Un Squirrel Search Algorithm discreto aplicado al problema Job Shop con operadores calificados

A Discrete Squirrel Search Algorithm applied to the Job Shop problem with skilled operators

DOI: http://doi.org/10.17981/ingecuc.15.2.2019.14

Artículo de Investigación Científica. Fecha de Recepción 16/07/2019. Fecha de Aceptación: 30/11/2019

César Andrés López Martínez
Universidad Pontificia Bolivariana. Montería (Colombia)
cesar.lopezma@upb.edu.co

Helman Enrique Hernández Riaño
Universidad de Córdoba. Montería (Colombia)
hhernandez@correo.unicordoba.edu.co

Manuel Jesús Soto de la Vega
Universidad Tecnológica de Bolívar. Cartagena (Colombia)
mjsoto@utb.edu.co

Para citar este artículo:
C. López Martínez, H. Hernández Riaño y M. Soto de la Vega. "Un Squirrel Search Algorithm discreto aplicado al problema Job Shop con operadores calificados", *INGE CUC*, vol. 15, no. 2, pp. 143–154, 2019. DOI: http://doi.org/10.17981/ingecuc.15.2.2019.14

Resumen

Introducción– El problema Job Shop Con Operadores Calificados o Job Shop With Skilled Operators (JSSO) es una extensión del problema clásico Job Shop en el cual, una operación debe ser ejecutada por un conjunto limitados de trabajadores, con el objetivo de minimizar el tiempo de terminación total de los trabajos o Makespan, situación que puede representar distintas aplicaciones en la vida cotidiana. Es un problema complejo y es catalogado como NP-HARD.

Objetivo– En este artículo, se aborda el problema JSSO desde la adaptación de un algoritmo conocido como Squirrel Search Algorithm (SSA).

Metodología– Se propone un esquema de codificación discreto para el algoritmo SSA utilizando el método Smallest Position Value (SPV). Además, para evitar soluciones que violen las relaciones de precedencia; se corrigen con el método Valid Particle Generator (VPG), el cual garantiza soluciones factibles. Dos versiones del algoritmo se colocan a prueba en 28 instancias propuesta en la literatura para validar su rendimiento.

Resultados– Los experimentos computacionales realizados muestran que los dos algoritmos propuestos alcanzan soluciones óptimas en 25 de las 28 instancias analizadas. Además, para las instancias en donde no se logró soluciones óptimas, el gap promedio no supera el 2% para ambas versiones de los algoritmos propuestos.

Conclusiones– El esquema de codificación propuesto garantiza la discretización del algoritmo, generando soluciones que convergen hacia el óptimo. Además, la codificación propuesta, permite utilizar de manera natural los operadores de movimiento propuestos originalmente para el algoritmo utilizado. El rendimiento obtenido por los algoritmos es adecuado y de alta calidad.

Palabras clave– Inteligencia de Enjambres; Secuenciación con Operadores; Posición del Valor más Pequeño; Generador Válido de Partículas, Optimización Combinatoria.

Abstract

Introduction– The Job Shop problem With Skilled Operators (JSSO) is an extension of the classic Job Shop in which, an operation must be executed by a limited set of workers, aiming to minimize jobs total termination time or Makespan. This situation can represent different applications in daily life. JSSO is a complex problem and its classified as NP-HARD..

Objective– In this article, the JSSO problem is addressed. It is made by adapting an algorithm known as Squirrel Search Algorithm (SSA).

Methodology– A discrete encoding scheme is proposed for the SSA algorithm and the Smallest Position Value (SPV) method are used. Also, solutions that can violate the precedent relationships are corrected with the Valid Particle Generator (VPG) method, which guarantees feasible solutions. Two versions of the algorithm were tested in 28 instances proposed in the literature to valid their performance.

Results– Computer experiments show that the proposed algorithms reach optimal solutions in 25 and 28 analyzed instances. In addition, for the instances where optimality was not achieved, the average gap does not exceed the 2% for both versions of the proposed algorithms.

Conclusions– The proposed encoding scheme guarantees the discretization of the algorithms, generating solutions that converge towards the optimum. In addition, the proposed encoding allows natural use of movement operators originally proposed for the algorithms used. Performance obtained by the algorithms is adequate and of high quality.

Keywords– Swarm Intelligence; Scheduling with Operators; Smallest Position Value; Valid Particle Generator, Combinatorial Optimiation.

I. Introduccion

En este artículo se propone solucionar por medio de un método metaheurístico el problema Job Shop con Operadores Calificados conocido en la literatura como Job Shop With Skilled Operators (JSSO) [1]. La motivación de este problema radica en la particularidad de algunos sistemas de producción en donde se requiere la coexistencia tanto de recursos tecnológicos, como de mano de obra. En estos sistemas de producción, como por ejemplo la industria tabacalera artesanal, así como la de la Joyería y Bisutería, la especialización del trabajo juega un papel fundamental. En estos sistemas de producción, se asume que los operadores tienen ciertas habilidades para realizar una tarea y de esta manera su ejecución dependerá, tanto de la pericia del operador, como de la herramienta o máquina a utilizar. Debido a que el número de operadores es siempre menor al número de recursos tecnológicos disponibles, algunas tareas deberán esperar para su ejecución mientras que el operador calificado esté disponible, o en su defecto, que el recurso tecnológico este libre para poder ejecutarla. Este problema es considerado complejo y es catalogado como NP-HARD.

La complejidad de algunos problemas de optimización motiva el uso de métodos de solución alternativos que brinden soluciones de buena calidad cuando se aborden problemas de gran tamaño. Las metaheurísticas surgen como una alternativa de solución para resolver estos tipos de problemas, y por esto, se consideran como estructuras algorítmicas que generalmente se aplican a una variedad de problemas de optimización con solo unas pocas modificaciones para adaptarse al problema dado [2]. En este orden de ideas, se propone adaptar una metaheurística para solucionar el problema JSSO, considerado como el movimiento de las ardillas al buscar alimento y se conoce como Squirrel Search Algorithm (SSA) [3]. Es una metaheurística bio-inspirada, clasificada dentro del grupo de los algoritmos de inteligencia de enjambres o Swarm Intelligence (SI).

Diversas adaptaciones de algoritmos de inteligencia de enjambres se han utilizado en los últimos años en problemas Job Shop o en algunas de sus extensiones. En otros trabajos, se adaptó la metaheurística Particle Swarm Optimization (PSO) con una heurística Variable Neighboord Search (VNS) [4], creando así un algoritmo híbrido para solucionar el problema Job Shop clásico con el objetivo de minimizar el tiempo de terminación total de los trabajos o Makespan. Por su parte, tambien se analiza un problema Flexible Job Shop (FJSP) considerando la llegada de nuevos trabajos, obligando a hacer una reprogramación [5] donde el problema es abordado desde la perspectiva de un algoritmo de colonia de abejas en dos etapas o Two-Stage Artificial Bee Colony (TABC) con el fin de minimizar el makespan. También se abordó un Problema Flexible Job Shop (FJSP), considerando que la incertidumbre de los tiempos de proceso son modelados bajo tiempos de procesamiento difusos [6], adaptando la metaheurística Artificial Bee Colony (ABC) para minimizar el tiempo máximo de finalización difusa y la carga de trabajo máxima difusa de la máquina. Una metaheurística inspirada en los acordes musicales o Harmony Search (HS), fue adaptada nuevamente al problema Flexible Job Shop (FJSP) [7]. En este trabajo, se creó una metaheurística híbrida al combinar la metaheurística HS con un modelo para resolver problemas de optimización conocido como Computación basada en Membranas o Membrane Computing (MC) con el objetivo de minimizar el Makespan.

En una interesante extensión del problema Job Shop se consideró la integración de la programación de turnos de empleados con la programación de máquinas y del personal dedicado al transporte de productos entre las máquinas considerado como “Transportadores” [8]. Se realiza la diferencia entre los empleados dedicados al procesamiento de las actividades y como se mencionó anteriormente, al personal dedicado al transporte de productos entre máquinas. En particular, este personal maneja diferentes velocidades de traslado, razón por la cual el tiempo de traslado entre dos máquinas va a depender de la velocidad del personal asignado. Los objetivos del problema propuesto son en primer lugar; asignar un conjunto de actividades a los empleados durante un turno de trabajo y determinar el tiempo de inicio de las actividades, así como también el de los desplazamientos de los transportadores. El problema es catalogado como NP-HARD y se adaptó una metaheurística denominada Anarchic Society Optimization Algorithm (ASOA) con el fin de minimizar el Makespan. En otro trabajo se adaptó una metaheurística basada en la migración e inmigración de especies entre hábitat llamada Biogeography-Based Optimization (BBO) con el fin de minimizar el Makespan [9]. Una nueva adaptación del algoritmo colonia de abejas o ABC fue utilizado para mantener un balance entre las fases de exploración y explotación, e incluye el fenómeno de la espuma de la cerveza como método de generación de nuevas posiciones [10]. La adaptación se realizó para un problema Job Shop Clásico con el fin de minimizar el Makespan. Por último, un eficiente algoritmo basado en las reacciones químicas llamado Chemical Reaction Optimization (CRO) fue adaptado para solucionar una extensión del problema Flexible Job Shop conocido como Distributed And Flexible Job Shop Problem (DFJSP) con el fin de minimizar el makespan [11].

En este punto, se puede observar que la adaptación e implementación de algoritmos basados en inteligencia de enjambre en problemas de alta complejidad, es una práctica que se ha venido realizando en los últimos años, proponiendo algoritmos capaces de brindar soluciones acordes a la complejidad del problema estudiado. Esto permite inferir la posibilidad de seguir trabajando la adaptación de estos métodos en problemas de alta complejidad en los próximos años. Ahora bien, considerar recursos restringidos en un problema clásico Job Shop es una de las extensiones que se puede encontrar en la literatura. Unos de los trabajos más significativos considerando esta extensión considera un problema Job Shop con operadores o Job Shop With Operators (JSO), en donde se tiene un conjunto de trabajos que requiere ser procesados en un conjunto de máquinas utilizando un conjunto de operadores [12]; y el conjunto de restricciones adicionales que diferencia este problema de un Job Shop clásico es que las operaciones deben ser realizadas por un operador y un operador puede tener como mucho una operación asignada. En este artículo, los autores demostraron que el mínimo caso para que este problema sea catalogado como NP-HARD, al considerar tres trabajos, tres máquinas y dos operadores. En este mismo trabajo, los autores proponen diferentes enfoques de solución, el cual compara un algoritmo basado en programación dinámica, un algoritmo Branch & Bound (B&B) y una heurística. Los experimentos computacionales muestran como el esquema de solución basado en programación dinámica es competitivo frente al algoritmo B&B y la heurística propuesta es aplicable a problemas de mayor tamaño.

En la misma perspectiva de métodos de solución aplicados al problema JSO [13], se presenta una metodología para generar un esquema de generación de programas inspirado en el algoritmo de Giffler & Thompson (G&T) llamado OG&T [14]. Este esquema es utilizado en el diseño de dos algoritmos, un algoritmo Best First Search y un algoritmo voraz (Greedy), los cuales se hibridan obteniendo una herramienta que supera en los experimentos computacionales al desarrollo por programación de restricciones. Además se propone un algoritmo genético para el problema JSOP [15] con un enfoque de solución que considera la generación de esquemas OG&T para la generación de soluciones activas [13]. El algoritmo propuesto es comparado con métodos de solución [15] obteniendo un eficiente rendimiento. Un enfoque similar [16], se aborda con el problema JSO bajo un algoritmo genético mejorado consistente en utilizar la Evolución Lamarckiana modificando el esquema de generación de soluciones OG&T, con el fin de reducir el tiempo ocioso de las máquinas y operadores. Esta mejora permite al algoritmo genético trabajar problemas de gran tamaño. Los experimentos computacionales realizados muestran como el algoritmo mejorado presenta un rendimiento eficiente frente los encontrados en la literatura.

En un enfoque alternativo para la solución del problema JSO se propone abordar la solución del problema en varias etapas [17]. La primera es hacer una relajación del problema solucionándolo sin tener en cuenta los operadores. Se propone resolver este problema bajo el enfoque Solving a Constraint Satisfaction and Optimization Problem (CSOP). En este punto, la solución obtenida es modificada incluyendo los operadores y se tiene en cuenta el concepto de robustez. Esta se logra asignado amortiguadores o “buffers” en la secuencia de los operadores. Finalmente, los amortiguadores son distribuidos uniformemente entre los operadores que no hacen parte de la ruta crítica. Los resultados obtenidos por medio de esta técnica propuesta muestran un rendimiento aceptable frente a los métodos utilizados hasta el momento.

Ahora bien, para el problema analizado en este artículo (JSSO) la literatura solo ha reportado soluciones heurísticas [1], motivando la adaptación e implementación de métodos de solución aproximados para solucionar este problema.

II. Metodología

Para este trabajo se definieron las etapas que se consolidan en la Fig. 1 donde se inicia con la identificación tanto del problema a estudiar como de la metaheurística a utilizar. Posteriormente, se adaptó la metaheurística al problema. Luego se realizaron los experimentos computacionales y finalmente se analizaron los resultados obtenidos para verificar la pertinencia de adaptar la metaheurística al problema.

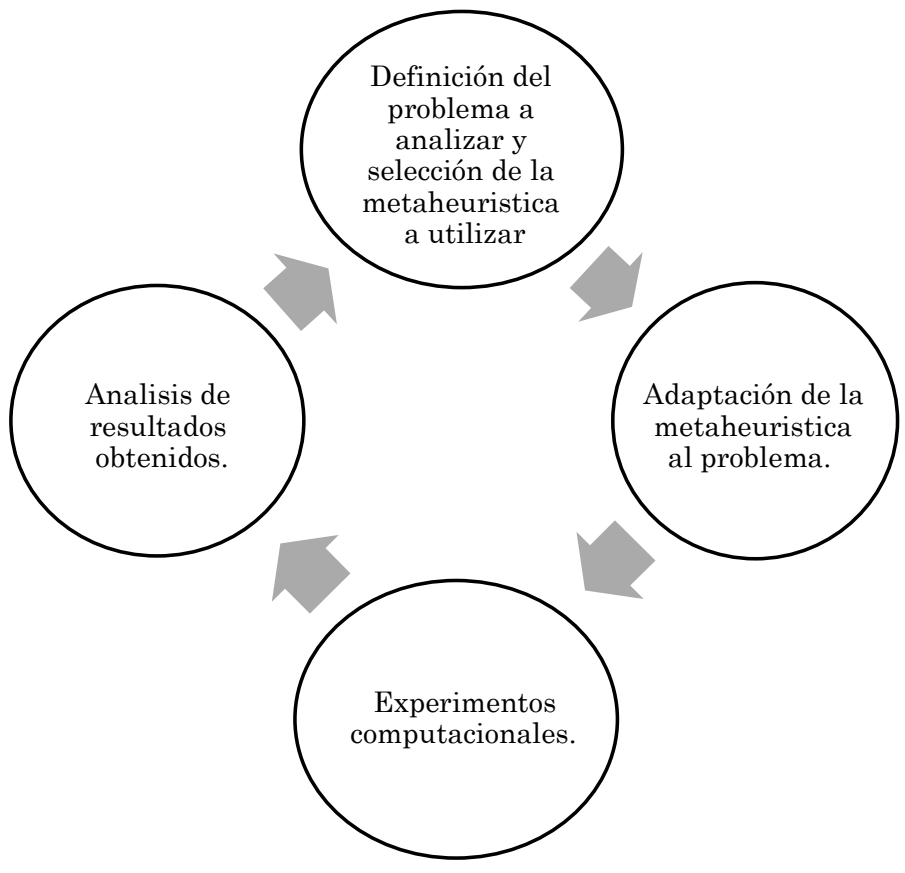

Fig. 1. Metodología Propuesta.
Fuente: Autores.

A. Problema Job Shop con Operadores calificados

Algunos procesos de producción presentan características especiales las cuales deben ser analizadas en detalle. En procesos de fabricación tales como el de tabaco artesanal, joyas y bisutería es necesaria la coexistencia de diferentes recursos tanto tecnológicos (máquinas, herramientas y equipos) como humanos. Además de esto, ciertas operaciones de los procesos mencionados involucran el uso de una habilidad específica de los operadores para ser desarrolladas. Las operaciones siguen relaciones de precedencia donde cada una gasta un tiempo en ser realizada y utilizan un recurso tanto tecnológico como de mano obra para ser ejecutadas. Para no afectar el tiempo de terminación total de las tareas, se debe establecer una adecuada asignación de los operadores a las operaciones y de estas a los recursos tecnológicos. Cuando se tienen estas características, teniendo en cuenta que el número de operadores es siempre menor al número de recursos tecnológicos, se define un problema de programación de producción de tipo Job Shop con Operadores Calificados [1] y se puede describir por medio del siguiente modelo matemático:

1. Índices

i = Índice para las tareas.
o = Índice para los trabajadores.
m = Índice para las máquinas.

2. Conjuntos

M = Conjunto de máquinas. $i \in M$.
O = Conjunto de trabajadores. $o \in O$.
T = Conjunto de tareas. $i \in T$.
T_o = Conjunto de tareas que realiza un trabajador o.
O_i = Conjunto de trabajadores que puede realizar una tarea i.
A = Conjunto de arcos (i, j).
Donde $G = (T, A)$ es un grafo y $(i, j) \in A$.

3. Parámetros

pt_i = Tiempo de procesamiento de la tarea i.
a_{io} = $\{$ 1, si el trabajador o puede realizar la tarea i / 0, en otro caso $\}$

4. Variables de Decisión

S_i = Tiempo de inicio de la tarea i.
C_i = Tiempo de finalización de la tarea i.
x_{io} = $\{$ 1, si se asigna la tarea i al trabajador o / 0, en otro caso. $\}$
y_{it} = $\{$ 1, si la tarea t inicia despues de la tarea i. / 0, en otro caso. $\}$

5. Función Objetivo

$$min\ C_{max} \tag{1}$$

6. Sujeto a:

$$C_{max} \geq C_i \qquad \forall\ i \in T \tag{2}$$

$$S_i + pt_i = C_i \qquad \forall\ i \in T \tag{3}$$

$$C_i \leq S_t + M(1 - y_{it}) \qquad \forall\ i,\ t \in T,\ m_i = m_t \tag{4}$$

$$C_t \leq S_i + My_{it} \quad \forall\ i,\ t \in T,\ m_i = m_t \tag{5}$$

$$C_i \leq S_i + M(1 - y_{it}) + M(2 - x_{io} - x_{to})\ \forall\ i,\ t \in T,\ \forall \in O \tag{6}$$

$$C_i \leq S_i + My_{it} + M(2 - x_{io} - x_{to})\ \forall\ i,\ t \in T,\ \forall \in O \tag{7}$$

$$S_i \geq C_i\ \forall\ (i,\ t) \in A \tag{8}$$

$$\sum_{i \notin T_o} x_{io} = 0 \qquad \forall\ o \in O \tag{9}$$

$$\sum_{o \in O_i} x_{io} = 1 \qquad \forall\ i \in T \tag{10}$$

$$S_i \geq 0 \quad \forall\ i \in T \tag{11}$$

$$x_{io} \in \{0,1\} \qquad \forall\ i \in T,\ \forall\ o \in O \tag{12}$$

Las ecuaciones (1) y (2) garantizan que el Makespan sea mayor al tiempo de terminación de cada tarea programada, esto es, minimizar el Makespan. La ecuación (3) asegura que al iniciar cada tarea esta debe terminarse sin demoras. Las ecuaciones (4) y (5) describen que dos tareas consecutivas que se realizan en una misma máquina no se traslapen. Las ecuaciones (6) y (7) garantizan que un trabajador no pueda estar realizando dos tareas al mismo tiempo. La ecuación (8) asegura que se cumplan las relaciones de precedencia. La ecuación (9) describe que un trabajador no puede ejecutar una tarea si no tiene la habilidad para hacerlo. La ecuación (10) asegura que se asigne una tarea a un operador. Las ecuaciones (11) a (12) aseguran el dominio de las variables.

El funcionamiento del modelo descrito anteriormente se puede observar analizando el siguiente ejemplo: Se cuenta con 11 tareas, 4 máquinas y 3 trabajadores. La Tabla 1 muestra la habilidad que tiene cada trabajador para realizar una tarea; la cual se representa por medio de una matriz binaria en donde el valor de 1 indica si el trabajador tiene la habilidad de realizar la tarea y 0 en caso contrario.

Tabla 1. Matriz de habilidades.

Trabajador	Tareas										
	1	2	3	4	5	6	7	8	9	10	11
1	1	1	0	1	0	1	1	0	1	0	1
2	0	1	0	0	1	1	0	1	1	0	1
3	1	1	1	1	0	1	0	0	1	1	0

Fuente: Autores.

La Tabla 2 muestra el tiempo que gasta cada tarea y la máquina en donde se debe realizar la misma. Las tareas cumplen relaciones de precedencia representadas en la Fig. 2.

Tabla 2. Tiempos y máquinas para la realización de cada tarea.

Tarea	Tiempo	Máquina
1	4	4
2	6	3
3	8	4
4	8	1
5	7	2
6	9	2
7	3	3
8	3	4
9	5	1
10	6	2
11	2	4

Fuente: Autores.

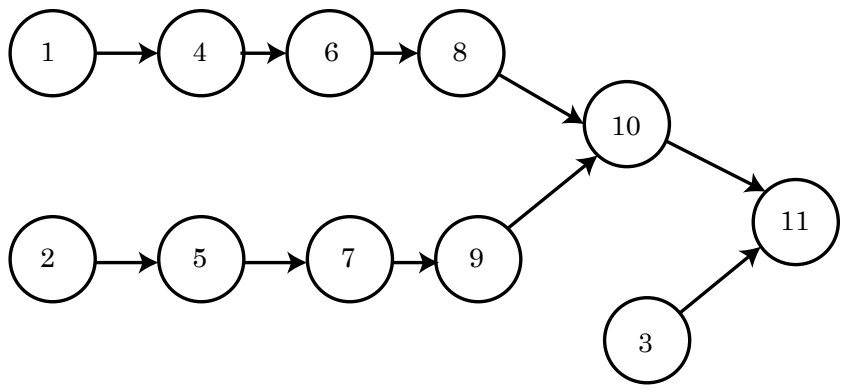

Fig. 2. Grafo de Precedencias.
Fuente: Autores.

El objetivo es asignar los operadores a las tareas, y las tareas a las máquinas, de tal manera que se minimice el Makespan cumpliendo con las restricciones (2) al (12). Al resolver el modelo matemático descrito se obtiene la siguiente solución mostrada en la Fig. 3:

Se obtiene una solución óptima con makespan de 33, garantizando el cumplimiento de todas las restricciones. Es de anotar la forma como se asignan los trabajadores a las tareas; para el caso de las tareas 1 y 2, tanto el trabajador 1 como el 3 tienen la habilidad para realizarlas. Al asignar el trabajador 3 a la tarea 1, se debe asignar el trabajador 2 a la tarea 2, puesto que al no hacerlo; la tarea 2 debe esperar que el trabajador 3 termine la tarea 1 para así poder iniciar la tarea 2, caso que obligaría a que la programación se demore más, puesto que esta tarea iniciaría al finalizar la tarea 1 y no al inicio de la programación como se muestra en la Fig. 3.

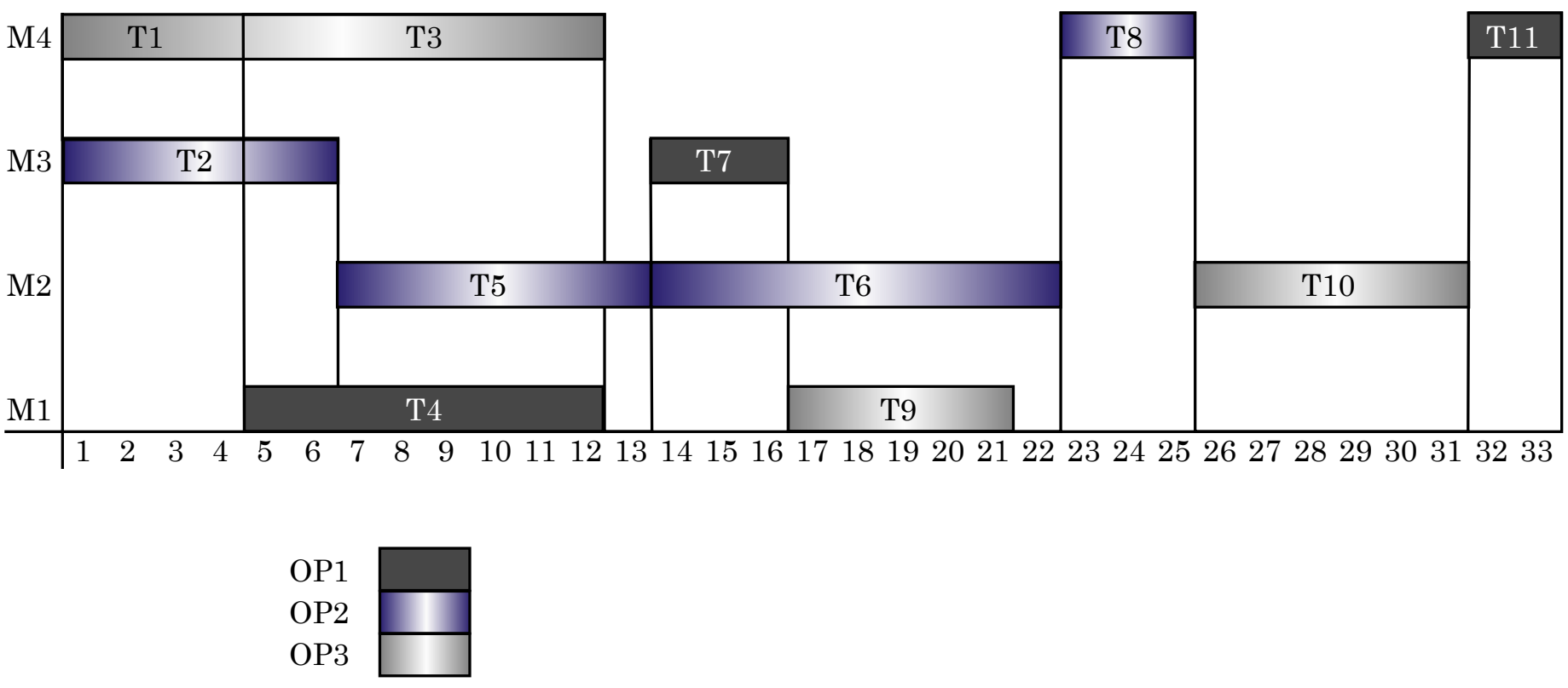

Fig. 3. Solución ejemplo.
Fuente: Autores.

Por otra parte, si se analizan las tareas 4 y 7, las cuales las puede realizar el trabajador 1, se muestra claramente cómo se garantiza el cumplimiento de las relaciones de precedencia, en el sentido de que la tarea 7 podría iniciarse a la finalización de la tarea 4 pero es infactible puesto que la tarea 5 no se ha terminado y es precedente a la tarea 7.

III. Squirrel Search Algorithm (SSA)

La metaheurística SSA es un método de solución basado en la dinámica de búsqueda y de un mecanismo eficiente de movimiento conocido como *"planeo"* de las ardillas voladoras presentes en el Sudeste asiático. Estas, inician el proceso de búsqueda de alimentos en la estación de otoño, explorando un bosque, planeando de un árbol a otro. Mientras hacen esta actividad, van cambiando su posición. Debido a las condiciones climáticas cálidas en esta época del año, las ardillas satisfacen sus requerimientos energéticos diarios consumiendo nueces de bellota, las cuales se presentan en gran cantidad. Hecho esto, las ardillas siguen buscando una fuente óptima de alimento para cuando llegue el invierno y esta se encuentra en los árboles que contienen nueces de nogal. Encontrar este tipo de árboles, mantiene sus requerimientos energéticos diarios en ambientes extremadamente fríos. Además, les permite aumentar sus probabilidades de supervivencia al disminuir sus salidas para buscar alimento. Sin embargo, con la llegada del invierno, las hojas de los árboles secan y caen, causando que las ardillas pierdan cobertura y exponiéndolas a posibles predadores. Esta condición obliga a las ardillas mermar su proceso de exploración, el cual se reactivará cuando termine el invierno.

Los siguientes supuestos facilitan el entendimiento del comportamiento descrito anteriormente:

1. Considere n como el número de ardillas presentes en el bosque, asuma que una ardilla estará ubicada en uno de estos.
2. Cada ardilla busca individualmente alimentos y utiliza de manera óptima los recursos alimenticios disponibles.
3. En el bosque, existen tres tipos de árboles: Árboles normales, árboles con nueces de bellota y árboles con nueces de nogal.
4. Se considera en el bosque a explorar, la existencia de tres árboles con nueces de bellota y un árbol con nueces de nogal.

A continuación, se explicará cómo es el funcionamiento del método de solución [3]:

A. Población Inicial Aleatoria

La posición de las ardillas en el bosque es representada por un vector de posición, el movimiento de las ardillas se puede dar en una dimensión, dos dimensiones, tres dimensiones o puede explorar en un espacio de búsqueda de más dimensiones. Por tanto, la población inicial contendrá los vectores de posición de cada una de las ardillas seleccionadas para iniciar el proceso de búsqueda. La población inicial se representa por la siguiente matriz mostrada en la ecuación 13:

$$FS = \begin{Bmatrix} FS_{1,1} & \cdots & FS_{1,d} \\ \vdots & \ddots & \vdots \\ FS_{n,1} & \cdots & FS_{n,d} \end{Bmatrix} \tag{13}$$

Donde $FS_{i,j}$ representa la dimensión d de la ardilla n.

Para generar la posición inicial para una ardilla se utiliza la ecuación 14:

$$FS_d = FS_L + U(0,1)\ x\ (FS_U - FS_L) \tag{14}$$

Donde FS_L y FS_U representan respectivamente las cotas inferior y superior de la posición de la ardilla y $U(0,1)$ es un numero aleatorio entre 0 y 1 distribuido de manera uniforme.

B. Evaluación de las soluciones

La posición de las ardillas en el espacio de búsqueda muestra que tan buena es con respecto a las demás. Cada posición debe generar un valor; en este caso, este es el de la función objetivo planteada.

C. Ordenamiento y ubicación aleatoria de la población inicial

Se ordena la población inicial en base al valor de la función objetivo almacenado. La ardilla con menor valor de función objetivo será asignada al árbol con nueces de nogal. Las tres ardillas siguientes serán asignadas a los árboles con nueces de bellota y las ardillas restantes estarán aleatoriamente asignadas en los árboles normales. Hecho esto, se considera que las ardillas ubicadas en los árboles con nueces de bellotas irán en dirección al árbol con nueces de nogal. De las ardillas restantes, una cantidad aleatorias de ellas se van a dirigir desde los árboles normales hacia el árbol con nueces de nogal, las restantes se van a mover hacia los árboles con nueces de bellotas.

D. Movimientos de las ardillas

Las ardillas exploran el espacio de solución (bosque) obedeciendo los movimientos descritos anteriormente. Cuando existe la presencia de una amenaza, las ardillas se mueven cautelosamente, dándole la capacidad de explorar el espacio solución. Los movimientos se producen según los casos siguientes:

Caso I

Las ardillas que están ubicadas en los árboles con nueces de bellota se mueven hacia la ardilla ubicada en el árbol con nueces de nogal, el movimiento se representa por medio de la ecuación 15:

$$FS_{ab}^{t+1} = \underbrace{\{FS_{ab}^{t} + d_g \text{ x } G_c (FS_{ao}^{t} - FS_{ab}^{t}), \text{ si } R_1 \geq P_{dp}}_{\textit{Posición Aleatoria}} \quad (15)$$

Donde d_g es la distancia aleatoria de planeo, R_1 es un número aleatorio entre 0 y 1 distribuido de manera uniforme. t denota la iteración actual FS_{ao}^{t} es la ardilla ubicada en el árbol con nueces de nogal, FS_{ab}^{t} es la ardilla ubicada en el árbol con nueces de bellota, G_c es la constante de planeo, la cual permite el balance entre la exploración e intensificación del algoritmo y P_{dp} es la probabilidad de aparición de un predador.

Caso II

Las ardillas ubicadas en los árboles normales se mueven hacia los árboles con nueces de bellota para satisfacer sus requerimientos diarios de energía (16):

$$FS_{anor}^{t+1} = \underbrace{\{FS_{anor}^{t} + d_g \text{ x } G_c (FS_{ab}^{t} - FS_{anor}^{t}), \text{ si } R_2 \geq P_{dp}}_{\textit{Posición Aleatoria}} \quad (16)$$

Donde R_2 es un número aleatorio entre 0 y 1 distribuido de manera uniforme. FS_{ab}^{t} y FS_{anor}^{t} son las posiciones de actuales de las ardillas ubicadas en los árboles con nueces de bellota y arboles normales respectivamente.

Caso III

Las ardillas que están ubicadas en los árboles normales se mueven hacia el árbol con nueces de nogal (17).

$$FS_{anor}^{t+1} = \underbrace{\{FS_{anor}^{t} + d_g \text{ x } G_c (FS_{ao}^{t} - FS_{anor}^{t}), \text{ si } R_3 \geq P_{dp}}_{\textit{Posición Aleatoria}} \quad (17)$$

R_2 es un número aleatorio entre 0 y 1 distribuido de manera uniforme. FS_{ao}^{t} es la posición actual de la ardilla ubicada en el árbol con nueces de nogal y FS_{anor}^{t} es la posición de la ardilla ubicada en los árboles normales.

E. Condición de monitoreo estacional

Debido a la fisionomía de las ardillas, en temperaturas bajas (invierno) pierden una significante cantidad de calor corporal, disminuyendo su proceso de búsqueda, además de colocarlas en riesgo de ser cazadas por un depredador. Por tanto, los cambios climáticos influyen el proceso de búsqueda de las ardillas. Para esto, se introduce una condición de monitoreo estacional, y evitar que el algoritmo quede atrapado en óptimos locales.

Se calcula la constante estacional S_c:

$$S_c^t = \sqrt{\sum_{k=1}^{d} (FS_{ab,\,k}^{t} - FS_{ao,\,k})^2} \quad (18)$$

Con $t = 1, 2, 3$.

Compruebe la condición de monitoreo estacional de esta forma $S_c^t \leq S_{min}$, donde S_{min} se calcula de la siguiente forma:

$$S_{min} = \frac{10E^{-6}}{(365)^{t/(t_m/2.5)}} \quad (19)$$

Donde t y t_m representa el valor de la iteración actual y el número máximo de iteraciones respectivamente. Si el criterio de monitoreo de estacionalidad se cumple, indica que el invierno ha terminado y por lo tanto, se mueven aleatoriamente a otro lugar del bosque para seguir con la búsqueda de fuentes de alimentos.

F. Movimiento aleatorio al final del invierno

En este punto, solo las ardillas que no pudieron dirigirse hacia el árbol con nueces de nogal se dirigirán a nuevas posiciones. Estos movimientos son modelados bajo la ecuación 20:

$$FS_{anor}^{New} = FS_L + Levy(n)\ x\ (FS_U - FS_L) \quad (20)$$

IV. SSA aplicado al problema JSSOP

A. Codificación de la solución

La codificación propuesta consiste en una matriz de tres filas por $n + 1$ columnas; donde es el número total de trabajos (Fig. 5).

La primera fila corresponde a números aleatorios generados uniformemente en un rango [-5, 5] cumpliendo la ecuación (14). La segunda fila corresponde a la secuencia de los trabajos a realizar; esta secuencia se obtiene utilizando el método Smallest Position Value (SPV), el cual ha sido ampliamente utilizado como generador de secuencias cuando se están trabajando con metaheurísticas basadas en trayectorias [10], [18], [19], [20]. Al aplicar este método es posible que algunas soluciones resulten infactibles, puesto que violan las restricciones de precedencia. Para evitar esto, se utiliza el método Valid Particle Generator (VPG) [21] para corregir este problema y obtener siempre soluciones factibles.

-2.78	-4.75	-2.92	-2.84	-0.56	-3.65	-1.09	-1.30	1.68	4.92	-3.82
1	2	3	4	5	6	7	8	9	10	11

Al aplicar SPV

-4.75	-3.82	-3.65	-292	-2.84	-2.78	-1.30	-1.09	-0.56	1.68	4.92
2	11	6	3	4	1	8	7	5	9	10

Al aplicar VPG

-4.75	-2.78	-2.92	-2.84	-3.65	-0.56	-1.30	-1.09	1.68	4.92	-3.82
2	1	3	4	6	5	8	7	9	10	11

Fig. 4. Proceso de generación de soluciones factibles.
Fuente: Autores.

Se puede detallar en la Fig. 4 el proceso descrito anteriormente. Aplicar el método SPV consiste en ordenar la primera fila de menor a mayor, de tal manera que los valores que estaban en la segunda columna pasarían a ser la primera, debido a que es la que tiene el menor valor de todos, de esta manera se organiza el resto del arreglo obteniendo una nueva solución. Esta solución sería factible si no existieran relaciones de precedencia entre las tareas. Ahora bien, la tarea 2 que se ubica en la primera posición no presenta inconveniente, debido a que es una de las tareas que no presenta predecesoras. Por su parte, la tarea 11 ubicada en la segunda posición necesita de la terminación de las tareas 3 y 10, las cuales se encuentran en posiciones posteriores en el arreglo generando que la solución encontrada sea infactible. Para evitar esto, se propone corregir la solución utilizando el método VPG, el cual analiza de manera detallada el grafo de precedencias detectando y corrigiendo cuando se puede incurrir en una violación de las relaciones. En este caso, se puede detallar que al aplicar este método las tareas 2, 1 y 3 se ubican en las primeras posiciones ya que no necesitan de tareas precedentes. Las tareas 4 y 6 aparecen en las siguientes posiciones del arreglo debido a que la tarea 1 ya se encuentra definida anteriormente. La tarea 5 aparece en la siguiente posición sin inconvenientes, porque ya la tarea 1 que era su predecesora se encuentra definida. De esta manera este método garantiza corregir cualquier solución obtenida al aplicar el método SPV garantizando que se trabajen solo con soluciones factibles.

Por último, la tercera fila corresponde a la asignación de los operadores a cada uno de los trabajos de la secuencia. La asignación de operadores a los trabajos se realiza de dos formas; la primera una asignación aleatoria, en la cual se identifica que operador(es) pueden realizar el trabajo y se asigna sin ningún criterio de decisión. Por otra parte, se utiliza la heurística Lowest First Decreasing (LFD) [22], la cual consiste en ordenar las tareas de la más larga a las más corta en duración y se asigna sucesivamente a un operador que pueda realizarla, teniendo en cuenta que esté disponible al momento de realizar esta asignación.

-4.75	-2.78	-2.92	-2.84	-3.65	-0.56	-1.30	-1.09	1.68	4.92	-3.82
2	1	3	4	6	5	8	7	9	10	11
2	3	2	3	1	2	2	1	3	3	1

Fig. 5. Codificación propuesta.
Fuente: Autores.

Como resultado, se obtienen dos variantes del método de solución: SSA1 para el caso de asignación bajo la heurística LFD y SSA2 para el caso de asignación aleatoria.

B. Población Inicial

La población inicial está conformada por un conjunto finito de individuos (ardillas). Dado que cada individuo está conformado por una matriz de tamaño de tres filas por columnas, el tamaño total de la población es definido por una matriz donde el número de filas es equivalente al producto del número total de ardillas por tres y el número de columnas sigue siendo $n + 1$. Esto, si se decide que el tamaño de la población sea de 100 individuos. entonces, se obtiene una matriz de población inicial de 300 filas y $n + 1$ columnas.

Al conformar la población inicial, se debe ordenar de manera creciente la matriz, tomando como referencia la columna $n + 1$, la cual contiene el valor del criterio de decisión a optimizar; en este caso, el mayor tiempo de terminación de los trabajos conocido como Lapso o Makespan.

En este punto, se define que el individuo con mejor valor de Makespan (Mínimo valor) va a ser ubicado en el árbol con nueces de nogal, los tres siguientes individuos en los árboles con nueces de bellota y el resto de las ardillas se ubican aleatoriamente en los árboles normales.

C. Movimientos de las ardillas

Los movimientos de las ardillas se enmarcan en los tres casos descritos anteriormente. En primer lugar, se genera un número aleatorio entre 0 y 1 distribuido de manera uniforme, si este número es mayor a la probabilidad de ser detectado por un depredador P_{dp}, la ardilla realiza su movimiento conforme a la ecuación (15). Si este número es menor a P_{dp}, la ardilla se mueve a una posición aleatoria siguiendo la ecuación (14).

Este proceso se repite con las demás ardillas ubicadas en los árboles restantes, sus movimientos serán conforme a las ecuaciones (16) y (17) según sea el caso.

D. Monitoreo de la estacionalidad

Se sigue el proceso de optimización evaluando la condición de monitoreo de la estacionalidad, para tal fin se calcula los valores de S_c^t y S_{min} con las ecuaciones (18) y (19), si se cumple el criterio $S_c^t \leq S_{min}$, indica que el invierno ha terminado y las ardillas se ubican en posiciones aleatorias conforme a la ecuación (20).

E. Criterio de parada

El proceso de búsqueda se repite un número determinado de iteraciones, cuando se alcanza este número de iteraciones el proceso de búsqueda termina y arroja la ardilla ubicada en el árbol con nuez de nogal, siendo en realidad la mejor solución obtenida en el proceso.

V. Experimentos Computacionales

Como se mencionó anteriormente, se proponen dos versiones de la metaheurísticas denominadas SSA1 y SSA2. Se escogieron 28 instancias propuestas [1] para medir los desempeños de estos y se identificaron dado el número de operadores P, el número de máquinas Q y el número de tareas T. En la Tabla 3 se resumen las características de las instancias analizadas:

Tabla 3. Grupo de instancias estudiadas.

Grupo Instancias	P	Q	T	Cantidad
1	10	15	100	3
2	10	20	200	4
3	15	30	150	4
4	15	30	200	4
5	15	30	250	1
6	15	50	150	4
7	20	30	200	4
8	20	30	250	4

Fuente: Autores.

Se realizaron experimentaciones previas para calibrar los niveles de los parámetros de las dos versiones del algoritmo propuesto en este trabajo. Se identificaron cuatro parámetros: Probabilidad de una ardilla ser detectada por un depredador P_{dp}, la constante de planeo G_c, el número de ardillas que van de los árboles normales hacia los árboles con nueces de nogal n_2 y el número de ardillas para la población inicial T_{am}. Para cada parámetro descrito, se estableció un nivel central y se varió con un delta hacia arriba y hacia abajo para obtener tres niveles (Tabla 4).

Tabla 4. Diseño factorial para la parametrización de los algoritmos.

Parámetro	Nivel inferior	Nivel central	Nivel superior	Δ
P_{dp}	0.5	0.6	0.7	± 0.1
G_c	1.6	1.9	2.2	± 0.3
N_2	0.4	0.5	0.6	± 0.1
T_{am}	2000	3500	5000	± 1500

Fuente: Autores.

De esta manera se configuró un experimento donde se replicó dos veces el total de combinaciones de los niveles escogidos. En total se obtuvieron 162 condiciones experimentales. Al finalizar las réplicas, se promediaron los resultados de la variables de respuesta obtenidos, en este caso, el tiempo total de terminación de los trabajos o Makespan, escogiendo como mejor resultado, aquella combinaciones de niveles que arrojaron menor promedio. Para esto se ejecutaron durante 24 horas en una misma instancia escogida al azar. Esto se realizó con el fin de que los algoritmos tengan las mismas condiciones para poder resolver instancias de gran tamaño.

Después de realizar estos experimentos, los niveles que se escogieron para cada parámetro de cada algoritmo se muestran en la Tabla 5:

Tabla 5. Parámetros Escogidos.

Versión	P_{dp}	G_c	N_2	Tam
SSA1	0.7	2.2	0.6	5000
SSA2	0.7	2.2	0.6	5000

Fuente: Autores.

En este punto, se resolvieron las 28 instancias propuestas, y para cada instancia se realizaron tres réplicas escogiendo como solución final la menor de estas tres. Los algoritmos fueron codificados bajo el lenguaje R versión 3.5.3 “Great Truth” utilizando R Studio versión 1.1.383 [23] y los experimentos se ejecutaron en un computador Dell Intel® Core™ i5-6500, CPU de 3.20 GHz a 3.19 GHz con memoria RAM de 8 GB.

VI. Resultados

Los resultados obtenidos se resumen en la Tabla 6, donde se detallan las soluciones para los dos métodos propuestos SSA1 y SSA2 además de las soluciones óptimas reportadas [1]. En este orden de ideas, se puede observar que ambos métodos propuestos encontraron soluciones óptimas. Para el método SSA1 se encontraron en 24 instancias soluciones óptimas, alcanzado un rendimiento del aproximadamente del 85%. Para el método SS2 se encontraron en 25 instancias soluciones optimas, lo que equivale a un rendimiento del 89%.

Ahora bien, la Tabla 7 muestra los óptimos alcanzados por cada grupo de instancias analizados. Es válido realizar algunos comentarios en base a estos resultados; por ejemplo, el número de instancias que conforman al grupo 1 es de 3; de este número, los métodos propuestos alcanzan dos soluciones óptimas. Si se realiza el mismo análisis para los demás grupos, se puede apreciar un apropiado rendimiento de los métodos propuestos al problema estudiado. Cabe resaltar, que en el grupo de instancias 5 solo se evaluó un problema de este tipo y particularmente para ese problema, los métodos propuestos no alcanzaron una solución óptima, esta es la razón por la cual aparece en la tabla de análisis con un valor de cero.

Tabla 6. Resultados obtenidos.

Grupo Instancia	*ID Instancia*	*SSA1*	*SSA2*	*Cplex*	*GAP_ SSA1*	*GAP_ SSA2*
1	1	3057	3041	2995	0,0203	0,0151
	2	2896	2896	2896	0,0000	0,0000
	3	4256	4256	4256	0,0000	0,0000
2	4	1788	1781	1781	0,0039	0,0000
	5	3502	3502	3502	0,0000	0,0000
	6	2429	2429	2429	0,0000	0,0000
	7	5697	5697	5697	0,0000	0,0000
3	8	5664	5664	5664	0,0000	0,0000
	9	4271	4271	4271	0,0000	0,0000
	10	4010	4010	4010	0,0000	0,0000
	11	4093	4093	4093	0,0000	0,0000
4	12	9132	9132	9132	0,0000	0,0000
	13	5132	5132	5132	0,0000	0,0000
	14	7973	7973	7973	0,0000	0,0000
	15	8645	8645	8645	0,0000	0,0000
5	16	11465	11465	11422	0,0038	0,0038
6	17	4841	4841	4841	0,0000	0,0000
	18	4983	4983	4983	0,0000	0,0000
	19	2951	2951	2951	0,0000	0,0000
	20	4193	4193	4193	0,0000	0,0000
7	21	7488	7488	7488	0,0000	0,0000
	22	6006	6006	6006	0,0000	0,0000
	23	5403	5403	5394	0,0017	0,0017
	24	8662	8662	8662	0,0000	0,0000
8	25	10262	10262	10262	0,0000	0,0000
	26	11547	11547	11547	0,0000	0,0000
	27	11760	11760	11760	0,0000	0,0000
	28	11862	11862	11862	0,0000	0,0000

Fuente: Autores.

Tabla 7. Resultados obtenidos.

	Óptimos Encontrados	
Grupo Instancias	SSA1	SSA2
1	2	2
2	3	4
3	4	4
4	4	4
5	0	0
6	4	4
7	3	3
8	4	4
Total	24	25

Fuente: Autores.

Por otra parte, es preciso analizar para aquellos problemas en donde no se alcanzaron soluciones óptimas, cual es la desviación de estas soluciones con respecto a las soluciones óptimas reportadas en la literatura. Como se puede apreciar en la Fig. 6, se pueden analizar los valores de esta desviación (Gap) para cada uno de los métodos propuestos. Para el método SSA1, se tiene que no se alcanzan soluciones óptimas en 4 instancias, los valores obtenidos presentan un gap de 0.0203, 0.0039, 0.0038 y 0.0017 respectivamente, lo que representan un gap promedio de 0.74%. Por su parte, el método SSA2 no alcanza soluciones óptimas en 3 instancias, los valores obtenidos presentan un gap de 0.0151, 0.0038 y 0.0017 respectivamente, lo que genera un gap promedio de 0.68%, por lo que se puede inferir que ambos métodos presentan un rendimiento aceptable al tener un gap promedio inferior al 1%.

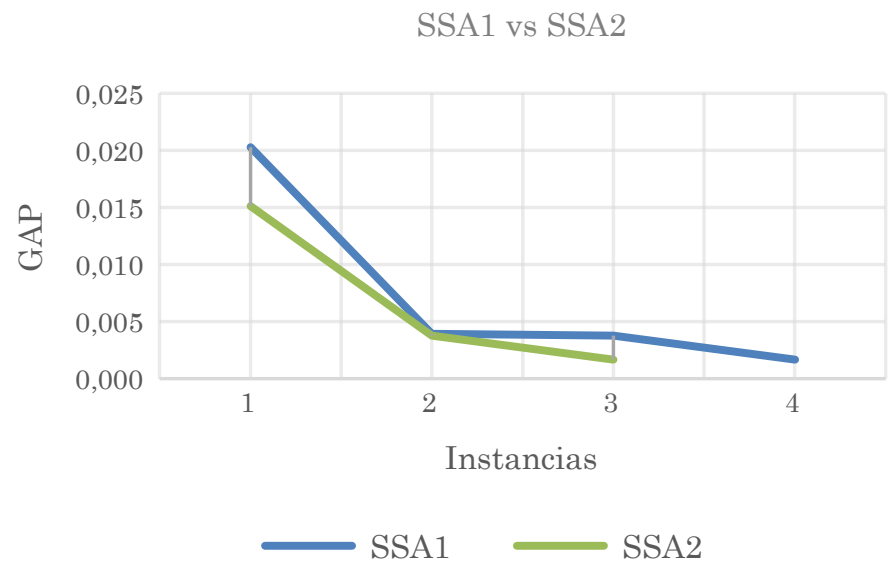

Fig. 6. Comparación entre los métodos propuestos.
Fuente: Autores.

V. Conclusiones

Se propuso en este artículo, adaptar e implementar un algoritmo basado en inteligencia de enjambres conocido como Squirrel Search Algorithm para solucionar el problema Job Shop con Operadores Calificados. El esquema de codificación propuesto garantiza la discretización del algoritmo. Al utilizar el método SPV para obtener la discretización de la solución, y luego realizar la corrección con el método VPG, se logran obtener soluciones factibles de alta calidad permitiendo al algoritmo converger hacia la solución óptima. Además, la codificación propuesta, permite utilizar de manera natural los operadores de movimiento propuestos originalmente para el algoritmo utilizado, razón por la cual, se considera que esta codificación es uno de los aportes más significativos de este estudio.

Por otra parte, por medio de las experimentaciones realizadas, se logró identificar los niveles de los parámetros adecuados para el funcionamiento del algoritmo, en su adaptación al problema. Sumado a esto, se pudo demostrar por medio de los experimentos computacionales; que las versiones del algoritmo propuestas en este estudio alcanzaron resultados satisfactorios.

Se propone como líneas de trabajo futuros, abordar el mismo problema estudiado y realizar comparaciones de este algoritmo frente a otros algoritmos basados en inteligencia de enjambres tales como PSO, Cuckoo Search, Algoritmos Meméticos y Colonia de abejas, entre otros. Por otra parte, se sugiere que este algoritmo también pueda ser adaptado a otros problemas discretos de optimización utilizando métodos de discretización tales como Sigmoid Function (SF) o Random Key (RK).

Agradecimientos

Este proyecto se realizó gracias al apoyo de la Facultad De Ingeniería Industrial perteneciente a la escuela de Arquitectura e Ingeniería de la Universidad Pontifica Bolivariana Seccional Montería.

Referencias

[1] A. Agnetis, G. Murgia and S. Sbrilli, "A job shop scheduling problem with human operators in handicraft production," *Int. J. Prod. Res.*, vol. 52, no. 13, pp. 3820–3831, Jul. 2014. https://doi.org/10.1080/00207543.2013.831220

[2] M. Abdel-Basset, L. Abdel-Fatah and A. K. Sangaiah, "Metaheuristic Algorithms: A Comprehensive Review," in *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*. Cambridge, MA, USA: Elsevier Inc., 2018, pp. 185–231. https://doi.org/10.1016/B978-0-12-813314-9.00010-4

[3] M. Jain, V. Singh and A. Rani, "A novel nature-inspired algorithm for optimization: Squirrel search algorithm," *Swarm Evol. Comput.*, vol. 44, pp. 148–175, Feb. 2019. https://doi.org/10.1016/j.swevo.2018.02.013

[4] L. Gao, X. Li, X. Wen, C. Lu and F. Wen, "A hybrid algorithm based on a new neighborhood structure evaluation method for job shop scheduling problem," *Comput. Ind. Eng.*, vol. 88, pp. 417–429, Oct. 2015. https://doi.org/10.1016/j.cie.2015.08.002

[5] K. Z. Gao, P. N. Suganthan, T. J. Chua, C. S. Chong, T. X. Cai and Q. K. Pan, "A two-stage artificial bee colony algorithm scheduling flexible job-shop scheduling problem with new job insertion," Expert Syst. Appl., vol. 42, no. 21, pp. 7652–7663, Nov. 2015. https://doi.org/10.1016/j.eswa.2015.06.004

[6] K. Z. Gao, P. N. Suganthan, Q. K. Pan, T. J. Chua, C. S. Chong and T. X. Cai, "An improved artificial bee colony algorithm for flexible job-shop scheduling problem with fuzzy processing time," *Expert Syst. Appl.*, vol. 65, pp. 52–67, Dec. 2016. https://doi.org/10.1016/j.eswa.2016.07.046

[7] A. Maroosi, R. C. Muniyandi, E. Sundararajan and A. M. Zin, "A parallel membrane inspired harmony search for optimization problems: A case study based on a flexible job shop scheduling problem," *Appl. Soft Comput. J.*, vol. 49, pp. 120–136, Dec. 2016. https://doi.org/10.1016/j.asoc.2016.08.007

[8] A. Ahmadi-Javid and P. Hooshangi-Tabrizi, "Integrating employee timetabling with scheduling of machines and transporters in a job-shop environment: A mathematical formulation and an Anarchic Society Optimization algorithm," *Comput. Oper. Res.*, vol. 84, pp. 73–91, Aug. 2017. https://doi.org/10.1016/j.cor.2016.11.017

[9] H. Piroozfard, K. Y. Wong and A. D. Asl, "An improved biogeography-based optimization for achieving optimal job shop scheduling solutions," *Procedia Comput. Sci.*, vol. 115, pp. 30–38, Aug. 2017. https://doi.org/10.1016/j.procs.2017.09.073

[10] N. Sharma, H. Sharma and A. Sharma, "Beer froth artificial bee colony algorithm for job-shop scheduling problem," *Appl. Soft Comput. J.*, vol. 68, pp. 507–524, Jul. 2018. https://doi.org/10.1016/j.asoc.2018.04.001

[11] B. Marzouki, O. B. Driss and K. Ghédira, "Solving Distributed and Flexible Job shop Scheduling Problem using a Chemical Reaction Optimization metaheuristic," *Procedia Comput. Sci.*, vol. 126, pp. 1424–1433, Jan. 2018. https://doi.org/10.1016/j.procs.2018.08.114

[12] A. Agnetis, M. Flamini, G. Nicosia and A. Pacifici, "A job-shop problem with one additional resource type," *J. Sched.*, vol. 14, no. 3, pp. 225–237, Jun. 2011. https://doi.org/10.1007/s10951-010-0162-4

[13] M. R. Sierra, C. Mencía and R. Varela, "New schedule generation schemes for the job-shop problem with operators," *J. Intell. Manuf.*, vol. 26, no. 3, pp. 511–525, Jul. 2013. https://doi.org/10.1007/s10845-013-0810-6

[14] B. Giffler and G. L. Thompson, "Algorithms for Solving Production-Scheduling Problems," *Oper. Res.*, vol. 8, no. 4, pp. 487–503, Aug. 1960. https://doi.org/10.1287/opre.8.4.487

[15] R. Mencia, M. R. Sierra, C. Mencia and R. Varela, "Genetic Algorithm for Job-Shop Scheduling with Operators," in New Challenges on Bioinspired Applications, *4th International Work-conference on the Interplay Between Natural and Artificial Computation*, IWINAC 2011, La Palma, Canary Islands, Spn, 30 May. - 3 Jun. 2011, pp. 305–314. https://doi.org/10.1007/978-3-642-21326-7

[16] R. Mencía, M. R. Sierra, C. Mencía and R. Varela, "A genetic algorithm for job-shop scheduling with operators enhanced by weak Lamarckian evolution and search space narrowing," *Nat. Comput.*, vol. 13, no. 2, pp. 179–192, May. 2013. https://doi.org/10.1007/s11047-013-9373-x

[17] F. Barber, J. Escamilla, C. Mencia, M. Rodriguez-Molins, M. A. Salido and M. R. Sierra, "Robust solutions to job-shop scheduling problems with operators," *Proc. - Int. Conf. Tools with Artif. Intell.* ICTAI, Athens, Greece, 7-9 Nov. 2012. https://doi.org/10.1109/ICTAI.2012.48

[18] M. F. Tasgetiren, M. Sevkli, Y.-Ch. Liang and G. Gencyilmaz, "Particle swarm optimization algorithm for single machine total weighted tardiness problem," *Proceedings of the 2004 Congress on Evolutionary Computation* (IEEE Cat. No. 04TH8753), Portland, OR, USA, USA, 19-23 Jun. 2004, pp. 1412–1419. https://doi.org/10.1109/CEC.2004.1331062

[19] R. Chaudhry, S. Tapaswi and N. Kumar, "Forwarding Zone enabled PSO routing with Network lifetime maximization in MANET," *Appl. Intell.*, vol. 48, no. 9, pp. 3053–3080, Sept. 2018. https://doi.org/10.1007/s10489-017-1127-5

[20] I. Dubey and M. Gupta, "Uniform mutation and SPV rule based optimized PSO algorithm for TSP problem," *Proc. 2017 4th Int. Conf. Electron. Commun. Syst.*, ICECS 2017, Coimbatore, India, 24-25 Feb. 2017, pp. 168–172. https://doi.org/10.1109/ECS.2017.8067862

[21] N. Kumar and D. P. Vidyarthi, "A model for resource-constrained project scheduling using adaptive PSO," *Soft Comput.*, vol. 20, no. 4, pp. 1565–1580, Feb. 2015. https://doi.org/10.1007/s00500-015-1606-8

[22] R. L. Graham, E. L. Lawler, J. K. Lenstra and A. H. G. R. Kan, "Optimization and approximation in deterministic machine scheduling: a survey," *Ann. Discret. Math.*, vol. 5, pp. 287–326, 1979. https://doi.org/10.1016/S0167-5060(08)70356-X

[23] R Core Team, "R: A language and environment for statistical computing" *R Foundation for Statistical Computing*, 2019. Disponible: https://www.r-project.org/

César Andrés López Martínez es Ingeniero Industrial de la Universidad de Córdoba (Montería, Colombia). Es candidato a Magister en Ingeniería de Producción de la Universidad Tecnológica de Bolívar (Cartagena de Indias, Colombia). Sus intereses investigativos incluyen tópicos en Metaheurísticas, Investigación de Operaciones, Programación de Producción, Optimización. Actualmente es Docente interno en la facultad de Ingeniería Industrial de la Universidad Pontificia Bolivariana (Montería, Colombia). https://orcid.org/0000-0003-2750-7699

Helman Enrique Hernández Riaño es Ingeniero Industrial de la Universidad Distrital (Bogotá, D. C., Colombia). Es Magister en Gestión de Organizaciones de la Universidad EAN (Bogotá, D.C., Colombia). Doctor en Ingeniería Industrial de la Universidad del Norte (Barranquilla, Colombia). Sus intereses investigativos incluyen tópicos en Metaheurísticas, Investigación de Operaciones, Programación de Producción, Logística y Optimización. Actualmente es Profesor Titular del Departamento de Ingeniería Industrial de la Universidad de Córdoba (Montería, Colombia). https://orcid.org/0000-0003-3042-2573

Manuel Jesús Soto de la Vega es Ingeniero Industrial de la Universidad de Córdoba (Colombia). Magister en Ingeniería Industrial de la Universidad de los Andes (Bogotá, D.C., Colombia). Actualmente profesor de la Universidad Tecnológica de Bolívar (Colombia) vinculado al programa de Ingeniería Industrial. Áreas de trabajo en investigación de operaciones, incluyendo trabajos en optimización y simulación de procesos.