DOI: http://dx.doi.org/10.18180/tecciencia.2017.23.5

TECCIENCIA

Programming of Job Shop Production Systems with Fuzzy Logic

Programación de Sistemas de Producción Job Shop con Lógica Difusa

Luis Eduardo Leguizamón Castellanos[1]*

[1]*Servicio Nacional de Aprendizaje, SENA, Colombia*

Received: 12 Feb 2018 *Accepted: 8 Jul 2018* *Available Online: 29 Aug 2018*

Abstract

This article proposes a decision-making algorithm based on the "fuzzy logic" as an optimization technique, which allows to find a good solution to the problem of determining the priority (sequencing) of service or manufacture of jobs in the programming of intermittent production systems, Job Shop. The combinatorial nature and complexity of the problem motivates the exploration of other alternatives solutions to the traditionally used. Initially, the fuzzy logic controller structure (number of input variables, rules and output) is determined in accordance with the objective functions to be optimized. Triangular membership functions are selected for the batch size, the delivery date, the processing time, the number of tools required in each operation, and the priority of the processing the jobs. The fuzzy rules base is defined, and the controller model is formulated (fuzzification, evaluation and defuzzification). The algorithm is developed in Matlab's ®Simulink ®Fuzzy logic toolbox, achieving better results than those obtained with other methods.

Keywords: Fuzzy Logic, Sequencing, Job Shop, Membership Function.

Resumen

Este artículo propone un algoritmo de toma de decisiones con base en la lógica difusa (fuzzy logic) como técnica de optimización, que permita encontrar una buena solución al problema de determinar la prioridad (secuenciación) de servicio o fabricación de trabajos en la programación de sistemas de producción intermitente (Job Shop). La naturaleza combinatoria y complejidad del problema motiva la exploración de otras alternativas de solución a las tradicionalmente usadas. Inicialmente se determina la estructura del controlador difuso (número de variables de entrada, reglas y salida) en concordancia con las funciones objetivo a optimizar. Se seleccionan funciones de membresía triangulares para el tamaño de lote, la fecha de entrega, el tiempo de procesamiento, el número de herramientas necesarias en cada operación y la prioridad de procesamiento de los trabajos. Se define la base de reglas difusas y se formula el modelo del controlador (fuzzificación, evaluación y defuzzificación). El algoritmo es desarrollado en ®Simulink y ®Fuzzy logic toolbox de Matlab, alcanzando mejores resultados que los obtenidos con otros métodos.

***Palabras clave*:** Lógica Difusa, Secuenciación, Sistemas de Producción Intermitente, Función de Membresía.

1. Introduction

Programming production involves assigning workloads, sequencing jobs and timing them. Of these activities, sequencing is perhaps the most studied and is defined as the priority of service or processing of the jobs that are in the waiting line of the production system [1].

This activity is possibly the most complex of the process of programming operations due to its nature of combinatorial problem, especially if it is about intermittent production systems (job shop), which are characterized by manufacturing a wide variety of products in batches (batch), through several operations in different routes and processing times.

***Corresponding Author.**
E-mail: *leleguizamon@misena.edu.co*

How to cite: Leguizamon, L.E., ***Programming of Job Shop Production System with Fuzzy Logic*** **,** TECCIENCIA, Vol. 13 No. 25, 39-45**,** 2018
DOI: http://dx.doi.org/10.18180/tecciencia.2017.23.5

One of the first to study and propose a method of heuristic solution to this problem was Jackson [2]. Since then, the different methodologies that have been designed to solve the problem have been classified as heuristics and exact methods [3], see Table 1.

Exact methods can only be used in the solution of small problems due to the large number of variables that must be defined, and the excessive computational time required in their solution. The most common approach to the intermittent production is to use dispatch rules with priorities [3].

Most methods focus on models that satisfy only a single goal, but in the real world, usually, more than one of them most be satisfied simultaneously [4]. The most efficient programming systems must be able to achieve objectives concurrently such as, minimize the time of completion of all jobs (makespan), maximize the use of the machines, minimize costs of enlistment and preparation of parts and machines, maintain the balance of the workloads and meet the delivery deadline among other things.

2. Materials and methods

The problem consists in determining the sequence of operation of n jobs on m machines, to optimize a certain measure of performance that measures the efficiency of the program. The manufacture of each job requires the execution, in a certain established order, of a series of prefixed operations where each operation is assigned to one of the m machines and has a determined and known processing time. Problems such as that described above originate in the most diverse sectors of goods and/or services production.

The problem can be solved by total enumeration. However, since there are $(n!)^m$ possible sequences of operation and several performance measures or objective functions to optimize, is classified as a NP – hard problem, due to there is no algorithm that solves it in a reasonable computational time [2].

For 15 jobs on 6 machines there are $(15!)^6$ possible sequences to be executed, here the question would be, which is the best? The answer to this question motivates the application of new solution alternatives and depends of the objective functions to optimize and the system variables. The parameters and variables of the model are observed in Table 2.

Given the difficulty of maximizing profits or minimizing costs in a programming process, termination time functions are used, where the goal is to minimize these functions [2]. Some of these are shown in Table 3.

Table 1 Solution Methods.

EXACT METHODS		HEURISTICS METHODS		
LINEAR PROGRAMMING	DYNAMIC PROGRAMMING	RULES OF DECISIÓN	HEURISTICS	METAHEURISTICS
*INTEGER *BINARY *MIXED		*FCFS (First Come, First Served) *SPT (Shortest Process Time) *EDD (Earliest Due Date)	*Palmer *CDS *Gupta *Dannenbring *Hundal *Ho and Chang	*Search in the neighborhood *Simulated annealing *Search tabu *Genetics algorithms *Swarm of particles

Table 2 Parameters and variables - job shop.

PARAMETERS AND VARIABLES

P_i = Job, order or part *i*, ∀ i=1, 2,..., n.

M_k = Machine *k*, ∀ k=1, 2,..., m.

O_j = Operation *j*, ∀ j=1, 2,..., p.

n = Jobs number.

m = Machines number.

p = Operations number.

d_i = Due date job *i*, ∀ i=1, 2,..., n.

g_i = Operations number of job *i*, ∀ i=1, 2,..., n.

t_{ji} = Processing time operation j in job *i*.

w_{ji} = Wait time operation j in job *i*.

c_i = Completion time of the job *i*.

The objective of this work is to find a production program by applying fuzzy logic for the case proposed in figure 1 [5], which maximizes the use of the machines and at the same time minimize the makespan, the maximum wait time, the maximum advance, the maximum delay and the product in process.

The variables to be considered in the problem are: the process time, the committed delivery deadline, the batch size, and the number of tools required in each operation, as shown in Table 4.

3. Proposed methodology

The information provided by the input variables is imprecise [6] because of its ambiguity, since they are bounded in intervals, for this reason we are in the presence of a deterministic uncertainty. Due to this, the method that is proposed involves the application of a Fuzzy Inference System (FIS), as an element of decision making. See flow diagram of Figure 2.

Table 3 Time functions - job shop.

TIME FUNCTIONS	
t_i = Processing time of job *i.*	
	$t_i = \sum_{j=i}^{g_i} t_{ji}$
w_i = Wait time of job *i.*	
	$w_i = \sum_{j=i}^{g_i} w_{ji}$
L_i = Lateness of job *i.*	$L_i = c_i - d_i$
T_i = Tardiness of job *i.*	$T_i = \max\{0 , L_i\}$
E_i = Earliness of job *i.*	$E_i = \max\{0 , -L_i\}$
Makespan	$C_{max} = \max\{c_i\}$
Maximum Tardiness	$T_{max} = \max\{T_i\}$
Minimum number of late job	Min $\sum_{i=1}^{n} T_i$
Minimize makespan	Min C_{max}
Minimize maximum tardiness	Min T_{max}
Minimize number of late job	Min $\sum_{i=1}^{n} T_i$

The *FIS* is comprised of the 4 input variables, a set of 81 rules of the *mamdani* type with the form *IF - THEN* (If - Then) and one output that represents the priority of the workpiece to be processed, as shown in Figure 3.

3.1 Inputs of the fuzzy logic controller

The 4 input variables that affect the problem have each one of them with a universe of discourse divided into 3 fuzzy sets, with small, medium and large linguistic variables [7] respectively, which represent the application of judgment and experience of the programmer when making a decision about the determination of the priority of the jobs.

The membership functions used are of the triangular type and right and left saturation, since they have been the ones that have shown the best behavior in similar studies [8] to this work. Figure 4 shows the fuzzy sets and the membership functions for the batch size.

Table 4 Variables of the proposed case.

PART	OPERATION	PROCESSING TIME	MACHINE	BATCH SIZE	NUMBER OF TOOLS	DUE DATE
1	1	18	M6	8	1	698
	2	25	M1		1	
2	1	24	M6	9	1	2822
	2	22	M2		1	
	3	26	M4		2	
3	1	11	M6	13	3	2202
	2	14	M1		1	
	3	19	M5		1	
	4	22	M2		2	
4	1	25	M5	9	1	2083
	2	16	M4		1	
	3	7	M3		1	
	4	21	M1		1	
	5	19	M2		1	
5	1	13	M2	12	1	3087
	2	23	M4		3	
	3	25	M1		1	
	4	7	M6		1	
	5	24	M5		3	
6	1	18	M5	9	1	2822
	2	25	M1		1	
	3	24	M4		1	
	4	22	M2		1	
7	1	26	M4	13	2	2202
	2	11	M5		3	
	3	14	M3		1	
	4	19	M6		1	
	5	22	M2		2	
8	1	25	M6	10	1	2156
	2	16	M5		1	
	3	7	M2		1	
9	1	21	M2	7	1	2089
	2	18	M3		2	
10	1	23	M6	9	1	2163
	2	17	M3		2	
11	1	18	M1	12	1	2265
	2	25	M3		2	
	3	21	M5		1	
12	1	23	M5	8	2	1975
	2	14	M1		1	
13	1	25	M5	10	1	2015
	2	18	M3		2	
14	1	28	M4	9	1	2114
15	1	22	M3	14	1	1875
	2	19	M4		1	

3.2 Output of the fuzzy logic controller

The output obtained corresponds to a priority value in percentage, which, ordered from largest to smallest, determines the processing sequence of the pieces. The universe of discourse is between 0 and 1 and is represented by nine fuzzy sets, seven of them are represented by triangular membership functions, and two by saturation functions (left and right) as can be seen in Figure 5.

3.3 Fuzzy Rules

The fuzzy rules also known as a blurry rules, combine input fuzzy sets or premises through logical conjuncts (and, or) that are associated with an output fuzzy set or consequence. These rules allow us to express the acquired knowledge about the relationship between antecedents and consequences of the problem under study [9].

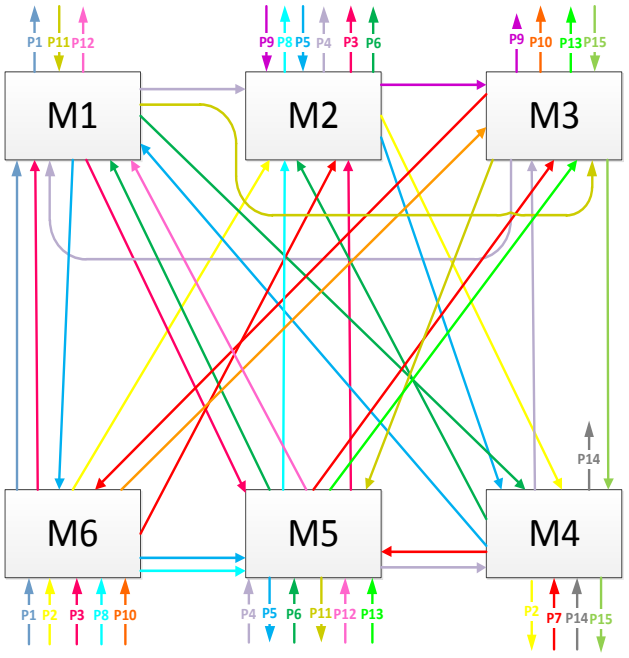

Figure 1 Production System *job shop*.

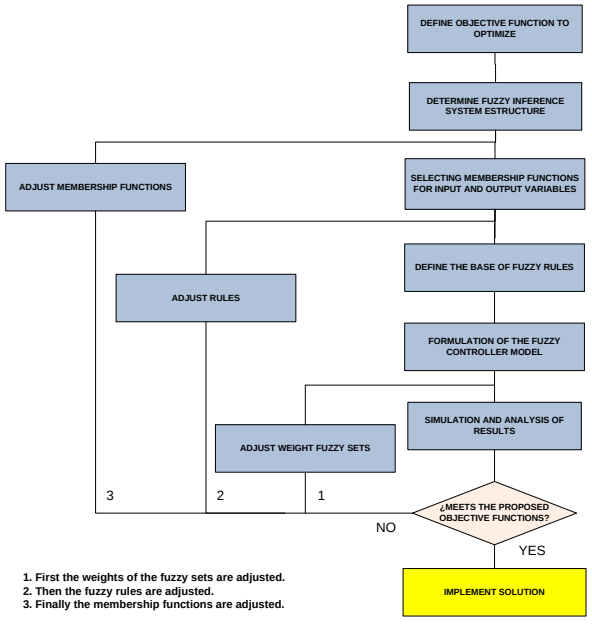

Figure 2 Proposed method flow diagram.

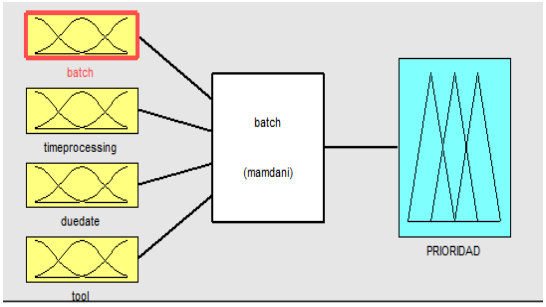

Figure 3 Structure of the *FIS*.

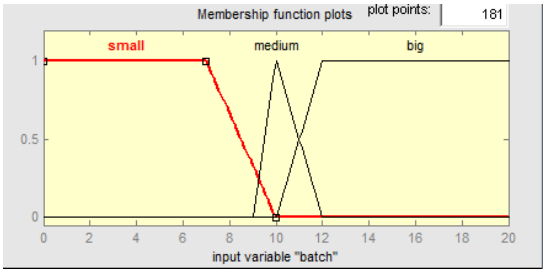

Figure 4 Fuzzy sets of the batch size.

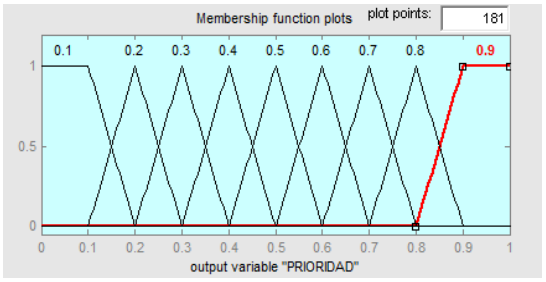

Figure 5 Fuzzy sets of the priority.

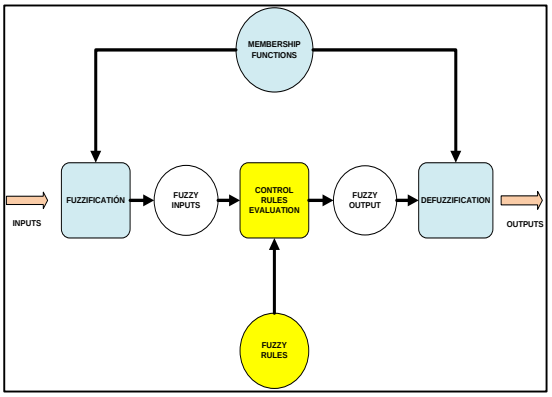

Figure 6 FIS Functioning [10].

Table 5 Pseudo-code algorithm proposed.

STEP	DESCRIPTION
1	Get : x1, x2, x3 y x4
2	Calculate values for membership functions (mf) mf1(i), mf2(j), mf3(k), mf4(l), ∀ i, j, k, l.
3	Calculate values for premises of (mf) prem(i, j, k, l) = min (min (min (mf1(i), mf2(j)), mf3(k)), mf4(l))
4	Find areas under (fm) for all possible implicit fuzzy sets imps(i, j, k, l) = areaimp(rule(i, j, k, l), prem(i, j, k, l))
5	Initialize numerator and denominator of the centroid num = 0, den = 0
6	Cycle to travel through all the areas to find the centroid For i =0.1 to 0.9, For j =0.1 to 0.9, For k =0.1 to 0.9, For l =0.1 to 0.9,
	Calculate centroid numerator num = num + imps(i, j, k, l) * center(rule(i, j, k, l))
	Calculate centroid denominator den = den + imps(i, j, k, l)
7	Next i, Next j, Next k, Next l
8	Output value calculated by the FIS Output $y = \frac{num}{den}$
9	Go to step 1.

The rules are the result of applying the decisions commonly made by the programmers of these production systems based on their experience [5]. Two of the eighty-one ("A" ˜^"x " "=" "3" ^"4") rules designed in the solution of this problem are described below.

- IF batch size is small and processing time is small and the delivery date is small, and the number of tools is small THEN priority is 0.9.

- IF batch size is small and processing time is large and delivery date is large and the number of tools is large THEN priority is 0.6.

3.4 Operation of the FIS

The fuzzification stage of the values of the input variables for each job consists of obtaining the fuzzy variables or belonging degrees of the input value x of the variable to each of the fuzzy sets "μ" _("A" _"i") "(x)" . See figure 6.

The second stage is the evaluation of the control rules, which consists of determining the rules that are activated in the face of a certain input value. Each rule has an associated weight i, that is calculated with the equation (1). This weight sets the belonging degree of the diffuse output variable in the fuzzy sets"μ" _("E" _"i") "(y)" .

$$\min\left[\min\left[\min\left[\mu_{Ai}(x_1),\mu_{Bi}(x_2)\right],\mu_{Ci}(x_3)\right],\mu_{Di}(x_4)\right] \quad (1)$$

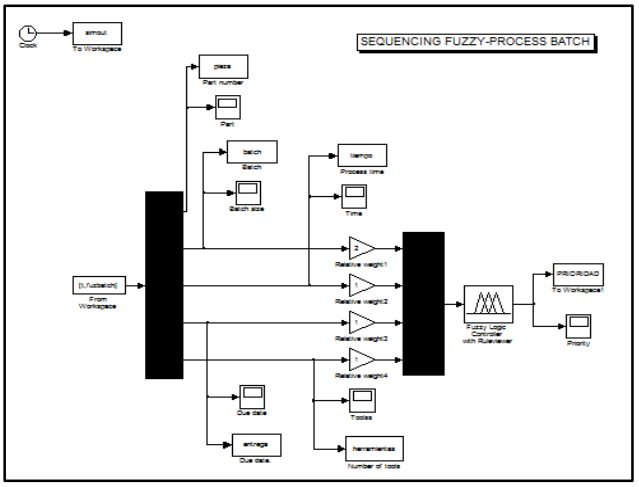

Figure 7 Block diagram - proposed method.

Finally, the defuzzification consists in finding a numerical value for the output from the fuzzy sets that compose it. Here the centroid method is used which consists in creating for the output (y) a membership "μ(y) " function to a new set resulting from the union of fuzzy sets to which the output value belongs partially [10]. These are obtained from the equation (2).

$$y = \frac{\sum_{k=1}^{n} y_k . \Delta_k(y)}{\sum_{k=1}^{n} \Delta_k(y)} \quad (2)$$

Where:

Where:
y = Centroid.
y_k = Centroid of the membership function on the output set k.
$\Delta_k(y)$=Subarea de k where is k.

For the implementation of the algorithm the pseudo-code shown in table 5 is developed, this provides a general description of the steps of how the fuzzy logic controller calculates the output (y) from the inputs (x1, x2, x3, x4).

3.5 Simulation and software

For the simulation of the FIS, the fuzbatch.mat file was created in ®Matlab with the information of the input variables and the fuzbatch.m file that executes the program of the operating instructions of the system.

With ®Fuzzy logic toolbox[11], the membership functions of membership for the inputs and the output were designed, as well as the control rules. Finally, the fuzbatch.cdm block diagram was developed in ®Simulink shown in figure 7, which allows to simulate and obtain the process priority of the pieces.

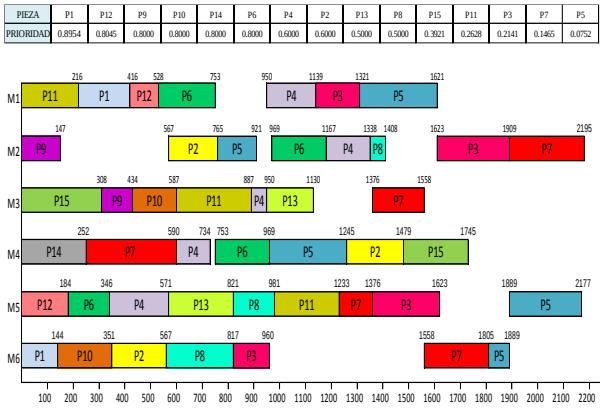

Figure 8 Fuzzy logic program with equal weight of the input variables.

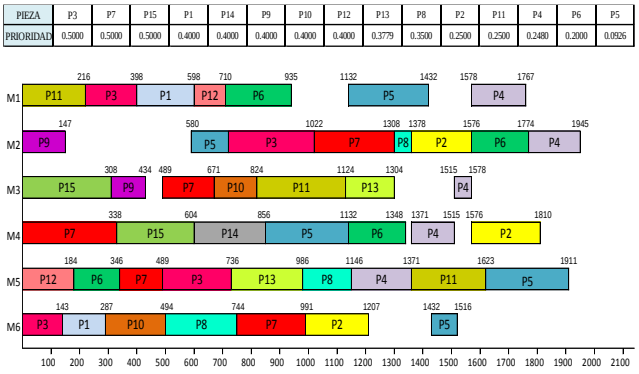

Figure 9 Fuzzy logic program with greater weight of the batch size.

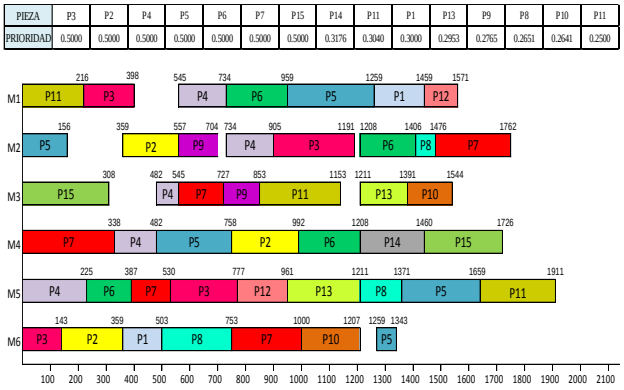

Figure 4 Optimal program with fuzzy logic.

Table 6 Comparison of results.

PRIMARY RULE	BREAKING RULE	MAKESPAN	MAXIMUM WAITING TIME	MAXIMUM EARLINESS	MAXIMUM TARDINESS	WORK IN PROCESS	MEAN MACHINE UTILIZATION
SPT	EDD	2398	1294	1862	12	7,64	0,6378
EDD	BATCH	2262	1461	1862	0	7,55	0,6761
BATCH	SPT	2377	1519	1862	175	7,49	0,6434
HTA	EDD	2494	1636	1862	292	7,12	0,6132
FUZZY LOGIC		2195	1171	1860	0	8,20	0,6967
FUZZY LOGIC-BATCH		1945	1162	1650	0	8,87	0,7863
OPTIMUM		1911	1263	1491	749	11,30	0,8003
FUZZY LOGIC-OPTIMUM		1911	1184	1830	761	11,22	0,8003

Table 7 Comparison of results with inputs of equal relative weight with respect to the dispatch rules.

PRIMARY RULE	BREAKING RULE	MAKESPAN	MAXIMUM WAITING TIME	MAXIMUM EARLINESS	MAXIMUM TARDINESS	WORK IN PROCES	MEAN MACHINE UTILIZATION
SPT EDD BATCH HTA	EDD BATCH SPT EDD	2262	1294	1862	0	7,12	0,6761
FUZZY LOGIC		2,96%	9,51%	0,11%	0,00%	15,17%	2,06%
FUZZY LOGIC-BATCH		14,01%	10,20%	11,39%	0,00%	24,58%	11,02%
OPTIMUM		15,52%	2,40%	19,92%	100,00%	58,71%	12,42%
FUZZY LOGIC-OPTIMUM		15,52%	8,50%	1,72%	100,00%	57,58%	12,42%

4. Results and Discussion

The results obtained by running the program and the simulation, indicates that the pieces with highest priority value should be processed first.

Initially there is the case for the fuzzy sets of the inputs with equal relative weight. Here, the piece that must be processed first is 1, then 12, 9,..., until finish with the work 5, as seen in figure 8, achieving a makespan of 2195 minutes. In table 6 shows that this program reaches better results than those obtained with the application of dispatch rules in five of the six objective functions to be optimized.

If the relative weight is increased for the batch size, the makespan is reduced to 1945 minutes. In this case the piece that must be processed first is 3, then 7, 15,..., until finishing with piece 5. See Figure 9.

Finally, the optimal solution is obtained by simultaneously increasing the relative weight for the batch size and the processing time. Here piece 3 must be processed first, followed by 2, 4,..., until finishing with piece 11, achieving a makespan of 1911 minutes. See Figure 10.

The results obtained in the three simulations are condensed in Table 6. It shows that the objective functions initially proposed are satisfied, and the optimal solution is achieved that minimizes the total flow time.

For the solution with inputs of equal relative weight with respect to the dispatch rules, the completion time is improved by 2.96%, the maximum wait time by 9.51%, the maximum advance by 0.11% and the use of the machines by 2.06%, with no backlog, see Table 7.

When the relative weight of the batch size is increased, improves the completion time by 14.01%, the maximum wait time by 10.20%, the maximum advance by 11.39% and the use of machinery in 11.02%, without any delay. It is then evident, that the relative weights of the inputs (batch size and processing time) directly influence the priority of the processing jobs in the *job shop* production system.

Acknowledgments

I would like to thank to the SENA Industrial Management Center and the ECCI University for their invitation as a speaker at the III Congress of Industrial Production Management and the VII Congress of Resource Management in Production Plants.

References

[1] L. E. Leguizamón Castellanos, «Programación de las actividades logísticas con algoritmos genéticos,» Bogotá DC, 2013.
[2] R. L. J. Daniel Sipper, Planeación y control de la producción, México, D.F.: McGraw-Hill, 1998.
[3] E. Alexander Alberto Correa Espinal, «Secuenciación de operaciones para configuraciones de planta tipo flexible Job Shop: Estado del arte ,» Revista Avances en Sistemas e Informática., vol. 5, nº 3, p. 11, 04 Febrero 2008.
[4] M. L. Pinedo, Scheduling Theory, Algorithms, and Systems, New York: Springer, 2012.
[5] O. A. y. S. K. O. Bilkay, «job shop scheduling using fuzzy logic,» The International Journal of Advanced Manufacturing Technology, vol. 23, pp. 606-619, 2004.
[6] M. J. Redondo, La logica Fuzzy y su aplicacion en la limitación de recursos, Valencia: Universidad Politecnica de Valencia, 2013.
[7] P. E. Onwuachu Uzochukwu C, «Application of Fuzzy Logic to Predictive Job Shop Scheduling in an Interconnected System,» International Journal of Computer Applications, vol. 145, nº 3, pp. 19-24, 2016.
[8] S. Y. Kevin M. Passino, Fuzzy Control, California: Addison Wesley Longman, 1998.
[9] A. S. M. Bonifacio Martin del Brio, Redes Neuronales y Sistemas Difusos, México, D.F: Alfaomega Grupo Editor, S.A, 2002.
[10] V. J. M. José R. Hilera, Redes Neuronales Artificiales., Santafé de Bogotá: Alfaomega, S.A, 2000.
[11] The Math Works, Inc, Fuzzy Logic Toolbox User´s Guide, Natick: 3 Apple Hill Drive, 2017.