be possible to continue applying them.

The advantages of this method with respect to the general methods is that it considerably reduces the number of equations to be solved, since it takes advantage of the conservative components (taking into account the limitation of marking locations covered). Another advantage is that it allows obtaining the spurious markings.

4.4. Recommendations on the proposed methods

In view of the described methods, the mode of operation to determine the states of a PN may be as follows:

- First, you must determine the type of PN in question: Conservative, non conservative but with marking invariants, or non conservative without marking invariants.
- In case there are invariant markings, determine all possible markings by the specific methods proposed. Apply the method of "evolution of markings by successive firing of transitions" to determine which markings are achievable. The possible markings that are not achievable, are spurious.
- In the case that the PN has no marking invariants, there is a "fast" method to determine all possible markings. It may be a reasonable option to search only those that are achievable by firing transitions. However, in PN with a limit of small tokens and few places, and always depending on the time and resources available, it may be applicable the general method of "Solving the state equation".

5. EXAMPLE OF APPLICATION

To exemplify the approach consider a production system with a robot, which takes pieces from two conveyor belts, each with a part type, and fed to two machines which make two types of products with those pieces (Fig.1).

The PN that models that process is presented in Fig2. The numbering of places and transitions, not included to simplify the drawing, is in both from top to bottom and from left to right. Thus the incidence matrix is as shown in Table 1, and the initial marking is:

m_0=[1,1,1,0,0,2,1,1,2,0,0,0,0,0,0]^T

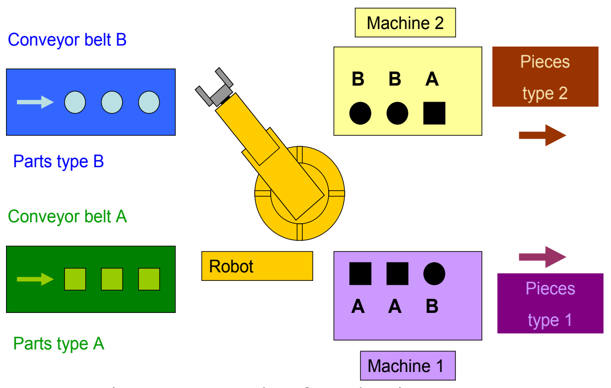

Figure 1: Sample of production system

Table 1: Incidence matrix of Pn in Fig. 2

-1	0	1	1	0	0	0	0	0	0
0	-1	0	0	1	1	0	0	0	0
-1	-1	1	1	1	1	0	0	0	0
1	0	-1	-1	0	0	0	0	0	0
0	1	0	0	-1	-1	0	0	0	0
0	0	-1	0	0	0	0	0	2	0
0	0	0	-1	0	0	0	0	0	1
0	0	0	0	-1	0	0	0	1	0
0	0	0	0	0	-1	0	0	0	2
0	0	1	0	0	0	-2	0	0	0
0	0	0	0	1	0	-1	0	0	0
0	0	0	1	0	0	0	-1	0	0
0	0	0	0	0	1	0	-2	0	0
0	0	0	0	0	0	1	0	-1	0
0	0	0	0	0	0	0	1	0	-1

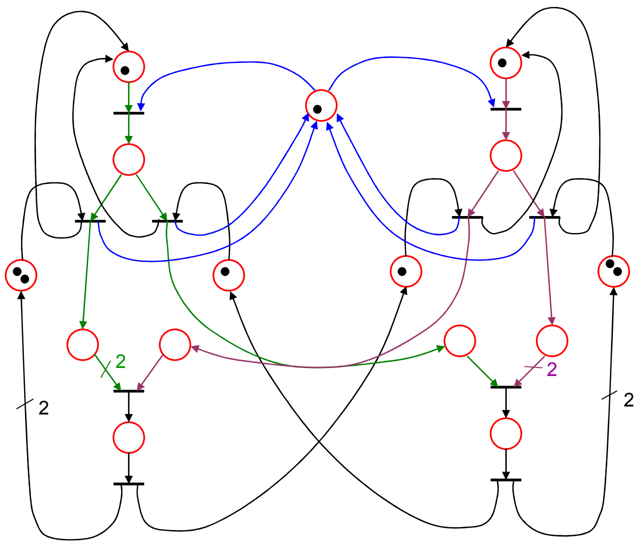

Figure 2: PN modeling system in Fig.1

From the graphic of the PN, or from the state equation, alternatively, we can study the reachability tree of the system, resulting in 147 states, simply using the method of successive firings.

Furthermore, calculating the marking invariant gives us the following seven solutions for the conservative components (table2):

Table 2: Conservative components (rows) of PN in Fig. 2

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	1	0	0	0	2	0
0	0	0	0	0	0	0	1	0	0	1	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	1	0	0	1
0	0	0	0	0	0	0	0	1	0	0	0	1	0	2

No column is zero, what implies that the network is conservative, and can be applied the proposed methodology for conservative PN. Therefore, the Diophantine equations are solved following the Contejean-Devie algorithm, and the 147 possible solutions are obtained, but with less computational time and effort (a summary of the states is presented in Table3, reduced to just 10 states for space cuestions).

Table 3: Summary of the 147 states of the system in Fig. 1 provided by the Contejean-Devie algorithm

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
1	0	1	0	1	0	0	0	0	0	0	0	0	0	1	1
2	0	1	0	1	0	0	0	0	0	0	0	1	2	1	0
3	0	1	0	1	0	0	0	0	0	2	1	0	0	0	1
4	0	1	0	1	0	0	0	0	0	2	1	1	2	0	0
5	0	1	0	1	0	0	0	0	1	0	0	1	1	1	0
6	0	1	0	1	0	0	0	0	1	2	1	1	1	0	0
7	0	1	0	1	0	0	0	0	2	0	0	1	0	1	0
8	0	1	0	1	0	0	0	0	2	2	1	1	0	0	0
9	0	1	0	1	0	0	0	1	0	2	0	0	0	0	1
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
147	1	1	1	0	0	2	1	1	2	0	0	0	0	0	0

6. CONCLUSIONS

The paper has presented a study on the different techniques to determine the states of a production system by analyzing the Petri net that models it. Two general methods have been discussed, one slower and depth that allows obtaining "spurious marked" and another one considerably faster but not allowing obtaining them.

The structure of Petri Nets may have different characteristics, so as to be classified into three types for those propouses: conservative to all places, nonconservative globally but for conservative for some places, and non-conservative to any place. The existence of conservation laws for all or some places considerably reduces the search for possible PN markings (ie, possible states of the system), thanks to new Diophantine equations systems that appear whose solutions are directly the potential markings. Accordingly, two methods have been described specifically for those Petri nets with these characteristics.

For the resolution of systems of Diophantine equations, the most common today algorithms have been examined, and the one which best fits the type of solutions expected in Petri Nets has been chosen.

ACKNOWLEDGMENTS

This paper has been partially supported by a grant of the University of La Rioja and Banco Santander.

REFERENCES

Ajili, F., Contejean, E., 1995. Complete solving of linear Diophantine equational and inequational systems without adding variables. *Unité de recherche INRIA* Lorraine, Metz.

Bradley, G. H. 1971. Algorithms for Hermite and Smith Normal Matrices and Linear Diophantine Equations. *Mathematics of computation* 25 (116).

Clausen, M., Fortenbacher, A., 1989. Efficient solution of linear Diophantine equations. *Journal of symbolic computation* 8, 201-216.

Contejean, E., Devie, H., 1994. An efficient incremental algorithm for solving systems of linear

Diophantine equations. *Information and computation* 113 (1), 143-172.

Girault, C., Valk, R., 2001. Petri Nets for systems engineering. A guide to modeling, verification and applications. Springer-Verlag, pp. 9-73.

Havas, G., Majewski, B. S., Matthews, K. R., 1998. Extended GCD and Hermite Normal form Algorithms via lattice basis reduction. *Experimental Mathematics* 7 (2), 125-136.

Hemmecke, R., 2011. Discrete optimization. Lecture notes SS 2011, TU Munich.

Jimenez, E., Perez, M., Latorre, J.I., 2006. Industrial applications of Petri nets: system modelling and simulation. *Proceedings of International Mediterranean Modelling Multiconference* 2006, pp. 159-164

Jimenez, E., Perez, M., Latorre, J.I., 2009. Modelling and simulation with discrete and continuous PN: semantics and delays. *Proceedings of 21st European Modeling and Simulation Symposium, Vol II*, pp. 14-19

Jimenez, E., Tejeda, A., Perez, M., Blanco, J., Martinez, E., 2012. Applicability of lean production with VSM to the Rioja wine sector. *International Journal Of Production Research*, 50 (7), 1890–1904

Latorre, J.I., Jimenez, E., Blanco, J., Sáenz-Díez, J.C., 2013a. Integrated methodology for efficient decision support in the Rioja wine production sector. *International Journal of Food Engineering*, (In press).

Latorre, J.I., Jimenez, E., Perez, M., 2009. Decision taking on the production strategy of a manufacturing facility. An integrated methodology. *Proceedings of 21st European Modeling and Simulation Symposium, Vol II*, pp. 1-7.

Latorre, J.I., Jimenez, E., Perez, M., 2013b. Simulation-based Optimisation of Discrete Event Systems by Distributed Computation. *Simulation-Transactions of the Society for Modeling and Simulation International*, (In press).

Latorre, J.I., Jimenez, E., Perez, M., 2013c. The optimization problem based on alternatives aggregation Petri nets as models for industrial discrete event systems. *Simulation-Transactions of the Society for Modeling and Simulation International*, 89 (3), 346–361.

Lazebnik, F., 1996. On systems of linear Diophantine equations. *Mathematics Magazine* 69 (4), 261-266.

Murata, T., 1989. Petri Nets: Properties, analysis and applications. Proceedings of the IEEE 77 (4), pp. 541-580.

Silva, M., 1993. Introducing Petri nets. In *Practice of Petri Nets in Manufacturing*, Di Cesare, F., (editor), pp. 1-62. Ed. Chapman&Hall.