Un modelo híbrido de inteligencia computacional para resolver el problema de Job Shop Scheduling

Jacob Meneses Angel, Marcela Rivera Martínez, Luis René Marcial Castillo, Lourdes Sandoval Solís

Benemérita Universidad Autónoma de Puebla, Puebla, Pue., México
{snake_n313, mrivmar}@hotmail.com
{lmc, sandoval}@cs.buap.mx

Resumen. En este trabajo se presenta un modelo híbrido de inteligencia computacional para resolver el problema de job shop scheduling, problema de secuenciación, clasificado como NP Completo. Se propone resolverlo usando la técnica de colonia de hormigas asistida con recocido simulado. Como estrategia de búsqueda global, se usó la técnica de colonia de hormigas y como estrategia de búsqueda local, la de recocido simulado. Dicha propuesta se validó experimentalmente con problemas prueba reportados en la literatura.

Palabras clave: Inteligencia Computacional, Job Shop Scheduling, Colonia de Hormigas, Recocido Simulado, Secuenciación, Makespan.

1 Introducción

El problema de *job shop scheduling* (JSS), es un problema de optimización combinatoria muy conocido en el área de investigación de operaciones e inteligencia artificial. En la manufactura de un producto se requiere de una serie de procesos, cada uno de éstos se efectúa en una máquina determinada. Esto origina diferentes problemas entre los que se encuentra secuenciar *n* trabajos en *m* máquinas. La secuenciación tiene como objetivo encontrar el orden de ejecución de *n* trabajos, que requieren una serie de procesos en *m* máquinas, de manera que se optimice el tiempo [14]. Acorde a la teoría de complejidad computacional, es un problema NP Completo, por lo tanto no puede resolverse usando métodos exactos [7]. En este trabajo se propone el uso de la metaheurística colonia de hormigas (Ant Colony Optimization, ACO) asistido con recocido simulado (Simulated Annealing, SA).

En los últimos años, de acuerdo con la literatura publicada, se han propuesto varios algoritmos híbridos para solucionar este problema en [9, 11, 15], pero no hay antecedentes de propuestas hibridas de colonia de hormigas con recocido simulado.

En la sección 2, se presentan los conceptos básicos del problema de *job shop scheduling*, la sección 3 describe el algoritmo de recocido simulado; el algoritmo de colonia de hormigas se describe en la sección 4, en la sección 5 se presenta el algoritmo propuesto en el presente trabajo, en la sección 6 se muestran las pruebas del algoritmo propuesto a problemas reportados en la literatura, en la sección 7 se mencionan las conclusiones del trabajo.

2 Conceptos básicos

En el esquema de *job shop scheduling* cada trabajo tiene su propia ruta predeterminada a seguir. Las prioridades dependen de cada trabajo debido a diferentes factores como: mínima manipulación del material, fácil transporte, no existe regreso, manufacturas modernas. En el *job shop scheduling* las operaciones consecutivas de un trabajo nunca se llevan a cabo en la misma máquina, cuando esto ocurre, las dos operaciones se consideran como una sola cuyo tiempo de procesamiento es la suma de los tiempos de procesamiento de las operaciones individuales. Cada trabajo consta de un conjunto de operaciones con una secuencia específica, las cuales requieren de una máquina para su procesamiento y cada máquina puede realizar sólo una tarea a la vez.

Formalmente, un problema de *job shop scheduling* se define como un conjunto finito J de n trabajos $J = \{J_1, \ldots\ldots J_n\}$ que tienen que ser programados en un conjunto finito M de m máquinas, $M = \{M_1, \ldots\ldots, M_m\}$. Cada trabajo J_i está compuesto de una serie de k operaciones O_{ik}, donde el subíndice k indica la máquina M_k en que las operaciones se deben procesar. El orden de las máquinas (secuenciación) para un trabajo J_i está predefinido; a cada operación O_{ik} se le asigna un tiempo de procesamiento entero no negativo P_{ik}. El objetivo es determinar la programación de una secuencia de operaciones en cada máquina M_k tal que las restricciones de precedencia y de capacidad del problema sean satisfechas [6].

Las permutaciones de operaciones representan una solución al problema, aunque no son prácticas con respecto al post procesado. Además de la posición dentro de la secuencia, se está interesado principalmente en que las operaciones de un mismo trabajo no se lleven a cabo al mismo tiempo en una misma máquina. Por esta razón, un candidato solución se describe preferentemente como un conjunto S de tiempos iniciales de operación S_{ik}, $S = \{S_{ik} \mid 1 \leq i \leq n, 1 \leq k \leq m\}$.

Por otra parte, la inteligencia computacional se ocupa de la teoría, diseño, desarrollo y aplicaciones de paradigmas computacionales motivados lingüística y biológicamente. La inteligencia computacional aplicada es un sistema de métodos e infraestructuras que mejora la inteligencia humana aprendiendo y descubriendo nuevos patrones, relaciones y estructuras complejas en ambientes dinámicos para resolver problemas prácticos [13].

3 Recocido simulado

En la metalurgia, el método del *recocido* se utiliza para obtener materiales más resistentes o más cristalinos, en general, para determinar las cualidades de un material. El proceso consiste en calentar el material a temperatura muy alta, en esa situación los átomos adquieren una distribución *azarosa* dentro de la estructura del material y la energía del sistema es máxima.

Después se hace descender la temperatura muy lentamente por etapas, de acuerdo con una calendarización, dejando que en cada una de esas etapas los átomos queden en equilibrio, es decir, que los átomos alcancen una configuración óptima para esa temperatura. Al final del proceso, los átomos forman una estructura cristalina altamente regular, el material alcanza las cualidades buscadas y la energía del sistema es mínima.

Experimentalmente se comprueba que si se hace descender la temperatura bruscamente, o si no se espera suficiente tiempo en cada etapa, al final, la estructura del material no es la óptima. El algoritmo de recocido simulado se desarrolló para simular el proceso de recocido con el fin de encontrar un mínimo global de la función objetivo. En el algoritmo de recocido simulado, la función objetivo se trata como la función de energía de un metal fundido y una calendarización de temperaturas artificiales se establece para enfriar gradualmente el material, análogo a la técnica de recocido. Esta temperatura artificial o conjunto de temperaturas actúa como una fuente de aleatoriedad, que es conveniente para evitar eventualmente un mínimo local [10].

El algoritmo requiere de una configuración inicial dentro del espacio de soluciones o conjunto de configuraciones R, una función de costo $C: R \rightarrow R$, una estructura de vecindad $\hat{V}: R \rightarrow R^2$, y $\hat{V}(r)$ como un subconjunto de $\hat{V}$, vecinos de la configuración $r, \forall\, r \in R$.

El algoritmo se define de la siguiente manera: Dada una configuración inicial X, y un número de iteraciones, en cada iteración se selecciona $Y \in \hat{V}(X)$ y se evalúa la expresión $\delta = C(Y) - C(X)$. Se realiza una transición de estado si y solo si $\delta < 0$ o usando una probabilidad de aceptación definida como $exp^{-\delta/T}$, donde T es la temperatura, la cual sufre un decremento durante la ejecución del algoritmo. Un algoritmo de recocido simulado es mostrado en la figura 1.

En la figura 1, X es la configuración inicial y solución local durante la ejecución, la asignación $X \leftarrow Y$ representa la aceptación de una nueva configuración Y. En las primeras etapas del algoritmo, si la temperatura es lo suficientemente grande, la expresión $exp^{-\delta/T}$ tiende a 1 haciendo que siempre se acepten nuevas configuraciones por la condicional $u < exp^{-\delta/T}$ cuando $C(Y) > C(X)$. A medida que se disminuye la temperatura, la probabilidad de que se acepten nuevas configuraciones tiende a 0, haciendo que la configuración X tienda a un mínimo.

```
Algoritmo 1.Recocido Simulado
Sea X∈R la configuración inicial
Sea T>0 la temperatura inicial
Sea N un número máximo de iteraciones.
Repetir
  n ← 0
  Repetir
    Generar un vecino Y ∈ V(X)
    δ ← C(Y) − C(X)
    X ← Y ↔ δ < 0    u < exp^(−δ/T); u ∈ (0,1) aleatorio
    n ← n + 1
  Mientras n<=N
Disminuir T
Mientras la temperatura T>0
```

Fig. 1. Algoritmo de recocido simulado.

En la figura 1, X es la configuración inicial y solución local durante la ejecución, la asignación $X \leftarrow Y$ representa la aceptación de una nueva configuración Y. En las primeras etapas del algoritmo, si la temperatura es lo suficientemente grande, la expresión $exp^{-\delta/T}$ tiende a 1 haciendo que siempre se acepten nuevas configuraciones por la condicional $u < exp^{-\delta/T}$ cuando $C(Y) > C(X)$.

A medida que se disminuye la temperatura, la probabilidad de que se acepten nuevas configuraciones tiende a 0, haciendo que la configuración X tienda a un mínimo.

4 Colonia de hormigas

Los algoritmos de colonia de hormigas son metaheurísticas propuestas por Dorigo et al, para la resolución de problemas de optimización combinatoria tipo NP Completo [4]. ACO tiene su fuente inspiradora en el comportamiento de las hormigas en el rastro de feromona, mecanismo que utilizan para la comunicación entre ellas.

ACO basa su funcionamiento en una colonia de agentes simples, hormigas artificiales, que similar al proceso biológico, se comunicarán indirectamente mediante el depósito, evaporación y seguimiento de feromona artificial.

Los rastros de feromona en ACO sirven de información distribuida entre las hormigas para construir, de forma probabilística, soluciones al problema que se va a resolver [5]. Dorigo propuso el esquema básico para la metaheurística de ACO [3]:

```
Algoritmo 2.Optimización por Colonia de Hormigas (ACO)
Paso1: Inicialización de parámetros y rastros de feromona
Mientras las condiciones de paro no se cumplan hacer
  Paso 2: Las Hormigas construyen soluciones.
  Paso 3: Aplicar búsqueda local (opcional).
  Paso 4: Actualizar feromonas.
Fin Mientras
```

Fig. 2. Algoritmo de Optimización por Colonia de Hormigas.

Para poder aplicar el algoritmo ACO, el problema de optimización debe ser traducido a un grafo $G = (V, L)$. V y L representan los vértices y las aristas del grafo respectivamente, y son definidos con las siguientes características y notaciones:

- Un conjunto finito de vértices del problema $V = \{V_1, V_2, \dots, V_n\}$.
- Un conjunto finito E de conexiones entre un subconjunto de elementos de *V*, tal que $|E| \leq N^2$. (*N* representa el número de vértices, $N = |V|$).
- Para cada $e_i \in E$ existe un costo de conexión $d_{i,j}$ que representa el costo de transitar del vértice *i* al vértice *j*, $\eta_{i,j} = 1/d_{i,j}$ es llamada la distancia heurística.
- Un número finito de restricciones Ω definidos sobre los elementos *V* y *E*.
- Dado un conjunto *S* de todas las posibles secuencias $< V_i, V_j, \dots, V_k, \dots >$ sobre los elementos de *V*, se requiere un subconjunto $\hat{S} \subseteq S$, que represente las secuencias factibles con respecto a Ω.
- Una solución Ψ es subconjunto de $\hat{S}$, $\Psi \subseteq \hat{S}$.
- Una función de costo $\Phi_\Psi(E, t)$ que represente el costo total de la solución, donde *t* representa el tiempo.

El algoritmo comienza por asignar un valor real positivo a cada $e_i \in E$ que va a representar el rastro de feromona y se evaporará cada cierto tiempo. Posteriormente, se crearán h hormigas artificiales junto con una lista asociada a ellas denotada como $Tabu_h$. Iniciarán desde una posición inicial arbitraria y la posición inicial será añadida a su lista *Tabu*.

En el paso 2 de la figura 2, las hormigas empiezan a construir soluciones visitando cada uno de los vértices V_i del grafo solo una vez, respetando las restricciones Ω del problema. Para toda hormiga h, se define la probabilidad de transición como la probabilidad de que la hormiga transite de un vértice V_i a un $V_j$en un tiempo *t* mostrada en:

$$P_{ij}^h = \begin{cases} \dfrac{[\tau_{i,j}(t)]^\alpha [\eta_{i,j}]^\beta}{\sum_{x \in Permitidos_h} [\tau_{i,x}(t)]^\alpha [\eta_{i,x}]^\beta} & si\ (i,j) \epsilon Tabu_k \\ 0 & en\ cualquier\ otro\ caso \end{cases} \quad (1)$$

donde $\tau_{i,j}(t)$ es el rastro de feromona en la arista del grafo (i, j) en el instante de tiempo *t*. $Permitidos_h$ es la lista de aristas por las que la hormiga h puede transitar manteniendo la

factibilidad con respecto de Ω; α y β son parámetros de control que determinan la importancia de rastro con respecto a la distancia heurística respectivamente.

Una vez que todas las hormigas hayan completado una secuencia Ψ, en el paso 3 de la figura 2 se actualizará el rastro de feromona en el grafo usando la información recolectada en la lista *Tabu* de las hormigas usando la fórmula:

$$\tau_{i,j}(t+n) = (1-\rho)\cdot\tau_{i,j} + \Delta\tau_{i,j} \qquad (2)$$

donde $\rho \in (0,1)$ representa un coeficiente real y $1-\rho$ representa el coeficiente de evaporación de la feromona en la arista (i,j) y $\Delta\tau_{i,j}$ representa la cantidad total de feromona depositada por las *h* hormigas y se calcula con la fórmula:

$$\Delta\tau_{i,j} = \sum_{k=1}^{h} \Delta\tau_{i,j}^{k}. \qquad (3)$$

la cantidad de feromona depositada en la arista (i,j) por una hormiga *h* es calculada por la fórmula:

$$\Delta\tau_{i,j}^{h} = \begin{cases} \frac{Q}{L_h} & si\ la\ h-ésima\ hormiga\ pasa\ por\ la\ arista\ (i,j) \\ 0 & en\ cualquier\ otro\ caso \end{cases} \qquad (4)$$

donde *Q* es un valor real positivo y $L_h = \Phi_\Psi$ el costo de la solución Ψ o longitud de la ruta encontrada por la hormiga *h*. Posteriormente se vacían las listas *Tabu* y el algoritmo comienza de nuevo, hasta completar el número máximo de iteraciones.

5 Modelo híbrido

El objetivo de la solución propuesta es resolver el problema de *job shop scheduling*, para ello se debe encontrar una secuencia factible de operaciones con el menor tiempo total de proceso posible, conocido como *makespan*. Para el problema del *job shop scheduling* se tienen las siguientes restricciones:

- Cada trabajo se procesa por una máquina solamente una vez.
- Cada operación de los trabajos deben ser procesados en el orden en que son dados, propiedad conocida como restricciones tecnológicas.
- Cada máquina solo puede procesar un trabajo a la vez.
- Cada operación debe ser procesada sin interrupción.
- Las operaciones de un mismo trabajo no pueden ser procesadas de manera concurrente.

Un problema de *job shop scheduling* se denota formalmente de la forma $n/m/G/C_{max}$ donde *n* es el número de trabajos, *m* el número de máquinas, *G* las restricciones tecnológicas del problema y C_{max} la función objetivo que debe minimizarse, en este caso el *makespan*.

Las restricciones tecnológicas *G* son representadas por una matriz. Por ejemplo para un problema de 2 trabajos y 3 máquinas, se tendría la matriz:

$$G = \begin{pmatrix} O_{1,3} & O_{1,2} & O_{1,1} \\ O_{2,2} & O_{2,1} & O_{2,3} \end{pmatrix} \quad (5)$$

para la matriz *G*, la *i*-ésima fila representa el *i*-ésimo trabajo y los elementos en la fila representan las operaciones. Por ejemplo la fila 1 corresponde al primer trabajo que se procesa de acuerdo con la siguiente secuencia: primero la máquina 3 después la máquina 2 y finalmente la máquina 1. Similarmente la secuencia del trabajo 2 establecido por la fila 2, se interpreta como máquina 2, máquina 1 y máquina 3. Además, se tiene una matriz *P* mostrada en (6) que representa los tiempos de procesado de las operaciones de los trabajos.

$$P = \begin{pmatrix} t_{1,1} & t_{1,2} & t_{1,3} \\ t_{2,1} & t_{2,2} & t_{3,3} \end{pmatrix} \quad (6)$$

para cada elemento $t_{i,j} \in P$, *i* corresponde el número de trabajo y *j* al número de máquina.

Para aplicar el algoritmo propuesto primero se requiere una representación adecuada del problema. La figura 3 muestra un ejemplo de la representación del grafo que será explorado por la colonia de hormigas para un problema de 2 trabajos y 3 máquinas.

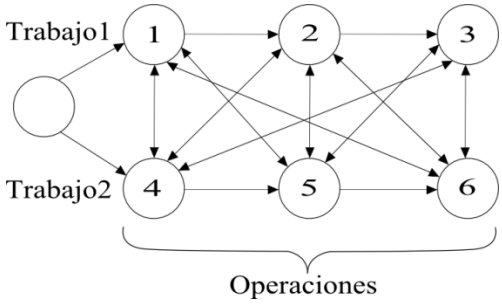

Fig. 3. Representación del problema requerido para aplicar ACO.

Para construir el grafo, cada nodo corresponderá a un elemento en la matriz *G* de la fórmula (5). Para 2 nodos *i*, *j* existirá un arco dirigido (*i*, *j*) en el grafo si y solo si *i* precede a *j* respecto a *G*, o si *i* y *j* no pertenecen al mismo trabajo.

La solución al problema de *job shop scheduling* propuesta está pensada para utilizar la capacidad de ACO de generar buenas soluciones y aprovechar la propiedad de convergencia de recocido simulado [12], para obtener el mejor local del subconjunto del espacio de soluciones (paso 3 de la figura 2) marcado por el rastro de feromona.

La figura 4 muestra la propuesta para el problema de *job shop scheduling*, usando colonia de hormigas asistida con recocido simulado.

En el paso 2 de la figura 4, cada hormiga tiene un nodo inicial asociado a su lista *Tabu*, y en el paso 3 cada hormiga *h* comienza a recorrer el grafo, transitando de un nodo a otro utilizando la probabilidad de transición de la ecuación (1).

```
Algoritmo 3. Modelo híbrido ACO-SA
Paso 1: Inicializar la matriz de feromonas, representación del grafo,
representación de las hormigas e inicialización de la posición inicial
de estas.
Para e ← 1 hasta NIteraciones hacer
    Paso 2: Agregar el nodo inicial a la lista de cada hormiga.
    Paso 3: Cada hormiga construe una solución factible.
    Paso 4: Calcular el mejor makespan encontrado por las hormigas,
    junto con su secuencia.
    Paso 5: Actualizar el makespan global.
    Paso 6: Determinar qué secuencia se usará como configuración inicial
    por recocido simulado.
    Paso 7: Aplicar recocido simulado a la configuración inicial elegida.
    Paso 8: Actualizar el makespan global por segunda vez.
    Paso 9: Actualizar la matriz de feromonas y vaciar las listas de
    cada hormiga.
Fin_para
```

Fig. 4. Modelo híbrido propuesto ACO-SA.

Para la representación de las soluciones y la configuración inicial de recocido simulado, se utiliza una secuencia factible de nodos recorridos por las hormigas. La figura 5 muestra dos ejemplos de secuencias sobre el grafo de la figura 3. La figura 5a muestra el caso cuando una secuencia de nodos es factible, y la figura 5b muestra el caso cuando una secuencia no es factible, es decir, no se respeta el conjunto de restricciones Ω del problema de *job shop scheduling*.

Para la figura 5b, la secuencia es correcta porque sí existe una arista del nodo 2 al nodo 5 pero no es factible ya que el nodo 5 solo puede ser visitado una vez que se visita el nodo 4. Esta situación es de importancia mientras las hormigas construyen las soluciones (paso 3 de la figura 4).

En el paso 6 de la figura 4, el criterio para elegir la configuración inicial para recocido simulado es el siguiente: Si las hormigas encontraron una nueva mejor solución, entonces esa secuencia es utilizada como configuración inicial, en caso contrario se elige al azar uno de los siguientes criterios:

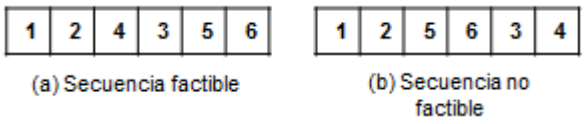

Fig. 5. Ejemplos de secuencias sobre el grafo.

- La mejor secuencia global con probabilidad menor a 0.15.
- La mejor secuencia encontrada por las hormigas en el paso 3 con probabilidad entre 0.15 y 0.50.
- Cualquier otra secuencia encontrada por las hormigas en el paso 3 con probabilidad mayor a 0.50.

El algoritmo de recocido simulado aplicado en el paso 7 de la figura 4, es el mismo descrito en la figura 1. En la solución propuesta, para una secuencia de nodos *X*, se obtiene un vecino $Y = \hat{V}(X)$ eligiendo una posición en la secuencia e intercambiándola con la siguiente siempre y cuando se mantenga la factibilidad de la solución. El método utilizado para disminuir la temperatura es multiplicando la temperatura *T* por un valor $\propto_r \in (0,1)$.

Después de aplicar recocido simulado en el paso 6 de la figura 4, se vuelve a actualizar en el paso 7, la mejor solución encontrada respecto a la solución encontrada por recocido simulado y finalmente en el paso 8, se actualiza la información de la matriz de feromonas igual al paso 4 de la figura 2.

6 Experimentos y resultados

El algoritmo propuesto se validó utilizando problemas prueba reportados en el repositorio de la librería Operation Research [2], en una máquina cuyas características son: procesador Intel Core i3 2.27Ghz, 3GB de memoria RAM, lenguaje de programación OCTAVE [8].

Para cada problema prueba de *job shop scheduling* se procesa la información de la matriz publicada, ya que ésta contiene secuenciación y tiempo de forma alternada.

Los parámetros utilizados en el algoritmo híbrido se calibraron para el *job shop scheduling*. Por ejemplo, para el problema la20 con 50 hormigas y 5 iteraciones, se obtuvo en 10 ejecuciones un promedio de 968 unidades de tiempo, para el mismo problema con 25 iteraciones se llegó a 942.

Se realizaron 10 ejecuciones del algoritmo para cada problema prueba, con los siguientes parámetros calibrados: cantidad de iteraciones 15, cantidad de hormigas 50, feromona inicial 20, $\rho = 0.30$, $\alpha = 1$, $\beta = 10$, Q = 100. En el caso de la técnica de recocido simulado los valores de los parámetros calibrados son: T = 50, N = 70, $\propto_r = 0.20$.

Los resultados obtenidos se comparan con los resultados reportados hasta ahora como los mejores para cada problema, aunque se hayan obtenido con diferentes técnicas [1, 16, 17].

En la tabla 1 se presenta los resultados de las pruebas, donde la primera columna tiene el nombre del problema, en la segunda columna está el tamaño del problema, cantidad de trabajos por la cantidad de máquinas, en la columna tres el mejor valor óptimo reportado, las columnas de la 4 a la 7 son del modelo híbrido propuesto, la primera de ellas es el mejor valor obtenido, en la siguiente es el peor valor obtenido y la última el promedio de correr 10 veces el problema.

Tabla 1. Resultados de las pruebas, donde la primera columna tiene el nombre del problema, en la segunda está el tamaño del problema, cantidad de trabajos por la cantidad de máquinas, en la columna tres el mejor valor óptimo reportado, las columnas de la 4 a la 7 son del modelo híbrido propuesto, la primera de ellas es el mejor valor obtenido, en la siguiente es el peor valor obtenido y la última el promedio de correr 10 veces el problema.

Problema	Tamaño	Mejor reportado	ACO-SA		
			Mejor	Peor	Promedio
La01	10x5	666	**666**	**666**	**666**
La02	10x5	655	**655**	663	656.6
La03	10x5	597	603	626	614.6
La04	10x5	590	**590**	600	596.2
La05	10x5	593	**593**	**593**	**593**
La06	15x5	926	**926**	**926**	**926**
La07	15x5	890	**890**	900	892
La08	15x5	863	**863**	**863**	**863**
La09	15x5	951	**951**	**951**	**951**
La10	15x5	958	**958**	**958**	**958**
La11	20x5	1222	**1222**	**1222**	**1222**
La12	20x5	1039	**1039**	**1039**	**1039**
La13	20x5	1150	**1150**	**1150**	**1150**
La14	20x5	1292	**1292**	**1292**	**1292**
La15	20x5	1207	**1207**	1212	1208
La16	10x10	945	978	988	984.2
La17	10x10	784	983	1016	1002
La18	10x10	848	897	935	921.8
La19	10x10	842	876	907	888.4
La20	10x10	902	914	961	934.6
La30	20x10	1355	1469	1540	1504
La40	15x15	1222	1407	1479	1444.2

Se puede observar en la tabla 1 que en 14 (resaltados en negritas) de 22 problemas prueba, es decir, en el 63.63% se llega a encontrar la mejor solución reportada hasta el

momento y que para los problemas en los cuales no se logra llegar al mejor reportado, se tiene una aproximación promedio del 92%.

7 Conclusiones

Con los resultados obtenidos se concluye que el algoritmo híbrido propuesto, ACO con recocido simulado como búsqueda local, converge en quince iteraciones, es decir, no se requiere de un número grande de iteraciones para encontrar una solución óptima.

En el 63.63% de los problemas prueba, se llega a la mejor solución reportada. En el resto de los problemas se tiene una aproximación promedio del 92%.

Como trabajo futuro se probará en recocido simulado, otras calendarizaciones del enfriamiento así como la elección de los vecinos. En el algoritmo de colonia de hormigas se propondrán formas alternativas para la actualización y evaporación de la feromona.

Referencias

1. Banharnsakun, A., Sirinaovakul, B., Achalakul, T.: Job Shop Scheduling with the Best-so-far ABC. Engineering Applications of Artificial Intelligence, Volume 25 Issue 3, 583-593 (2011)
2. Beasley, J.: OR-Library: Distributing test problems by electronic mail. The Journal of the Operational Research Society, 41(11), 1069-1072 (1990)
3. Dorigo, M., Birattari, M., Stutzle, T.: Ant Colony Optimization: Artificial Ants as a Computational Intelligence Technique. Technical Report, Iridia - Technical Report Series No.23 (2006)
4. Dorigo, M., Maniezzo, V., Colorni, A.: Ant System: Optimization by a colony of cooperating agents. In: IEEE Transactions on Systems, Man and Cybernetics, Part B, pp. 29-41 (1996)
5. Dorigo, M., Stutzle, T.: The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances. In: International Series in Operations Research & Management Science. Vol.57, pp. 250-285 (2003)
6. Duan, L., Havens, W. S.: Appling Systematic Local Search to Job Shop Scheduling Problems: Basic Concepts and Methods. VDM Verlag (2008)
7. Garey, M. R., Johnson, D. S., Sethi, R.: The Complexity of Flowshop and Jobshop Scheduling. Mathematics of operations research, 117-129 (1976)
8. GNU Octave, `http://www.gnu.org/software/octave/`
9. Huang, K. L., Liao, C. J.: Ant colony optimization combined with taboo search for the job shop scheduling problem. Computers and Operations Research, Vol. 35, No. 4, pp. 1030-1046 (2008)
10. Kirkpatrick Jr., S., Gelatt,C. D., Vecchi, M.: Optimization by simulated annealing. Science, Vol.220, No.4598, pp. 671–680 (1983)
11. Meeran, S., Morshed, M. S.: A hybrid genetic tabu search algorithm for solving job shop scheduling problems: a case study. J. of Intelligent Manufacturing, Vol. 23, No.4, pp. 1063-1078 (2012)
12. Mitra, D., Romeo, F., and Vincentelli, A.S.: Convergence and Finite-Time Behavior of Simulated Annealing. Advances in Applied Probability Vol. 18, No. 3, 747-771 (1986)

13. Muñoz Pérez, J.: Inteligencia Computacional Inspirada en la Vida. Servicio de Publicaciones de la Universidad de Málaga, (2010)
14. Prawda, J.: Métodos y Modelos de Investigación de Operaciones, Vol. 2 Modelos estocásticos. Limusa, México (2000)
15. Rojas Santiago, M., Damodaran, P., Muthuswamy, S.: Makespan minimization in a job shop with a BPM using simulated annealing. International Journal of Advanced Manufacturing, Vol. 68, No.9, pp. 2283-2391 (2013)
16. Sureshkumar, S., Saravanan, G., Thiruvenkadam S.: Optimizing Makespan In JSSP Using Unordered Subsequence Exchange Crossover In GA. IOSR Journal of Computer Engineering, Volume 8 Issue 5, 41-46 (2013)
17. Xueni Q., Henry, Y. K. L.: An AIS-based hybrid algorithm for static job shop scheduling problem. J. Intelligent Manufacturing, Vol. 25, No.3, pp. 489-503 (2014)