Modeling and Simulation of Flow Shop Scheduling Problem through Petri Net Tools

Joselito Medina Marin, Norberto Hernández Romero, Juan Carlos Seck Tuoh Mora, Erick S. Martinez Gomez

***Abstract*—**The Flow Shop Scheduling Problem (FSSP) is a typical problem that is faced by production planning managers in Flexible Manufacturing Systems (FMS). This problem consists in finding the optimal scheduling to carry out a set of jobs, which are processed in a set of machines or shared resources. Moreover, all the jobs are processed in the same machine sequence. As in all the scheduling problems, the makespan can be obtained by drawing the Gantt chart according to the operations order, among other alternatives. On this way, an FMS presenting the FSSP can be modeled by Petri nets (PNs), which are a powerful tool that has been used to model and analyze discrete event systems. Then, the makespan can be obtained by simulating the PN through the token game animation and incidence matrix. In this work, we present an adaptive PN to obtain the makespan of FSSP by applying PN analytical tools.

***Keywords*—**Flow-shop scheduling problem, makespan, Petri nets, state equation.

I. Introduction

A FMS is a discrete event dynamic system that is composed of jobs and shared resources [1]. FMS engineers The typical problem that engineers faced when they are either designing a FMS 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 [2], [3]. This problem is called FSSP, which is a combinatorial problem classified as NP-hard [4]. 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.

There have been published several research papers about finding the minimum value of makespan in the FSSP. For instance, a D.S. Palmer proposed a method to find an acceptable sequence in less time than exhaustive search [5]. Another algorithm based on heuristic strategies to find suitable solutions was proposed in [6]. Dannenbring performed a similar work, where he proposed eleven heuristics to solve the FSSP [7]. Nawas proposed an algorithm based on the assumption that jobs with higher processing time must be treated first; his algorithm is applied to static and dynamic sequencing environment [8]. In [9], Taillard applied taboo search to solve FSSP; moreover, he implemented a parallel version of taboo search to improve the algorithm execution time. Framinan and Leisten proposed a heuristic taking into account the optimization of partial schedules; instead of optimize the whole schedule [10]. Later, Framinan, Leisten and Ruiz-Usano proposed two multi-objective heuristics, whose objectives to solve are makespan and flowtime minimization [11].

Several metaheuristics have been used to find the minimum value for the makespan, such as Simulated Annealing [12], [13]; Taboo Search [14], [15]; Genetic Algorithms [16]-[18]; Ant Colony Optimization [19]-[20]; Iterated Local Search [21]; and Particle Swarm Optimization Algorithms [22], [23], [27]. These proposals can find reasonable results in less time than exact methods. The main outcome of these methods is that the global minimum could not be found; however, good approximations are obtained in a short time. Thus, all of them need a way to represent the FSSP in order to calculate the makespan. FSSP modeling should be understandable and able to calculate the makespan of a job operations sequence.

FMSs have been modeled via 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) [24]. Moreover, PN theory offers analytical and graphical tools to study the modeled 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 timed PN to calculate the makespan taking into account the PN state equation.

II. FSSP

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 [25], [26].

In the FSSP, given the processing times p_{jk} for each job j on every machine k, and a job sequence $S = (s_1, s_2, \ldots, s_n)$ where n jobs ($j = 1, 2, \ldots, n$) will be processed by m machines ($k = 1, 2, \ldots, m$), so the aim of FSSP is to find a sequence order for operation processing with the minimum value for the makespan. For instance, Table I shows a FMS with three machines, four jobs, and each job has three serial operations.

III. PNs 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.

Joselito Medina is with the Department of Industrial Engineering, Autonomous University of Hidalgo State, Pachuca, Mexico (e-mail: jmedina@uaeh.edu.mx).

TABLE I
FMS CONFIGURATION WITH OPERATION TIMES

Items	Jobs	Operation Serial Number		
		M_1	M_2	M_3
Operation time	J_1	96	90	35
	J_2	74	57	91
	J_3	13	5	7
	J_4	71	23	38

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 as follows [24]: A PN 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 [24]:

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 $p_j \in P$ [28]. Output transitions t_i for each p_j will be enabled once the time indicated in p_j is reached.

A. 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.

B. 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:

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

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.

IV. ADAPTIVE TIMED PLACE PN

In this paper we apply 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. $F_d \subseteq \{P \times T\} \cup \{T \times P\}$. $F \cap F_d = \varnothing$.

A. One Operation Modeling

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

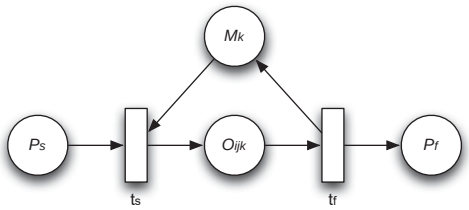

Fig. 1 Operation O_{ijk} processed by machine M_k denoted as a PN model

B. One Job Modeling

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} (Fig. 2).

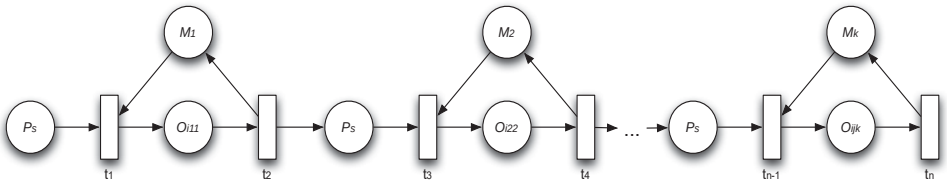

Fig. 2 Operations O_{ijk} of job J_i processed by machines M_k denoted as a PN model

C. FMS Modeling

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}. Fig. 3 shows the PN model for the FMS described in Table I.

In Fig. 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.

D. Algorithm to Convert a TPPN into an ATPPN

At this time, PN model of Fig. 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.

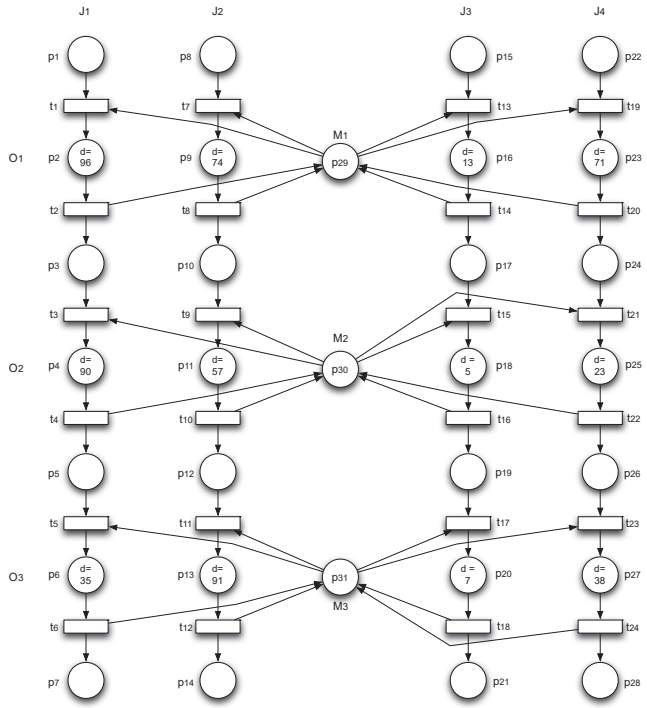

Fig. 3 PN model for the FMS described in Table I

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

1. For q=1 to i×j
    k = machineOf(OS(q))
    add(M0(k),OS(q))
End For
2. For k = 1 to NumberOfMachines
    For i = 1 to NumberOfJobs – 1
        p1 = placeOf(M0(k,i))
        p2 = placeOf(M0(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, Fig. 4 shows the ATPPN obtained, based on the operations denoted in Fig. 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 Fig. 4 is used to calculate the makespan for the sequence defined in vector OS.

V. 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).

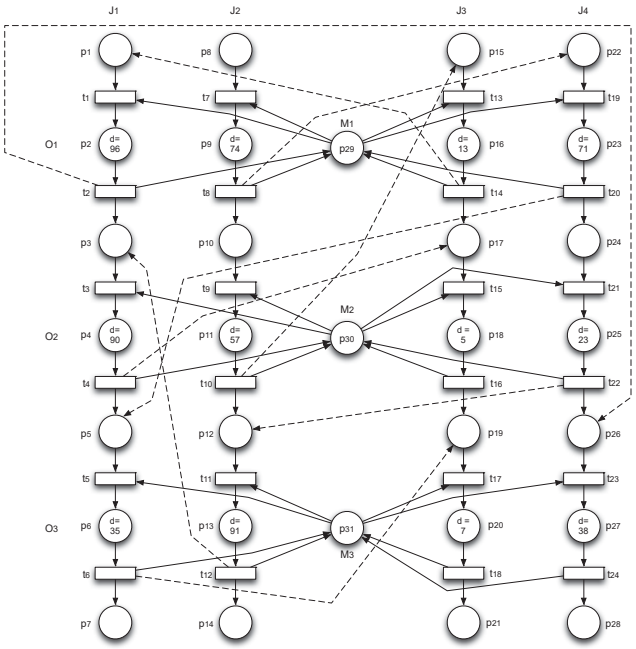

Fig. 4 ATPPN model obtained applying the algorithm TTPN_into_ATPPN

```
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| x 1
    TV = [0 0 … 0]  1 x (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| x 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•
        indexP = ceil(p/pxj)
        If indexT == indexP
          TV(indexP) = τ_accum
        Else
          AT(p) = 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| x 1
   ∀et ∈ ET, Uk(et) = 1
End While
5. 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 time accumulated, 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.

VI. Conclusions and Future Work

FSSP 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 FSSP is the makespan value, which depends on the sequence of operations for each job and the order of machine utilisation.

In this paper, we use a different way to calculate the makespan by means of mathematical tools of PNs, 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 a heuristic to obtain de minimum makespan in the FSSP.

References

[1] M.C. Zhou, and K. Venkatesh, *Modeling, Simulation, and Control of Flexible Manufacturing Systems*. New York: World Scientific, 1999.
[2] M.L. Pinedo, *Scheduling: Theory, Algorithms, and Systems,* Fourth Edition, New York: Springer, 2012.
[3] J.K. Lenstra, A.H.G. Kan, P. Brucker, "Complexity of machine scheduling problem," *Annals of Discrete Mathematics*, vol. 1, pp. 343 – 362.
[4] A.H.G. Rinnooy Kan, *Machine Scheduling Problems: Classification, Complexity and Computations*, Nojhoff, The Hague, 1976.
[5] D.S. Palmer, "Sequencing jobs through a multistage process in the minimum total time: A quick method of obtaining a near-optimum," *Operational Research Quarterly*, vol. 16, pp. 101-107, 1965.
[6] H.G. Campbell, R.A. Dudek, M.L. Smith, "A heuristic algorithm for the n job, m machine sequencing problem," *Management Science,* vol. 16, no. 10, pp. B630-B637, 1970.
[7] D.G. Dannenbring, "An evaluation of flow shop sequencing heuristics," *Management Science*, vol. 23, no. 11, pp. 1174-1182, 1977.
[8] M. Nawaz, E.E. Enscore Jr., I. Ham, "A heuristic algorithm for the m-machine, n-job flow shop sequencing problem," *OMEGA*, vol. 11, no. 1, pp. 91-95, 1983.
[9] E. Taillard, "Some efficient heuristic methods for the flowshop sequencing problems," *European Journal of Operational Research*, vol. 47, pp. 65-74, 1990.
[10] J. M. Framinan, R. Leisten, "An efficient constructive heuristic for flowtime minimisation in permutation flow shops," *OMEGA*, vol. 31, pp. 311-317, 2003.
[11] J.M. Framinan, R. Leisten, R. Ruiz-Usano, "Efficient heuristics for flowshop sequencing with the objectives of makespan and flowtime minimisation," *European Journal of Operational Research*, vol. 141, pp. 559-569, 2002.
[12] 1. Osman, C. Potts, "Simulated annealing for permutation flow shop scheduling," *OMEGA*, vol. 17, no. 6, pp. 551-557, 1989.
[13] F. Ogbu, D. Smith, "The application of the simulated annealing algorithm to the solution of the n/m/Cmax flowshop problem," *Computers and Operations Research*, vol. 17, no. 3, pp. 243-253, 1990.

[14] J. Grabowski, M. Wodecki, "A very fast tabu search algorithm for the permutation flowshop problem with makespan criterion," *Computers and Operations Research*, vol. 31, no. 11, pp. 1891-1909, 2004.
[15] E. Nowicki, C. Smutnicki, "A fast tabu search algorithm for the permutation flowshop problem", *European Journal of Operational Research*, vol. 91, pp. 160-175, 1996.
[16] T. Aldowaisan, A. Allahverdi, "New heuristics for no-wait f1owshops to minimize makespan," *Computers and Operations Research*, vol. 30, no. 8, pp. 1219-1231, 2003.
[17] T. Murata, H. Ishibuchi, H. Tallaka, "Genetic algorithms for f1owshop scheduling problems," *Computers and Industrial Engineering*, vol. 30, no. 4, pp. 1601-1071, 1996.
[18] R. Ruiz, C. Maroto, J. Alcaraz, "Two new robust genetic algorithms for the flowshop scheduling problems," *OMEGA*, vol. 34, pp. 461-476, 2006.
[19] C. Rajendran, H. Ziegler, "Ant-colony algorithms for permutation f1owshop scheduling to minimize makespan/total flowtime of jobs," *European Journal of Operational Research*, vol. 155, no. 2, pp. 426-438, 2004.
[20] T. Stutzle, "An ant approach to the f1owshop problem," In: *Proceedings of the 6th European Congress on Intelligent Techniques and Soft Cmputing (EUFIT'98)*, Verlag Mainz, Aachen, Germany, pp. 1560-1564, 1998.
[21] T. Stutzle, "Applying iterated local search to the permutation f1owshop problem," *Technical Report, AIDA-98-04*, Darmstad University of Technology, Computer Science Department, Intellctics Group, Darmstad, Germany, 1998.
[22] M.F. Tasgetiren, M. Sevkli, Y.C. Liang, and G. Gencyilmaz, "Particle swarm optimization algorithm for permutation flowshop sequencing problem," In: *Proceedings of the 4th International Workshop on Ant Colony Optimization and Swarm Intelligence (ANTS2004)*, *LNCS* 3172, Brussels, Belgium, pp. 382-390, 2004.
[23] M.F. Tasgetiren, Y.C. Liang, M. Sevkli, G. Gencyilmaz, "Particle swarm optimization algorithm for makespan and total f1owtime minimization in the permutation f1owshop sequencing problem," *European Journal of Operational Research*, 2006.
[24] T. Murata, "Petri Nets: Properties, Analysis and Applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541 – 580, 1989.
[25] M.A. Gonzalez-Hernandez, "Metaheuristics solutions for Job-Shop Scheduling Problem with sequence-dependent setup times," *PhD Thesis*. Universtiy of Oviedo, 2011.
[26] R. Qing-dao-er-ji, Wang, Y. "A new hybrid genetic algorithm for job shop scheduling problem." *Computers and Operations Research*, vol. 39, pp. 2291-2299, 2012.
[27] Q.K. Pan, M.F. Tasgetiren, Y.C. Liang, "A Discrete Particle Swarm Optimization Algorithm for the Permutation Flowshop Sequencing Problem with Makespan Criterion," *Research and Development in Intelligent Systems XXIII*, Springer London, pp. 19-31, 2007.
[28] Z. Zhao, G. Zhang, Z. Bing, "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.

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.

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.

Juan C. 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.

Erick S. Martinez-Gomez He 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.