APPLYING AN ADAPTIVE PETRI NET TO CALCULATE THE MAKESPAN IN THE JOB SHOP SCHEDULING PROBLEM

Joselito Medina-Marin[(a)], Juan Carlos Seck-Tuoh-Mora[(b)], Norberto Hernandez-Romero[(c)], Nayeli Jazmin Escamilla-Serna [(d)]

[(a,b,c,d)]Autonomous University of Hidalgo State, Advanced Research Centre in Industrial Engineering, Pachuca de Soto, Hidalgo, México

[(a)]jmedina@uaeh.edu.mx, [(b)]jseck@uaeh.edu.mx, [(c)]nhromero@uaeh.edu.mx, [(d)]nj_esser@hotmail.com

ABSTRACT

The Job Shop Scheduling Problem (JSSP) is one of the typical problems that engineers face on designing Flexible Manufacturing Systems (FMS). In this problem, it is important to find the optimal scheduling to perform a set of tasks in the minimum time. Moreover, the JSSP has some restrictions, such as the tasks order and the number of shared resources where the tasks are carried out. To find the optimal tasks sequence it is necessary to obtain the makespan for each sequence. On this way, FMS can be modeled with Petri nets (PNs), which are a powerful tool that have been used to model and analyze discrete event systems. So, the JSSP can be analyzed in a PN representation of the FMS, and the makespan can be calculated by using the PN model. In this work we propose an adaptive PN to obtain the makespan by applying PN analytical tools.

Keywords: job-shop scheduling problem, makespan, Petri nets, state equation.

1. INTRODUCTION

A Flexible Manufacturing System (FMS) is a discrete event dynamic system that is composed by jobs and shared resources (Zhou and Venkatesh 1999). The typical problem that engineers faced when they are either designing a Flexible Manufacturing System or planning the master production plan for the FMS, is how they should make the best sequence of jobs in the FMS in order to carry all operations out in the minimum time (Pinedo 2012; Lenstra 1977).

This problem is called the Job Shop Scheduling Problem (JSSP), which is a combinatorial problem classified as NP-Complete (Lenstra et al., 1977). There have been published several research papers about finding the minimum value of makespan in the JSSP (Shuai, and Zhi-Hua 2014; Qing-dao-er-ji, and Wang 2012; Zhao, Zhang, and Bing 2011; Ardakan, Hakimian, and Rezvan 2014). The makespan is the time that all the jobs are processed in the FMS, and it depends on the order that all the tasks are performed.

Several exact methods have been analyzed to find the minimum value for the makespan (Dey, Sarkar and Basu 2010; Wang, Cai, and Feng 2010; Wang and Zou 2003). These methods such as branch and bound, linear programming and Langrangian relaxation can find the global minimum value, however for problems with a bigger number of resources and jobs they need a huge amount of computational time to have the final result.

On the other hand, there are also research papers that apply meta-heuristics and/or evolutionary computing to find the minimum makespan time (Qing-dao-er-ji, and Wang 2012; Zhao, Zhang, and Bing 2011). In this case, these proposals can find reasonable results in less time than exact methods. The main drawback of these methods is that the global minimum could not be found, but good approximations are obtained in a short time.

FMSs have been modeled via Petri Nets (PNs) in order to simulate and analyze them. PN theory is adequate to represent in a graphical and mathematical way Discrete Event Systems (DES) such as FMSs, because their dynamic behavior based on event occurrence can be modeled by PN elements (places and transitions) (Murata 1989; Zhou and Venkatesh 1999). Moreover, PN theory offers analytical tools to study the modelled systems, based on the relationship among the FMS resources denoted as PN elements.

One important point in search methods is the calculus of the makespan, taking into account a certain processing order of the tasks. In this paper, we propose the use of an adaptive PN to calculate the makespan by means of the PN state equation.

2. JOB SHOP SCHEDULLING PROBLEM

Scheduling tasks in a FMS is a typical combinatorial problem where it is needed to organize the processing of a set of jobs divided in operations, and each operation is carried out in a shared resource (Gonzalez-Hernandez 2011; Quen-dao-er-ji and Wang 2012).

In the JSSP there are n jobs, and each job consists of m operations, and each operation is processed in a shared resource or machine during a fixed time. Some restrictions should be considered: operations of the same job have a sequence established previously, a job can visit each machine only once, each machine can

process only one job at any time, and there are not restrictions about the precedence among operations of different jobs.

The aim of JSSP is to find a sequence order for operation processing with the minimum value for the makespan.

For instance, Table 1 shows a FMS taken from (Zhang 2010) with some modifications to respect the JSSP restrictions. This example has three machines, four jobs, and each job has three serial operations.

Table 1: FMS configuration with operation times.

Items	Jobs	Operation Serial Number		
		1	2	3
Machine utilization	J_1	M_3	M_1	M_2
	J_2	M_2	M_3	M_1
	J_3	M_3	M_1	M_2
	J_4	M_2	M_1	M_3
Operation time	J_1	$O_{1,1,3}$ *96*	$O_{1,2,1}$ *90*	$O_{1,3,2}$ *35*
	J_2	$O_{2,1,2}$ *74*	$O_{2,2,3}$ *57*	$O_{2,3,1}$ *91*
	J_3	$O_{3,1,3}$ *13*	$O_{3,2,1}$ *5*	$O_{3,3,2}$ *7*
	J_4	$O_{4,1,2}$ *71*	$O_{4,2,1}$ *23*	$O_{4,3,3}$ *38*

Where $O_{i,j,k}$ denotes the *j*-th operation of the *i*-th job to be carried out by the *k*-th machine.

3. PETRI NETS CONCEPTS

A PN is a graphical and mathematical tool that has been used to model concurrent, asynchronous, distributed, parallel, non-deterministic, and/or stochastic systems.

The graph of a PN is directed, with weights in their arcs, and bipartite, whose nodes are of two types: *places* and *transitions*. Graphically, places are depicted as circles and transition as boxes or bars. PN arcs connect places to transitions or transition to places; it is not permissible to connect nodes of the same type. The state of the system is denoted in PN by the use of *tokens*, which are assigned to place nodes.

A formal definition of a PN is presented in table 2 (Murata 1989).

Table 2: Formal definition of a PN

A Petri net is a 5-tuple, $PN = (P, T, F, W, M_0)$ where:
$P = \{p_1, p_2, \ldots, p_m\}$ is a finite set of places,
$T = \{t_1, t_2, \ldots, t_n\}$ is a finite set of transitions,
$F \subseteq \{P \times T\} \cup \{T \times P\}$ is a set of arcs,
$W = F \rightarrow \{1, 2, 3, \ldots\}$ is a weight function,
$M_0 = P \rightarrow \{0, 1, 2, 3, \ldots\}$ is the initial marking,
$P \cap T = \varnothing$ and $P \cup T \neq \varnothing$.

The token movement through the PN represents the dynamical behaviour of the system. In order to change the token position, the following transition firing rule is used (Murata 1989):

1. A transition $t \in T$ is enabled if every input place $p \in P$ of t has $w(p,t)$ tokens or more. $w(p,t)$ is the weight of the arc from p to t.
2. An enabled transition t will fire if the event represented by t takes place.
3. When an enabled transition t fires, $w(p,t)$ tokens are removed from every input place p of t and $w(t,p)$ tokens are added to every output place p of t. $w(t,p)$ is the weight of the arc from t to p.

A Timed Place Petri Nets (TPPN) is an extended PN, where a new element is added. It is a six-tuple $TPPN = \{P, T, F, W, M_0, D\}$, where the first fifth elements are similar to PN definition presented above, and $D = \{d_1, d_2, \ldots, d_m\}$ denotes the time-delay for each place $p_j \in P$ (Zhao, Zhang, and Bing 2011). Output transitions t_i for each p_j will be enabled once the time indicated in p_j is reached.

3.1. Analysis methods

In this paper, we are applying the matrix equation approach as the analytical method of PN theory in order to calculate de makespan of the FMS modelled.

3.1.1. Incidence matrix and state equation

A *PN* with n transitions and m places can be expressed mathematically as an $n \times m$ matrix of integers $A = [a_{ij}]$. The values for each element of the matrix are given by: $a_{ij} = a_{ij}^{+} - a_{ij}^{-}$, where a_{ij}^{+} is the weight of the arc from t_i to p_j, and a_{ij}^{-} is the weight of the arc from p_j to t_i.

The state equation is used to determine the marking of a *PN* after a transition firing, and it can be written as follows:

$$M_k = M_{k-1} \times A^T U_k, \text{ k=1,2,…} \quad (1)$$

where u_k is a $n \times 1$ column vector of n - 1 zeros and one nonzero entries, which represents the transition t_j that will fire. The nonzero entry is located in the position j of u_k. A^T is the transpose of incidence matrix. M_{k-1} is the marking before the firing of t_j. And Mk is the reached marking after the firing of t_j denoted in u_k.

4. ADAPTIVE TIMED PLACE PETRI NET

In this paper we propose an adaptive TPPN (ATPPN), which adds some arcs according to tasks sequence of the FMS.

The formal definition of an ATPNN is as follows: An ATPNN is a seven-tuple $(P, T, F, W, M_0, D, F_d)$, where the first six elements are similar to TPNN elements, and the last one, F_d, is the set of dynamic arcs that change depending on the job operations order. $F_d \subseteq \{P \times T\} \cup \{T \times P\}$. $F \cap F_d = \varnothing$.

4.1. One operation modelling

The ATPNN to model one operation O_{ijk} of a job J_i processed by machine M_k is depicted in figure 1.

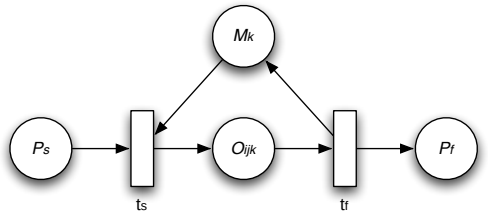

Figure 1: Operation O_{ijk} processed by machine M_k denoted as a PN model.

4.2. One job modelling

As we mentioned above, one job J_i is composed of operations O_{ijk}, so the PN model for each J_i is a concatenation of its operations O_{ijk} (Figure 2).

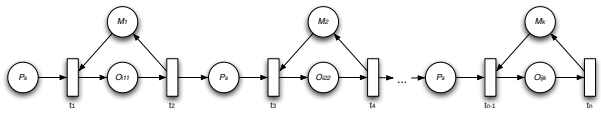
Figure 2: Operations O_{ijk} of job J_i processed by machines M_k denoted as a PN model.

4.3. FMS modelling

In order to model the whole FMS, we add the PN structure for each job J_i and connect every M_k place with its corresponding input (output) transition from (to) operation O_{ijk}. Figure 3 shows the PN model for the FMS described in Table 1.

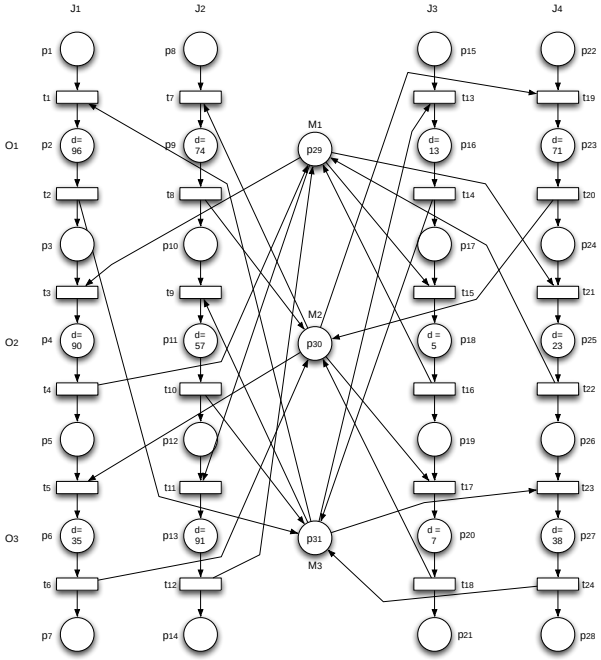

Figure 3: PN model for the FMS described in Table 1.

In Figure 3, each column corresponds to each job J_i, and some places have a label *d*, which denotes the time delay for processing an operation O_{ijk} in the connected machine M_k.

4.3.1. Algorithm to convert a TPPN into an ATPPN

At this time, PN model of figure 3 only represents the FMS, but it is also necessary to set the priority in the operations processing by means of arcs connection in the PN model. So, we need to define the elements of F_d to denote this priority.

First of all, the operations sequence is defined in a row vector $OS = [os_1\ os_2\ \dots\ os_{i\times j}]$, where each *OS* value corresponds to one operation O_{ijk}. The following algorithm receives as inputs the row vector OS and the TPPN model, as output of the algorithm we obtain the ATPPN.

Algorithm `TTPN_into_ATPPN`
Input: `TPPN, OS`
Output: `ATPPN`

```
1. For q=1 to i×j
    k = machineOf(OS(q))
    add(MO(k),OS(q))
End For
2. For k = 1 to NumberOfMachines
    For i = 1 to NumberOfJobs - 1
        p1 = placeOf(MO(k,i))
        p2 = placeOf(MO(k,i+1))
        t1 = p1•
        t2 = •p2
        p3 = •t2
        W(t1, p3) = 1
        W(p3, t2) = 2
    End For
End For
```

In Step 1, a $k\times i$ matrix called M_O is created. Operations $os \in OS$ (O_{ijk}) carried out by the same machine M_k are added in the row k of M_O. The sequence order for the same machine is taking into account.

In Step 2, new arcs $(t,p) \in F_d$ are created, which connect output transitions of places representing operations O_{ijk} with the input place of next operation O_{ijk} in the sequence order defined in M_O. Moreover, a value 2 is assigned to weight $W(p_3,t_2)$, to assure the order in the operations processing.

To illustrate the algorithm result, figure 4 shows the ATPPN obtained, based on the operations denoted in figure 3 and following the order: $OS = [O_1J_2,\ O_1J_4,\ O_2J_2,\ O_1J_3,\ O_2J_4,\ O_1J_1,\ O_3J_2,\ O_3J_3,\ O_2J_1,\ O_3J_1,\ O_2J_3,\ O_3J_3]$.

The ATPPN model presented in figure 4 is used to calculate the makespan for the sequence defined in vector *OS*.

5. ALGORITHM TO OBTAIN THE MAKESPAN

The proposed algorithm takes into account the mathematical representation of the ATPPN. In particular, the incidence matrix and the state equation are utilised to obtain the time delay for each O_{ijk}.

As input data the algorithm needs the ATPPN which includes its input arcs matrix (a_{ij}^-), the output arcs matrix (a_{ij}^+), the time delays column vector *D*, and the initial marking M_0, the total number of jobs (*nj*), the total number of operations per job (*no*), and the total number of shared machines (*nm*).

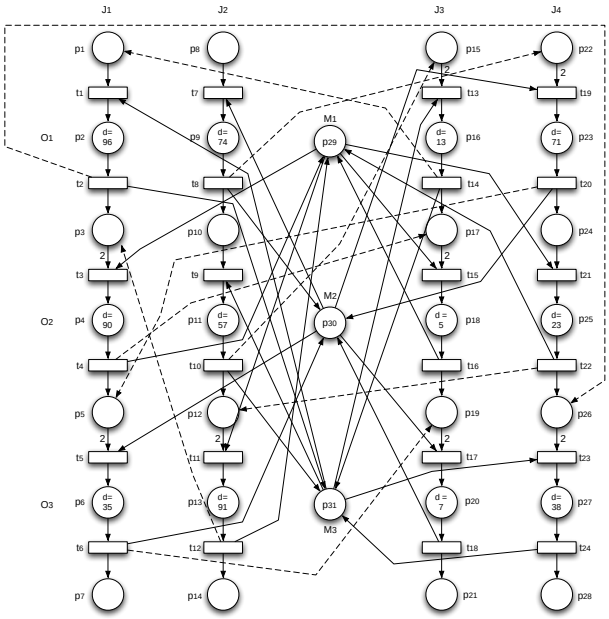

Figure 4: ATPPN model obtained applying the algorithm TTPN_into_ATPPN.

Algorithm Calculate_Makespan

Input: ATPPN, a_{ij}^-, a_{ij}^+, D, M_0, nj, no, nm

Output: *makespan*

1. Initialise variables:
   - txj = |T| / nj
   - pxj = (|P| - nm) / nj
   - AT = [0 0 … 0]' $_{|P| \times 1}$
   - TV = [0 0 … 0] $_{1 \times (nj)}$
2. ET = enabledTransitions(M_0, a_{ij}^-)
3. ∀et ∈ ET, U_k (et) = 1
4. While |ET| > 0
   - For each t ∈ ET
     - indexT = ceil(t/txj)
     - Uk_{tmp}^T = [0 0 … 0]' $_{|T| \times 1}$
     - Uk_{tmp}(t) = 1
     - τ = D' × (a_{ij}^-)' × Uk_{tmp}
     - τ_{accum} = AT' × (a_{ij}^-)' × Uk_{tmp}
     - max_τ_{accum} = max(TV(indexT), τ_{accum}/2) + τ
     - For each p ∈ $t^\bullet$
       - indexP = ceil(p/pxj)
       - If indexT == indexP
         - TV(indexP) = max_τ_{accum}
       - Else
         - AT(p) = max_τ_{accum}
       - End if
     - End For
   - End For
   - M_i = M_{i_1} + (a_{ij})' * Uk
   - ET = enabledTransitions(M_i, a_{ij}^-)
   - Uk^T = [0 0 … 0]' $_{|T| \times 1}$
   - ∀et ∈ ET, Uk(et) = 1
5. End While
6. makespan = max(TV)

In step 1, four variables are initialised: the number of transitions per job (*txj*), the number of places per job (*pxj*), a column vector AT to assign the accumulative time for each place, and a row vector TV utilised to save the time used for each job.

Step 2 obtains the enabled transitions (ET) for an initial marking M_0. Step 3 creates the *Uk* vector from ET transitions.

Step 4 makes an iterative process while the ATTPN is alive, i.e., while there exists at least one enabled transition in the current marking. So, for every enabled transition *t*, we identify the job J_i where the transition belongs (indexT), initialise a temporal *Uk* (Uk_{tmp}) to fire transition t. The time delay τ corresponding to current operation O_{ijk} is calculated multiplying the transpose of the time delay vector D' by the transpose of the input arcs matrix (a_{ij}^-), and the result is multiplied by the firing vector Uk_{tmp} taking into account only transition t.

The accumulated time, denoted as τ_{accum}, represents the time that the needed machine M_k has been busy previous to the current operation O_{ijk}. And it is calculated in a similar way that τ, but in this case we use an auxiliary vector *AT* where the accumulative time for each place is stored, instead of the time delay vector *D*.

Then, we compare both times, the time when the machine *Mk* is ready and the time when the operation O_{ijk} is also ready to be processed. The maximum time plus the time delay for operation O_{ijk} is assigned to max_τ_{accum}. For every $p \in t^\bullet$, if *p* and *t* are in the same job line then max_τ_{accum} is assigned to the time vector variable *TV*. On the other hand, if *p* and *t* belong to different job lines, max_τ_{accum} is assigned to the vector *AT*.

Finally, the ATPPN marking M_i changes according to the result of the equation state. From this new marking M_i, the new enabled transitions are assigned to vector ET and vector *Uk* is generated from them.

6. APPLICATION EXAMPLE

In order to show the applicability of this approach, the FMS configuration showed in Table 1 is taken. This FMS has four jobs, and each job is divided in three operations performed in three different machines.

Operation times showed in Table 3 come from the transposed matrix corresponding to operation times from Table 1.

Table 3: Processing time for each operation per job.

Operation serial number	J_1	J_2	J_3	J_4
1	96	74	13	71
2	90	57	5	23
3	35	91	7	38

Furthermore, to identify every operation in the whole FMS, an operation number was assigned to each one. (Table 4).

Table 3: Number assigned to each operation.

Operation serial number	J_1	J_2	J_3	J_4
1	1	4	7	10
2	2	5	8	11
3	3	6	9	12

Finally, the machine utilization in the FMS is showed in Table 5.

Table 5: Machine utilization.

Operation serial number	J_1	J_2	J_3	J_4
1	3	2	3	2
2	1	3	1	1
3	2	1	2	3

Based on the operation numbers from Table 4, we take the following operation sequence: *OS* = (4 10 5 7 11 1 6 12 2 3 8 9), which is a valid sequence.

From this sequence, the algorithm TPPN_into_ATPPN is executed which has as input data the operation sequence *OS* and the TPPN of figure 3, obtaining as output the ATPPN showed in figure 4. The dashed arcs represent the sequence of the machine operation on the PN model.

Next, the algorithm Calculate_Makespan needs the following data:

1. ATPPN (figure 4)
2. a_{ij}^{+} (figure 4)
3. a_{ij}^{-} (figure 4)
4. $D_{31\times1}$, is a column vector with 31 elements, denoting the processing time in the corresponding place (figure 4). For instance, place p_2 is holding a processing time of 96 time units, so D(2) = 96, which represents the operation time of machine 3 for the operation number 1 (Table 4), and so on.
5. M_0 is the initial marking, where $M_0(i) = 1$, for $i = 1,8,15,22,29,30,31$. And $M_0(j) = 0$ for $j \neq i, 1 \leq j \leq 31$. This marking denotes the starting of FMS operations (p_1, p_4, p_{15}, p_{22}) and the availability of the machines M_1, M_2 and M_3 (p_{29}, p_{30}, p_{31}).
6. $nj = 4$ (number of jobs)
7. $no = 3$ (number of operations per job)
8. $nm = 3$ (number of machines)

Under these conditions, the algorithm Calculate_Makespan starts the transition firing and the token game animation. In order to obtain the processing time for each transition that is fired, the Eq. (2) is computed.

$$\tau = D' \times a_{ij}^{-'} \times u_k \qquad (2)$$

The accumulated time for each job line is saved in vector $TV_{1\times4}$. Table 6 shows the TV values for each transition firing. There are some cases where two transitions were fired simultaneously.

Table 6: Time vector state after transition firings.

Transitions fired	*Time Vector (TV)*			
	J_1	J_2	J_3	J_4
t_7	0	0	0	0
t_8	0	74	0	0
t_9 t_{19}	0	74	0	74
t_{10} t_{20}	0	131	0	145
t_{13} t_{21}	0	131	131	145
t_{14} t_{22}	0	131	144	168
t_1 t_{11}	144	168	144	168
t_2 t_{12}	240	259	144	168
t_3 t_{23}	259	259	144	240
t_4 t_{24}	349	259	144	278
t_5 t_{15}	349	259	349	278
t_6 t_{16}	384	259	354	278
t_{17}	384	259	384	278
t_{18}	384	259	**391**	278

At the end, the maximum value of *TV* (391 time units) represents the makespan for this operation sequence.

The Gantt chart obtained, is showed in figure 5, where the maximum time is 391 time units. The value *#n* over each bar denotes the operation number assigned in Table 4.

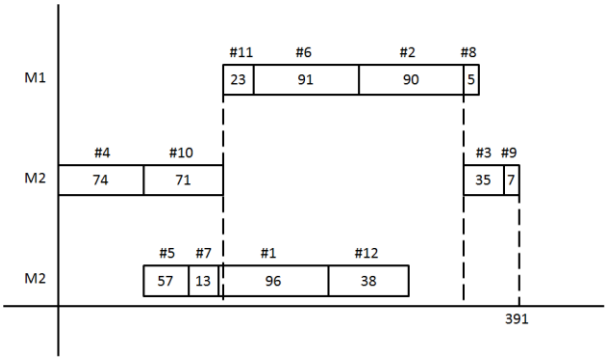

Figure 5: Scheduling chart Gantt for the operation sequence (4 10 5 7 11 1 6 12 2 3 8 9).

The makespan obtained from the scheduling Gantt chart is equal to the makespan obtained from the proposed approach.

7. CONCLUSIONS AND FUTURE WORK

JSSP is a NP-hard problem that has been analysed applying different kinds of techniques, such as exact models and heuristics strategies. One important calculus in the JSSP is the makespan value, which depends on the sequence of operations for each job and the order of machine utilisation.

In this paper, we propose a different way to calculate the makespan by means of mathematical tools

of Petri nets, such as the equation state and the incidence matrix. Firstly, we describe an algorithm to create an ATPPN from a FMS description. The ATPPN arcs indicate the order in which operations O_{ijk} must be done in each job. Moreover, arcs are connected adequately to set the operations order for each machine. And secondly, the marking evolution by using the state equation is taken into advantage to calculate the makespan. We added a time delay vector in order to consider the processing time for every operation involved in the FMS, and it is included in the matrix operations to be able to obtain the makespan for each job.

As future work, we are including these algorithms as part of a study based on evolutionary computing. Moreover, we are interested in analyse the feasibility of PN tools as part of an heuristic to obtain de minimum makespan in the JSSP.

REFERENCES

Ardakan, M. A., Hakimian, A., Rezvan, M. T., 2014. A branch-and-bound algorithm for minimising the number of tardy jobs in a two-machine flow-shop problem with release dates. *International Journal of Computer Integrated Manufacturing*. 27(6): 519-528

Dey, S., Sarkar, D., Basu, A., 2010. A tag machine based performance evaluation method for job-shop schedules. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(7):1028-41.

Gonzalez-Hernandez, M.A., 2011. *Metaheuristics solutions for "Job-Shop Scheduling Problem with sequence-dependent setup times".* PhD Thesis. Universtiy of Oviedo.

Lenstra, J.K., Kan, A.H.G., Brucker, P., 1977. Complexity of machine scheduling problem. *Annals of Discrete Mathematics*, 1: 343 – 362.

Murata, T., 1989. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4), 541 – 580.

Pinedo, M.L., 2012. *Scheduling: Theory, Algorithms, and Systems, Fourth Edition*, New York:Springer.

Qing-dao-er-ji, R., Wang, Y., 2012. A new hybrid genetic algorithm for job shop scheduling problem. *Computers and Operations Research*, 39:2291-2299.

Shuai, J., Zhi-Hua, H., 2014. Path-relinking Tabu search for the multi-objective flexible job shop scheduling problema. *Computers and Operations Research*, 47: 11-26.

Wang, L., Cai, N., Feng, H.Y., 2010. An adaptive setup planning approach for dynamic machine assignments. *IEEE Transactions on Automation Science and Engineering*, 7(1): 2-14.

Wang, S.F., Zou, Y.R., 2003. Techniques for the job shop scheduling problem: a survey. *Systems Engineering – Theory & Practice*, 23: 49-55.

Zhao, Z., Zhang, G., Bing, Z., 2011. Scheduling Optimization for FMS Based on Petri Net Modeling and GA. *Proceedings of the IEEE International Conference on Automation and Logistics*, pp. 422-427. August 2011, Chongqing, China.

Zhou, M.C., and Venkatesh, K., 1999. *Modeling, Simulation, and Control of Flexible Manufacturing Systems*. New York: World Scientific.

AUTHORS BIOGRAPHY

Joselito Medina-Marin. He received the M.S. and Ph.D. degrees in electrical engineering from the Research and Advanced Studies Centre of the National Polytechnic Institute at Mexico, in 2002 and 2005, respectively. Currently, he is a Professor of the Advanced Research in Industrial Engineering Centre at the Autonomous University of Hidalgo State at Pachuca, Hidalgo, México. His current research interests include Petri net theory and its applications, active databases, simulation, and programming languages.

Juan Carlos Seck-Tuoh-Mora. He received the M.S. and Ph.D. degrees in electrical engineering (option: Computing) from the Research and Advanced Studies Centre of the National Polytechnic Institute at Mexico, in 1999 and 2002, respectively. Currently, he is a Professor of the Advanced Research in Industrial Engineering Centre at the Autonomous University of Hidalgo State at Pachuca, Hidalgo, México. His current research interests include cellular automata theory and its applications, evolutionary computing and simulation.

Norberto Hernandez-Romero. He received the M.S. degree from Department of Electrical Engineering, Laguna Technological Institute at México, in 2001 and the Ph. D. from Autonomous University of Hidalgo State at México in 2009. Currently, he is a professor of the Advanced Research in Industrial Engineering Centre at the Autonomous University of Hidalgo State at Pachuca, Hidalgo, México. His current research interests include system identification, feedback control design, genetic algorithms, fuzzy logic, neural network and its applications

Nayeli Jazmin Escamilla-Serna. She is a postgraduate student in Industrial Engineering in the Advanced Research in Industrial Engineering Centre at the Autonomous University of Hidalgo State at Pachuca, Hidalgo, México.