Received: 19 Apr 2020 | Accepted: 21 Jun 2021 | Available online: 30 Ago 2021
DOI: http://dx.doi.org/10.18180/tecciencia.2021.31.1

TECCIENCIA

UNIVERSIDAD · ECCI

RESEARCH ARTICLE
Engineering

Programación de Sistemas de Producción Flow Shop con Optimización de Enjambre de Partículas

Scheduling of Flow Shop Production Systems with Particle Swarm Optimization

L.E. Leguizamón[1*] | L.A. Leguizamón[2†] | C.E. Leguizamón[3‡]

[1]Ingeniero Industrial, Esp. en Gerencia de Mercadeo y Automatización Industrial, M.Sc. Ingeniería de Control Industrial – leleguizamon@misena.edu.co – SENA – Centro de Gestión Industrial – Tecnología en Gestión de la Producción Industrial

[2]Ingeniero Electrónico y de Telecomunicaciones, MBA, DBA(c) – a20147494@pucp.pe – Pontificia Universidad Católica del Perú.

[3]TE, Ingeniero industrial, M.Sc.(c) Logística Aeronáutica – cristhian.leguizamon@esufa.edu.co – Fuerza Aérea Colombiana.

Correspondence
L.E. Leguizamón
Email: leleguizamon@misena.edu.co

Editors: Robert Paul Salazar

How to cite: L.E. Leguizamón, L.A. Leguizamón & C.E. Leguizamón, **Scheduling of Flow Shop Production Systems with Particle Swarm Optimization**, TECCIENCIA, Vol. 16, No. 31, 1-13, 2021
DOI:http://dx.doi.org/10.18180/tecciencia.2021.31.1

Abstract. This article proposes a decision-making algorithm as an optimization technique based on a particle swarm (particle swarm optimization-PSO), which allows finding a good solution to the problem of determining the priority (sequencing) of service or manufacturing of jobs scheduling of Flow Shop production systems. The combinatorial nature and complexity of the problem motivates the exploration of other alternative solutions to those traditionally used. Previously defined the objective functions to optimize and the size of the swarm, the position and velocity of the particles are initialized. The objective function is then calculated and the best individual and global positions in the swarm are determined. Finally, speed and position are updated, repeating this procedure according to the number of iterations proposed. The algorithm is developed in Microsoft® Excel® and ®Matlab, achieving better results than those obtained with other methods, such as the service factor, which increases by 19.9% for one machine and 20% for two machines.

Keywords: Particle swarm, sequencing, flow shop, objective function.

Resumen. Este artículo propone un algoritmo de toma de decisiones como técnica de optimización con base en enjambre de partículas (particle swarm optimizatión-PSO), 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 Flow Shop. La naturaleza combinatoria y complejidad del problema motiva explorar otras alternativas de solución a las tradicionalmente usadas. Previamente definidas las funciones objetivo a optimizar y el tamaño del enjambre, se inicializan la posición y velocidad de las partículas. Luego se calculan la función objetivo y se determinan las mejores posiciones individuales y globales en el enjambre. Finalmente se actualizan velocidad y posición, repitiendo este procedimiento según el número de iteraciones propuesto. El algoritmo es desarrollado en Microsoft® Excel® y ®Matlab, alcanzando mejores resultados que los obtenidos con otros métodos, como es el caso del factor servicio, que aumenta en 19.9% para n trabajos en una máquina y en 20% para dos máquinas.

Palabras clave: Enjambre de partículas, secuenciación, flow shop, función objetivo.

1 | INTRODUCCIÓN

Programar la producción implica asignar cargas de trabajo, secuenciar trabajos y temporizarlos. De estas actividades la más estudiada es la secuenciación que consiste en determinar la prioridad de servicio o procesamiento de los trabajos que se encuentran en la línea de espera del sistema de producción. Esta actividad es

*Equally contributing authors.

considerada la más compleja del proceso de programación de operaciones debido a su naturaleza de problema combinatorio, como es el caso de los sistemas de producción secuenciales flow shop [1], que se caracterizan por que los productos pasan a través de todos los procesos (máquinas) en el mismo orden, es decir que tienen una relación de procesos y secuencias idénticas.

Johnson fue uno de los primeros en estudiar y plantear un algoritmo de solución óptima para n trabajos en 2 máquinas [2][3]. Desde entonces las diferentes metodologías propuestas para resolver el problema se han clasificado como métodos exactos y heurísticos, ver Tabla 1, [4].

TABLA 1 Métodos de solución.

EXACT METHODS		HEURISTICS METHODS		
LINEAR PROGRAMMING		**RULES OF DECISIÓN**	**HEURISTICS**	**METAHEURISTICS**
*INTEGER *BINARY *MIXED	**DYNAMIC PROGRAMMING**	*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

Los métodos exactos solo pueden ser usados en la solución de pequeños problemas debido a la gran cantidad de variables que se deben definir y al excesivo tiempo computacional que requiere su solución [4]. De los métodos heurísticos, las reglas de decisión y las heurísticas son tradicionalmente los más usados, por lo general son capaces de buscar y garantizar un óptimo local [5].

Las metaheuristicas imitan simples fenómenos que son observados en la naturaleza al parecer asociados con la inteligencia artificial (IA). Estos algoritmos tratan de adaptar el comportamiento de algunas especies a soluciones de problemas altamente complejos, mediante mecanismos específicos que permiten alcanzar un óptimo global, pero no garantizan la solución óptima del problema [6].

La mayoría de métodos se centran en modelos que satisfacen solamente un único objetivo, pero en el mundo real, por lo general hay que satisfacer en forma simultánea más de uno de ellos [3]. Los sistemas de programación más eficientes deben ser capaces de alcanzar en forma paralela objetivos como, reducir al mínimo el atraso, el adelanto, el producto en proceso y el tiempo de terminación de todos los trabajos (makespan), maximizar la utilización de máquinas y cumplir con la fecha comprometida de entrega entre otras más [4]. El aporte que se obtiene al aplicar enjambre de partículas a este problema es lograr optimizar en forma simultanea varias de las funciones objetivo antes mencionadas.

2 | DESCRIPCIÓN DEL PROBLEMA

El problema consiste en establecer la secuencia de operación de n trabajos en m máquinas, con el objeto de optimizar una función de desempeño que mida la eficiencia del programa. Para su fabricación cada trabajo requiere la ejecución en cierto orden de operaciones previamente definidas y asignadas a una de las m máquinas con un tiempo de proceso estándar y conocido. Este tipo de problemas se originan en los más diversos sectores de la producción de bienes y/o servicios [7].

El problema puede ser resuelto por enumeración total. Sin embargo al existir "n!" secuencias posibles de fabricación y diversas funciones objetivo a optimizar se clasifica como un problema NP- duro, debido a que no hay un algoritmo que lo pueda solucionar en tiempo computacional razonable [1]. Por ejemplo 10 trabajos alcanzan "10!" (3.628.800) secuencias posibles de ejecución en una máquina, aquí la pregunta sería ¿cuál es la mejor?, la respuesta a esta pregunta motiva la exploración y aplicación de nuevas opciones de solución y va a depender de las variables y de las funciones objetivo a optimizar. Los parámetros y variables del modelo se observan en la Tabla 2.

TABLA 2 Parámetros y variables flow shop.

PARAMETERS AND VARIABLES

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

M_k = Machine *k*, ∀ k=1, 2,..., m.

O_{ji} = Operation *j* in job i, ∀ j=1, 2,..., p.

n = Jobs number.

m = Machines number.

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

t_{ij} = Processing time job *i* in machine *j*.

c_i = Completion time of the job *i*.

TABLA 3 Funciones de tiempo flow shop.

TIME FUNCTIONS

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	$T_{max} = \max\{T_i\}$
Minimize number of late job	Min $\sum_{i=1}^{n} T_i$
Minimize makespan	Min C_{max}
Minimize maximum tardiness	Min T_{max}

Debido a la dificultad de minimizar costos o maximizar ganancias en un proceso de programación, se recurre a minimizar funciones de tiempo de terminación [4]. Algunas de estas se muestran en la Tabla 3.

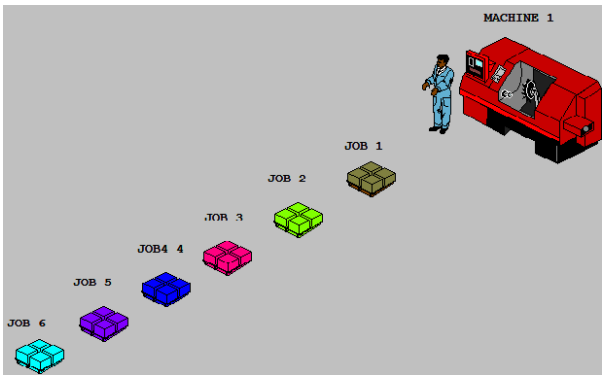

FIG. 1 Sistema de producción flow shop en una máquina.

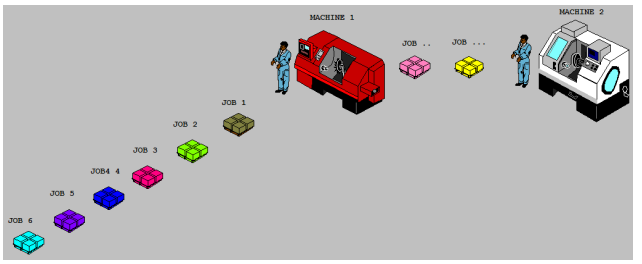

FIG. 2 Sistema de producción flow shop en dos máquinas.

El objetivo de este trabajo consiste en hallar un programa de producción aplicando la metaheuristica enjambre de partículas para los casos de n trabajos en una máquina (Fig. 1) y n trabajos en dos máquinas como se observa en la Fig. 2, de manera que se maximice la utilización de las máquinas y al mismo tiempo se minimice el makespan, el atraso máximo y el producto en proceso.

Las variables a tener en cuenta en el problema son: el tiempo de proceso de los trabajos en cada máquina y la fecha comprometida de entrega.

3 | METODOLOGÍA PROPUESTA

La complejidad del problema amerita proponer otras alternativas de solución como es el caso de particle swarm optimización (PSO). Esta técnica de optimización matemática, iterativa y estocástica se inspira en el comportamiento individual y social de bandadas de aves o bancos de peces para darle solución a funciones no lineales. Pertenece al "Área de la Inteligencia Artificial (IA), dedicada al estudio de la inteligencia colectiva emergente de un grupo de agentes simples" [8].

Debido a la naturaleza discreta y permutacional del problema, de las tres variantes principales del PSO (codificación continua, binaria y permutaciones de enteros), el método propuesto utiliza la permutación de enteros PSO-IP (particle swarm optimizatión - integer permutation) [9] [10], con actualización sincrónica de las partículas [11]. Ver diagrama de flujo de la Fig. 3.

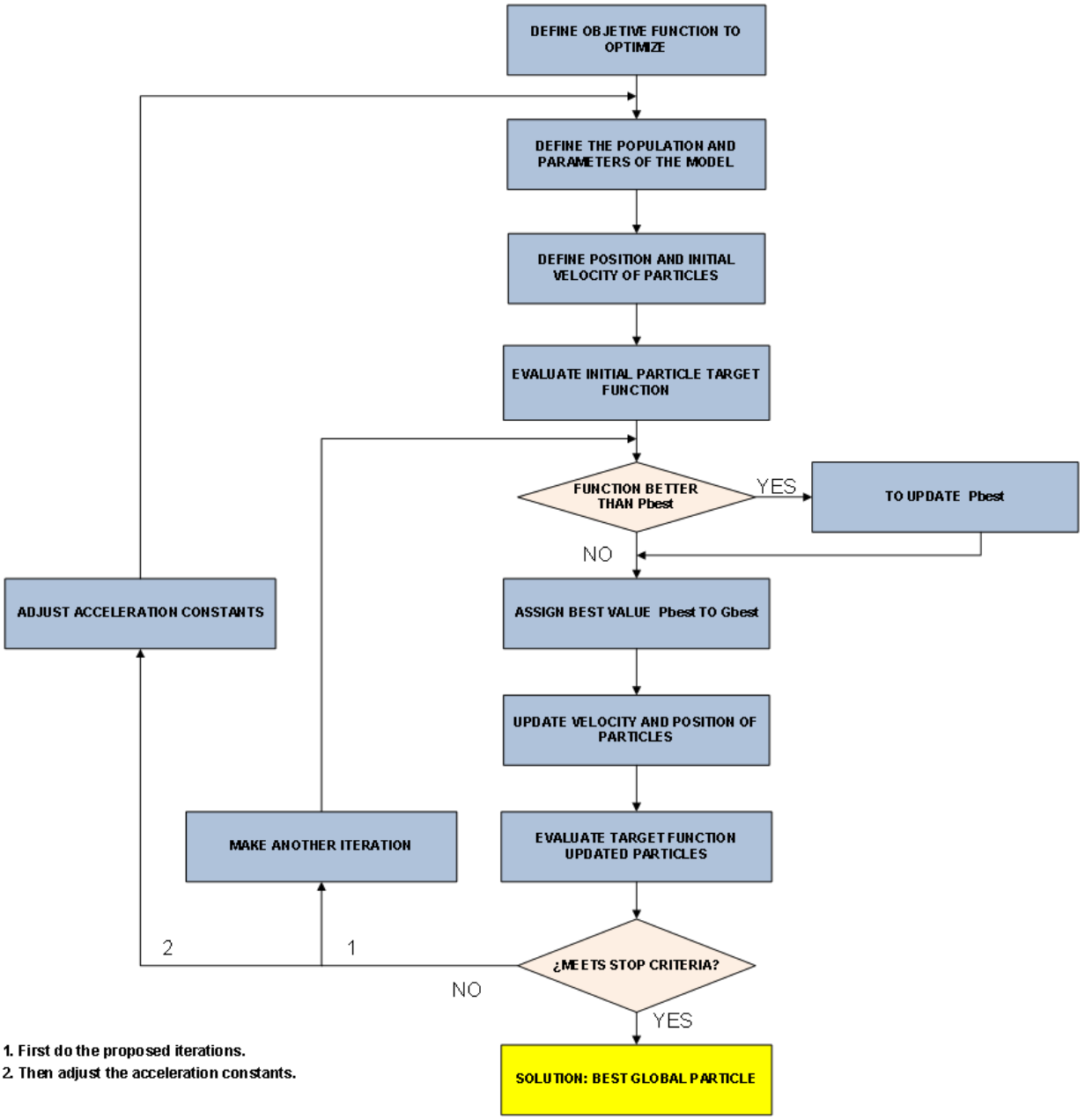

FIG. 3 Diagrama de flujo del método propuesto.

En PSO-IP, cada partícula se mueve en un espacio multidimensional que representa su espacio social. Cada partícula tiene memoria mediante la cual conserva parte de su estado anterior. Las líneas negras en la Fig. 4 representan la dirección actual de los vectores de velocidad: Vi,kPbest es la velocidad de la mejor posición de la partícula, V_i,kGbest es la velocidad de la mejor partícula encontrada por el enjambre y $V_{i,k}$ es la velocidad actual de la partícula. La línea roja $V_{i,k+1}$ representa la dirección que toma la partícula para moverse desde la posición $X_{i,k}$ hasta la posición $X_{i,k+1}$ [10].

3.1 | Vector posición de la partícula

La posición o localización de cada partícula en el espacio de búsqueda representa una secuencia o solución potencial al problema que se está resolviendo y se describe con la siguiente ecuación

$$\boldsymbol{X}_i = [p_{i1}, p_{i2}, \ldots, p_{in}] \tag{1}$$

donde $\boldsymbol{X}_i$ es la secuencia de la partícula, p_{ij} es el j-ésimo trabajo de la i-ésima partícula, y n denota el número de trabajos (Ver Fig. 4).

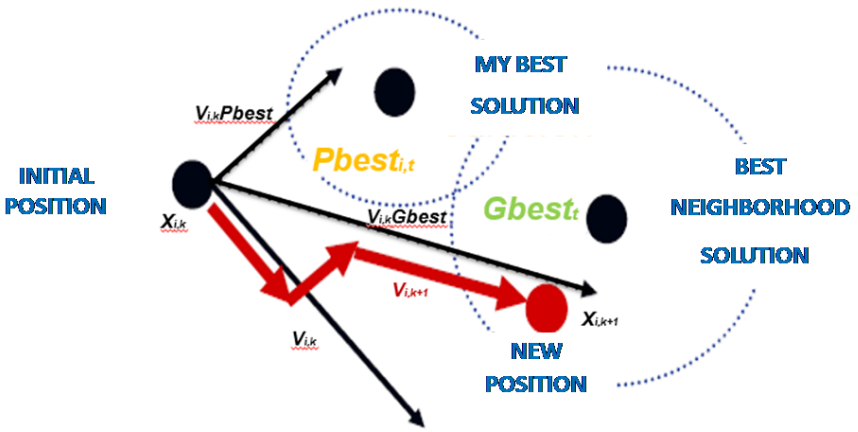

FIG. 4 Movimiento de la partícula.

3.2 | Vector velocidad de la partícula

La velocidad o dirección en la cual se moverá la partícula es representada por una lista de pares de trabajos $(p_i \longrightarrow p_j)$. Cada par de trabajos es un intercambio a realizar en $\boldsymbol{X}_{i,k}$ para obtener $\boldsymbol{X}_{i,k+1}$. La velocidad es

$$\boldsymbol{V}_i = [v_{i1}, v_{i2}, \ldots, v_{in}] = [(p_i \longrightarrow p_j)_{i1}, (p_i \longrightarrow p_j)_{i2}, \ldots, (p_i \longrightarrow p_j)_{in}] \tag{2}$$

donde $\boldsymbol{V}_i$ es la velocidad de la partícula i, v_{ij} es la velocidad asociado al trabajo j de la particula i, y $\boldsymbol{X}_{i,k}$ es la secuencia k de la partícula i.

3.3 | Actualización Vector velocidad

El vector velocidad de cada partícula es actualizado en cada iteración con la velocidad anterior, un componente individual cognitivo y un componente social, esto esta dado por la siguiente ecuación

$$\boldsymbol{V}_{i,k+1} = \boldsymbol{V}_{i,k} \bigoplus \boldsymbol{C}_1.\mathbf{rand} \bigotimes (\boldsymbol{Pbest}_{i,j} \bigominus \boldsymbol{X}_{i,k}) \bigoplus \boldsymbol{C}_2.\mathbf{rand} \bigotimes (\boldsymbol{Gbest}_k \bigominus \boldsymbol{X}_{i,k}) \tag{3}$$

donde $\bigominus, \bigoplus, \bigotimes$ son los operadores de permutación de enteros, C_1, C_2 son constantes de aceleración, $\mathbf{rand}$ es un número aleatoreo, $\boldsymbol{Pbest}_k$ se refiere a la mejor secuencia de las k secuencias de la partícula i, y $\boldsymbol{Gbest}_k$ es la mejor secuencia de las k de todas las partículas.

3.4 | Actualización Vector posición

Las partículas se desplazan de una posición a otra en el espacio de búsqueda multidimensional, sumando al vector posición $X_{i,k}$ el vector velocidad $V_{i,k+1}$. El nuevo vector posición es

$$\boldsymbol{X}_{i,k+1} = \boldsymbol{X}_{i,k} \bigoplus \boldsymbol{V}_{i,k+1}. \tag{4}$$

3.5 | Funcionamiento del PSO-IP

Para encontrar la mejor solución del problema, PSO-IP comienza a partir de diferentes soluciones (partículas) iniciales que exploran en paralelo diferentes regiones del espacio de búsqueda, luego retroalimentan las mejores posiciones encontradas al enjambre y de estas se selecciona la mejor [12]. La descripción especifica de su dinámica se hace en los siguientes ítems (ver Fig. 5).

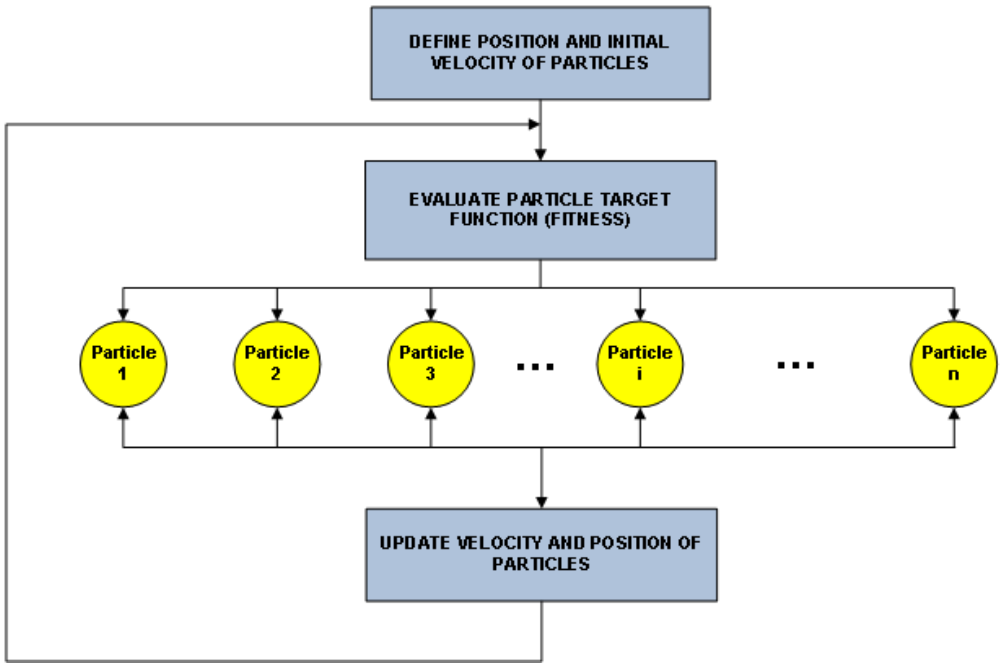

FIG. 5 Exploración del espacio en paralelo.

3.5.1 | Población inicial de partículas

Una vez determinado el número de partículas del enjambre en función de la complejidad del problema [13] [14], se generan en forma aleatoria las posiciones y las velocidades iniciales de las partículas. En la Tabla 4 se observan estos dos vectores para una partícula.

TABLA 4 Posición y velocidad inicial.

POSITION	P		P		P		P		P		P	
X_1	1		5		6		2		3		4	
VELOCITY	P	P	P	P	P	P	P	P	P	P	P	P
V_1	5	4	5	6	6	2	3	2	2	4	6	5

3.5.2 | Función objetivo

A cada partícula se le calcula la función objetivo definida, ya sea minimizando el atraso o el makespan, de acuerdo a las ecuaciones planteadas en la Tabla 3.

3.5.3 | Determinar Pbest y Gbest

Por comparación se halla la mejor posición por partícula (Pbest) y la mejor posición de todas las partículas (Gbest).

3.5.4 | Actualización de velocidad

Para el cálculo del comportamiento individual y social se seleccionan las constantes de aceleración C_1 y C_2, y se utilizan los operadores para permutación de enteros.

- **Operador resta de posiciones** $\ominus$. Al restar dos posiciones el resultado es una lista velocidad.

$$\boldsymbol{X}_1 \bigominus \boldsymbol{X}_1 = [(p_{21} \longrightarrow p_{11}), (p_{22} \longrightarrow p_{12}), \ldots, (p_{2n} \longrightarrow p_{1n})]$$

  donde $\boldsymbol{X}_1 = [p_{11}, p_{12}, \ldots, p_{1n}]$ y $\boldsymbol{X}_2 = [p_{21}, p_{22}, \ldots, p_{2n}]$.
- **Operador suma de velocidades** $\oplus$. La suma de dos velocidades consiste en solapar los pares de dichas velocidades, obteniendo una nueva velocidad.

$$(p_i \longrightarrow p_j) \bigoplus (p_j \longrightarrow p_k) = (p_i \longrightarrow p_k)$$

$$(p_i \longrightarrow p_j) \bigoplus (p_k \longrightarrow p_m) = (p_i \longrightarrow p_j)$$

- **Operador producto coeficiente velocidad** $\otimes$. El producto de un coeficiente por una velocidad genera una nueva velocidad.

$$\text{rand}_j \leq C_1 \Rightarrow (p_i \longrightarrow p_j) \rightarrow (p_i \longrightarrow p_j)$$

$$\text{rand}_j > C_1 \Rightarrow (p_i \longrightarrow p_j) \rightarrow (p_i \longrightarrow p_j)$$

3.5.5 | Actualización de posición

La actualización de la posición de las partículas se realiza con la ecuación (4), aplicando el operador suma de posición con velocidad $\otimes$, este operador permuta los trabajos de una secuencia (posición) para generar una nueva secuencia por intercambio de trabajos. Una aplicación es la siguiente: sean

$$\boldsymbol{X}_1 = [p_1, p_4, p_3, p_2], \quad y \quad \boldsymbol{V}_2 = [(p_1 \longrightarrow p_4), (p_3 \longrightarrow p_1), (p_3 \longrightarrow p_2), (p_1 \longrightarrow p_3)]$$

entonces

$$\boldsymbol{X}_1 \bigoplus \boldsymbol{V}_2 = [p_4, p_1, p_3, p_2]$$

$$\boldsymbol{X}_1 \bigoplus \boldsymbol{V}_2 = [p_4, p_3, p_1, p_2]$$

$$\boldsymbol{X}_1 \bigoplus \boldsymbol{V}_2 = [p_4, p_2, p_1, p_3]$$

$$\boldsymbol{X}_1 \bigoplus \boldsymbol{V}_2 = [p_4, p_2, p_1, p_3]$$

la nueva posición es

$$\boldsymbol{X}_1 \bigoplus \boldsymbol{V}_2 = [p_4, p_2, p_3, p_1]$$

3.5.6 | Número de iteraciones

Una vez actualizada la posición se repite el procedimiento como se ve en la Fig. 3. La metaheuristica termina cuando se cumpla con el criterio de parada, que por lo regular consiste en proponer inicialmente un numero de iteraciones a desarrollar o cuando no se consiga alguna mejora.

TABLA 5 Pseudocódigo del algoritmo propuesto.

```
S = initial particle swarm ()

While stop condition not reached do

    for i=1 to size(S) do

        evaluate each particle Xi of the swarm S

        if fitness(Xi) is better than fitness(Pbesti) then
            Pbesti = Xi ; fitness(Pbesti) = fitness(Xi)
        end if

        if fitness(Pbesti) is better than fitness(Gbesti) then
            Gbesti = Pbesti ; fitness(Gbesti) = fitness(Pbesti)
        end if

    end for

    for i=1 to size(S) do

        update velocity and position of each particle Xi of the swarm S

        V_{i,k+1} = V_{i,k} ⊕ C1.rand ⊗ (Pbest_{i,k} ⊖ X_{i,k}) ⊕ C2.rand ⊗ (Gbest_k ⊖ X_{i,k})

        X_{i,k+1} = X_{i,k} ⊕ V_{i,k+1}

    end for

end while

Exit: best solution found
```

Para la implementación del método se desarrolla el pseudocódigo que se muestra en la Tabla 5, este proporciona una descripción general de los pasos de cómo se generan y actualizan las posiciones y velocidades de las partículas [15] [16] [17].

3.6 | Simulación y software

Para la simulación del PSO-IP, se creó una hoja dinámica Microsoft® Excel® que genera y actualiza posiciones y velocidades de las partículas, luego con ®Matlab [18] [19] se simula el vuelo o desplazamiento, tal como se muestra en la Fig. 6.

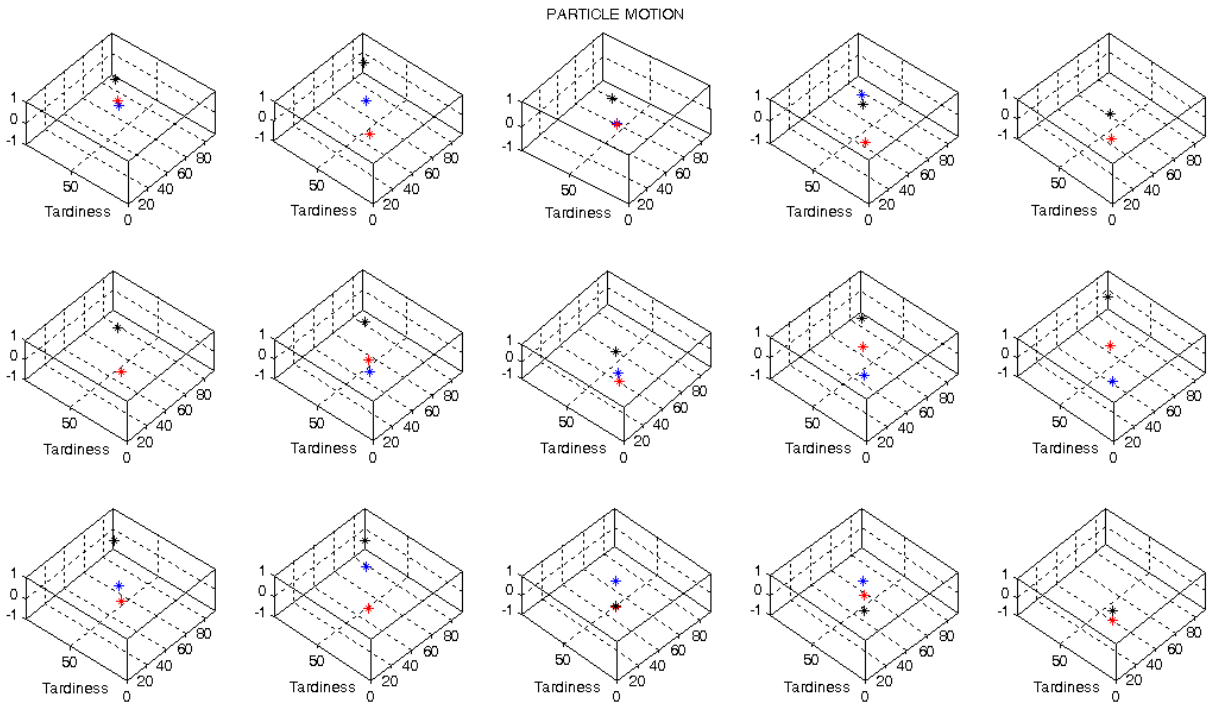

FIG. 6 Desplazamiento de las partículas.

4 | RESULTADOS Y DISCUSIÓN

4.1 | n-trabajos en una máquina

Para este caso, tradicionalmente solo se utiliza el tiempo de proceso de los trabajos para minimizar el makespan. Sin embargo, como el objetivo es multicriterio se requiere del tiempo de entrega para optimizar también el factor servicio otorgado al cliente y la utilización de las máquinas. En la Tabla 6 se presentan estos tiempos para 6 trabajos en una máquina.

TABLA 6 Seis trabajos en una máquina.

Job	t_{ij}	te_{ij}
P1	10	15
P2	3	16
P3	16	24
P4	8	30
P5	4	35
P6	11	37

De las tres partículas iniciales en el enjambre, dos de ellas alcanzan la solución óptima en dos iteraciones, obteniendo las siguientes secuencias:

$$\boldsymbol{X}_2 = [p_2, p_1, p_5, p_4, p_5, p_3], \qquad \boldsymbol{X}_1 = [p_2, p_1, p_5, p_4, p_5, p_3].$$

En cada una de ellas la sumatoria del atraso es de 28 horas. En la Tabla 7 se ve que simultáneamente estas secuencias alcanzan la solución óptima y mejores resultados en cinco de las ocho funciones objetivo a optimizar, cuando se comparan con reglas de despacho y metaheuristicas como búsqueda en la vecindad (intercambio adyacente por pares, intercambio por pares e inserción).

TABLA 7 Comparación de resultados M1.

SEQUENCING METHOD	MAKESPAN	SUM TARDINESS	MAXIMUM WAITING TIME	MAXIMUM EARLINESS	MAXIMUM TARDINESS	WORK IN PROCESS	MEAN MACHINE UTILIZATION	SERVICE FACTOR
FCFS(First Come, First Served)	52	33	41	5	15	3,50	100%	33,33%
SPT(Shortest Process Time)	52	38	36	28	28	2,65	100%	66,67%
EDD(Earliest Due Date)	52	33	41	5	15	3,50	100%	33,33%
SLACK(Slack Time)	52	50	48	5	17	3,88	100%	16,67%
METAHEURISTICS-BVIAP	52	31	41	5	15	3,42	100%	50,00%
METAHEURISTICS-BVIP	52	29	36	21	28	2,96	100%	66,67%
METAHEURISTICS-BVI	52	29	36	31	28	2,85	100%	66,67%
FULL ENUMERATION (OPTIMUM)	52	28	36	10	28	3,01	100%	83,33%
PARTICLE SWARM OPTIMIZATION2	52	28	36	18	28	2,80	100%	83,33%
PARTICLE SWARM OPTIMIZATION1	52	28	36	18	28	2,94	100%	83,33%

Con respecto a las reglas de secuenciación y las metaheuristicas, se mejora la sumatoria del atraso en 3.45% y el factor servicio en 19.9%, conservando la espera máxima, el tiempo de terminación, y la utilización de las maquinas igual a la solución óptima. ver Tabla 8.

TABLA 8 Análisis de resultados M1.

SEQUENCING METHOD	MAKESPAN	SUM TARDINESS	MAXIMUM WAITING TIME	MAXIMUM EARLINESS	MAXIMUM TARDINESS	WORK IN PROCESS	MEAN MACHINE UTILIZATION	SERVICE FACTOR
FCFS(First Come, First Served)					15			
SPT(Shortest Process Time)			36			2,65		
EDD(Earliest Due Date)				5				
SLACK(Slack Time)								
METAHEURISTICS-BVIAP	52						100%	
METAHEURISTICS-BVIP		29						66,67%
METAHEURISTICS-BVI								
FULL ENUMERATION (OPTIMUM)	0,00%	3,45%	0,00%	50,00%	46,43%	11,96%	0,00%	19,99%
PARTICLE SWARM OPTIMIZATION2	0,00%	3,45%	0,00%	72,22%	46,43%	5,36%	0,00%	19,99%
PARTICLE SWARM OPTIMIZATION1	0,00%	3,45%	0,00%	72,22%	46,43%	9,86%	0,00%	19,99%

5 | N-TRABAJOS EN DOS MÁQUINAS

De forma similar a lo planteado para una máquina, para el caso de dos máquinas se requieren los tiempos de proceso en cada máquina y el tiempo comprometido de entrega de los trabajos, que se observan en la tabla 9.

TABLA 9 Seis trabajos en dos máquinas.

Job	t_{i1}	t_{i2}	te_{ij}
P1	3	2	40
P2	5	7	16
P3	6	4	45
P4	7	8	18
P5	10	9	32
P6	8	6	40

A partir de estos datos, dos de las tres partículas iniciales consiguen obtener la posición objetivo en la cuarta iteración, generando así, los programas que se muestran en la Fig. 7 y Fig. 8, respectivamente.

$$\boldsymbol{X}_2 = [p_2, p_4, p_5, p_1, p_6, p_3]$$

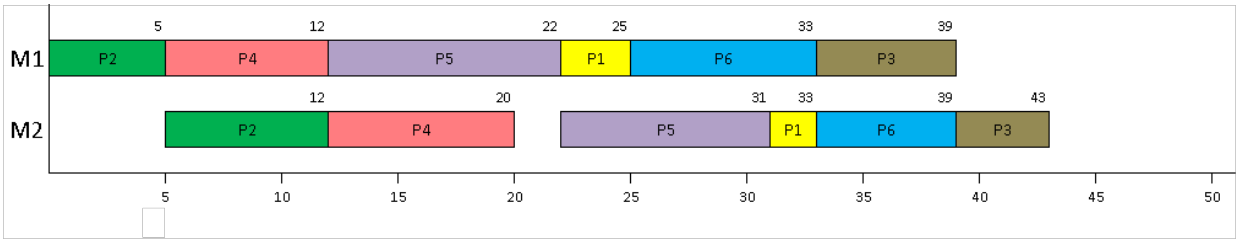

FIG. 7 Programa óptimo PSO-P2.

Al igual que el algoritmo de Johnson, se observa que las dos secuencias alcanzan un makespan óptimo de 43 horas.

$$\boldsymbol{X}_3 = [p_4, p_2, p_5, p_6, p_1, p_3]$$

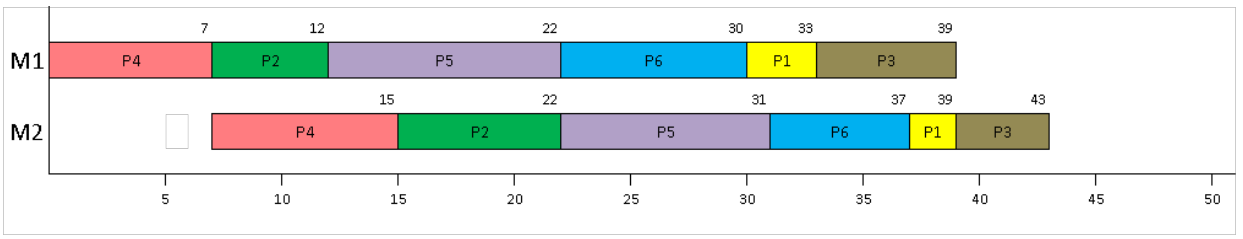

FIG. 8 Programa óptimo PSO-P3.

La Tabla 10 muestra que las dos secuencias (partículas 2 y 3), logran mejores resultados en seis de las ocho funciones objetivo a optimizar, al ser comparadas con heurísticas y reglas de despacho.

Aquí se disminuye la sumatoria del atraso en 40%, el atraso máximo en 33.33% y el factor servicio aumenta en 20%, se mantienen igual a la respuesta optima el makespan, el adelanto máximo y la utilización de las maquinas (ver Tabla 11).

TABLA 10 Comparación de resultados M2.

SEQUENCING METHOD	MAKESPAN	SUM TARDINESS	MAXIMUM WAITING TIME	MAXIMUM EARLINESS	MAXIMUM TARDINESS	WORK IN PROCESS	MEAN MACHINE UTILIZATION	SERVICE FACTOR
FCFS(First Come, First Served)	46	25	32	35	11	3,35	81,52%	50,00%
SPT(Shortest Process Time)	48	40	29	35	19	3,17	78,13%	50,00%
HEURISTICS-CDS	43	5	38	4	3	4,28	87,21%	66,67%
HEURISTICS-PALMER	45	6	31	18	5	3,80	83,33%	50,00%
JOHNSON (OPTIMUM)	43	5	38	4	3	4,28	87,21%	66,67%
PARTICLE SWARM OPTIMIZATION2	43	3	33	4	2	4,14	87,21%	83,33%
PARTICLE SWARM OPTIMIZATION3	43	5	34	6	4	4,35	87,21%	83,33%

TABLA 11 Análisis de resultados M2.

SEQUENCING METHOD	MAKESPAN	SUM TARDINESS	MAXIMUM WAITING TIME	MAXIMUM EARLINESS	MAXIMUM TARDINESS	WORK IN PROCESS	MEAN MACHINE UTILIZATION	SERVICE FACTOR
FCFS(First Come, First Served)								
SPT(Shortest Process Time)			29			3,17		
HEURISTICS-CDS	43	5		4	3		87,21%	66,67%
HEURISTICS-PALMER								
JOHNSON (OPTIMUM)	0,00%	0,00%	23,68%	0,00%	0,00%	25,92%	0,00%	0,00%
PARTICLE SWARM OPTIMIZATION2	0,00%	40,00%	12,12%	0,00%	33,33%	23,42%	0,00%	20,00%
PARTICLE SWARM OPTIMIZATION3	0,00%	0,00%	14,71%	33,33%	25,00%	27,11%	0,00%	20,00%

Conclusiones

Se puede afirmar que la metodología propuesta con base en PSO-IP permite solucionar el problema combinatorio de secuenciación en sistemas de producción flow shop, atendiendo al cumplimiento simultaneo de varias funciones objetivo a optimizar (makespan, factor servicio, utilización de máquinas), a partir de la emulación del comportamiento individual y social de grupos de animales como lo son las aves y los peces. Esta alternativa evita la compleja modelación matemática que tradicionalmente envuelve este tipo de problemas. Como se comprobó las soluciones que se obtienen con PSO-IP igualan el makespan (43 horas) y la utilización de las maquinas (87.21%) que se obtiene con el algoritmo óptimo de Johnson para dos máquinas, mejorando el factor servicio en 20%. Para el caso de una maquina se mejora la sumatoria del atraso en 3.45% y el factor servicio en 19.9%.

AGRADECIMIENTOS

Agradecemos al grupo de investigación NEURONA, semillero de investigación TABÚ del Centro de Gestión Industrial SENA, por el financiamiento de este trabajo.

References

[1] L. C, “Programación de las actividades logísticas con algoritmos genéticos,” 2013.

[2] P. Companys and I. Ribas, “Johnson revisited, ten years later,” 2009.

[3] M. Pinedo, *Scheduling Theory, Algorithms, and Systems*. New York: Springer, 2012.

[4] L. C, “Programming of job shop production systems with fuzzy logic,” *Tecciencia*, vol. 13, no. 25, p. 39–45, 2018.

[5] J. M. Framinan, J. N. Gupta, and R. Leisten, "A review and classification of heuristics for permutation flow-shop scheduling with makespan objective," *Journal of the Operational Research Society*, vol. 55, no. 12, pp. 1243–1255, 2004. **DOI:** 10.1057/palgrave.jors.2601784

[6] A. Esmorís, "Algoritmos heurísticos en optimización," 2013.

[7] E. M. T. Ocampo, M. G. Echeverri, and Y. S. Restrepo, "Adaptación de la técnica de particle swarm al problema de secuenciamiento de tareas.," *Scientia et technica*, vol. 3, no. 32, pp. 307–312, 2006.

[8] J. Kennedy, R. Eberhart, and S. Yuhui, *Swarm Intelligence*. San Francisco: Morgan Kaufmann Publishers, 2001.

[9] M. Clerc, "Discrete particle swarm optimization, illustrated by the traveling salesman problem," in *New optimization techniques in engineering*, p. 219–239, Berlin: Springer, 2004. **DOI:** 10.1007/978-3-540-39930-8_8

[10] J. Nieto, "Algoritmos basados en cúmulos de partículas para la resolución de problemas complejos," *Universidad de Málaga, Málaga*, 2006.

[11] J. Trespalacios, *Metodología de Galerkin libre de malla (EFG) para la optimización de forma de estructuras elásticas*. Universidad Tecnológica de Bolívar, Cartagena de Indias, 2018.

[12] C. Corona, *Estrategias coordinadas paralelas basadas en soft-computing para la solución de problemas de optimización*. Granada: Universidad de Granada, 2005.

[13] A. Engelbrecht, "Fitness function evaluations: Afair stopping condition?," in *de IEEE Symposium on Swarm Intelligence*, (Orlando, FL, USA), 2014. **DOI:** 10.1109/SIS.2014.7011793

[14] S. Strasser, R. Goodman, J. Sheppard, and S. Butcher, "A new discrete particle swarm optimization algorithm," in *de Genetic and Evolutionary Computation Conference*, (Denver, CO, USA), 2016. **DOI:** 10.1145/2908812.2908935

[15] E. Mansour, J. Bassem, and S. Patrick, "Combinatorial particle swarm optimization for solving blocking flowshop scheduling problem," *Journal of Computational Desing and Engineering*, pp. 295–311„ 2016.

[16] S. Sridhar, S. Sabareesan, and R. Kannan, "Particle swarm optimization approach for flow shop scheduling problem- a case study," *International Journal of Mechanical Engineering and Technology*, vol. 9, pp. 72–80„ 2018.

[17] L. Bewoor, V. Chandraprakash, and S. Sapkal, "Evolutionary hibrid particle swarm optimization algorithm to minimize makespan toschedule a flow shop with no wait," *Journal of Engineering Science and Technology*, vol. 14, no. 2, pp. 609–628„ 2019.

[18] *The Math Works, Inc, Matlab User s Guide*. Natick: 3 Apple Hill Drive, 2020.

[19] H. Moore, *Matlab para ingenieros*. México D.F: Prentice Hall, Inc, 2018.