ISSN 0717-9103
ISSN Online 0718-8307

ESTRATEGIAS MMAS PARA MINIMIZACIÓN DEL MAKESPAN EN LA PROGRAMACIÓN DE UNA MÁQUINA CON SETUP

MMAS STRATEGIES FOR THE SINGLE MACHINE WITH SETUP TIMES AND MAKESPAN MINIMIZATION SCHEDULING PROBLEM

Eduardo Salazar Hornig[1], Oscar Sánchez Pinto[1]
[1]Departamento de Ingeniería Industrial, Universidad de Concepción, Concepción. Chile

RESUMEN

En este trabajo se aplica un algoritmo de optimización de colonia de hormigas con límites de feromona superior e inferior denominado Max–Min Ant System, para resolver problemas de programación de una máquina con tiempos de preparación dependientes de la secuencia y minimización de makespan. El estudio presenta y compara tres variantes del Max–Min Ant System, en base a la forma de actualización de feromona aplicadas a un conjunto de problemas de prueba de 100 trabajos. Las variantes de los algoritmos se comparan para determinar la mejor de ellas y, también, con una heurística constructiva greedy del Mejor Vecino. Se concluye que adecuadas modificaciones a la estrategia de actualización de feromona mejoran significativamente la calidad de las soluciones, obteniéndose soluciones cercanas al óptimo, dado que las diferencias promedio con respecto de una cota inferior del makespan son menores al 1%.

Palabras Claves: Colonia de hormigas, MMAS, una máquina, makespan, setups, mejor vecino.

ABSTRACT

This paper studied an ant colony optimization algorithm with upper and lower pheromone limits called the Max–Min Ant System is applied to solve the single machine scheduling problem with sequence dependent setup times and makespan minimization. The study presents and compares three variants of the algorithm based on the pheromone update applied to a set of problems of size 100 jobs. The algorithms variants are compared between them and with a constructive greedy heuristic of the best neighbor. We conclude that adequate modifications to the pheromone actualization strategy improve significantly the solution quality, obtaining solution near the optimum due to the average differences less than 1% with respect to a lower bound of the makespan.

Keywords: Ant colony optimization, MMAS, single machine, makespan, setups, best neighbor.

Autor para correspondencia: esalazar@udec.cl
Recibido: 09.01.2011 Aceptado: 05.03.2012

INTRODUCCION

Los sistemas de producción por lotes se caracterizan por elaborar múltiples productos en lotes de producción, utilizando la misma instalación, clasificándose estos sistemas según la variedad y homogeneidad de los productos a fabricar (Heizer & Render, 2007), como también por el número de máquinas del sistema Pinedo (2008). Uno de éstos es el sistema de producción de una máquina. El problema de programar *n* trabajos en un *taller de una máquina* consiste en determinar una secuencia de los *n* trabajos a procesar en la única máquina, donde eventualmente se tienen tiempos de preparación dependientes de la secuencia, problema denotado por $1|s_{ij}|C_{max}$ según la notación propuesta por Graham *et al.* (1979).

Diferentes problemas de programación de una máquina con *setup* han sido resueltos con una amplia variedad de heurísticas y algoritmos. Bianco *et al*., 1993 resuelven el problema mediante programación dinámica en base a un algoritmo *branch and bound*, mientras que Laguna *et al.* (1991) utilizan algoritmos *tabu search* y Feo *et al.* (1995) utilizan GRASP. Ambos algoritmos se comparan en problemas de tamaño mediano. Miller *et al*. (1999) utilizan un algoritmo genético híbrido para minimizar costos de *setup* más costos de inventario. Kolahan & Liang (2000) resuelven un problema $1|s_{ij}|C_{max}$ real para la programación de una máquina perforadora de moldes, mediante un algoritmo *tabu search* en combinación con algoritmos genéticos. Lee & Asllani (2004) comparan programación matemática entera con algoritmos genéticos para minimizar el número de trabajos atrasados y el *makespan*. Eren & Guner (2006) proponen una heurística similar a la heurística NEH que entrega una solución inicial para la posterior aplicación de un algoritmo de *tabu search,* el cual obtiene buen rendimiento para problemas con más de 100 trabajos. Se han utilizado algoritmos GRASP y de búsqueda en vecindad para minimizar la tardanza (Gupta & Smith, 2006), como también se han aplicado algoritmos ACO para resolver el mismo problema comparándolo con otros algoritmos (Liao & Juan, 2007).

También se hace referencia a estudios aplicados a problemas del vendedor viajero (TSP o ATSP) por su similitud con el problema $1|s_{ij}|C_{max}$. El TSP consiste en recorrer n ciudades en un circuito cerrado a un costo mínimo propuesto por Dantzig en 1954. La similitud del problema $1|s_{ij}|C_{max}$ con el problema del vendedor viajero asimétrico (ATSP) asocia las distancias asimétricas entre las ciudades ($d_{ij} \neq d_{ji}$) con los *setup* dependientes de la secuencia ($s_{ij} \neq s_{ji}$) característica que hace del ATSP la representación más cercana para resolver el problema $1|s_{ij}|C_{max}$ (Pinedo, 2008).

Para la resolución del ATSP se utilizan algoritmos genéticos en combinación con heurísticas de búsqueda local (Freisleben & Merz, 1996). Choi *et al*. (2003) proponen modificaciones a algoritmos genéticos expandiendo el espacio de búsqueda a regiones de soluciones no factibles, y que una vez reparadas puedan mejorar la solución actual. Métodos exactos, que consideran un subconjunto de arcos específicos para la obtención de soluciones cercanas al óptimo en menor tiempo computacional, son propuestos por Kwon *et al.* (2005).

DEFINICIÓN DEL PROBLEMA

En este trabajo se trata el problema de programar *n* trabajos en un *taller de una máquina con tiempos de preparación dependientes de la secuencia*, que consiste en secuenciar los *n* trabajos de manera de minimizar el *makespan* (C_{max}), es decir, minimizar el intervalo de tiempo entre el inicio del procesamiento del primer trabajo (tiempo de referencia 0) y el tiempo de finalización del procesamiento del último. El tiempo de proceso de cada trabajo está fijo y existen tiempos de preparación de máquina que dependen del orden en el que se procesan los trabajos en la máquina. Se consideran los siguientes supuestos:

- El tiempo de proceso del trabajo *i* está dado por p_i (i = 1,..., *n*).
- Los tiempos de preparación (*setup*) para procesar el trabajo *j* después de procesar el trabajo *i* está dado por s_{ij} (*i* = 0, 1,...,*n*; *j* = 1,…,*n, i* ≠ *j*), donde s_{0i} representa la preparación inicial cuando el trabajo *i* es el primer trabajo procesado en la máquina.
- El proceso de un trabajo en la máquina no se puede interrumpir (*non preemption*).
- Todos los trabajos son independientes entre sí y se encuentran disponibles en el instante inicial.
- La máquina opera sin fallas en el horizonte de programación.
- El objetivo es minimizar C_{max}.

Bajo la notación introducida por Graham *et al*. (1979), el problema de una máquina caracterizado por los supuestos mencionados anteriormente se denota por $1|s_{ij}|C_{max}$, y es un conocido problema NP-Hard (Blazewicz *et al*., 1996, Pinedo, 2008), lo que hace impracticable la obtención de la solución óptima para problemas de mediano a gran tamaño (este problema tiene similar estructura a una variante del problema del vendedor viajero asimétrico ATSP, en el que el vendedor no retorna a la ciudad de origen; los trabajos se asocian a las ciudades y los tiempos de *setup* s_{ij} se asocian a las distancias d_{ij}; así, minimizar la función objetivo C_{max} es equivalente a minimizar la suma de setups). En este trabajo se resuelve el problema $1|s_{ij}|C_{max}$ mediante variantes de un algoritmo MMAS, comparándolo con una heurística *greedy* del mejor vecino (MV). La relevancia de los problemas de programación con tiempos y/o costos de preparación dependientes de la secuencia queda de manifiesto en una amplia gama de configuraciones productivas (Allahverdi *et al*., 2008).

El *makespan* se obtiene como la suma de los *setups* $s_{[i-1][i]}$ que se producen entre el (i – 1) – ésimo e i – ésimo trabajos de la secuencia representados por [i – 1] y [i] respectivamente, más la suma de los tiempos de proceso de los *n* trabajos:

$$C_{max} = \sum_{i=1}^{n} s_{[i-1][i]} + \sum_{i=1}^{n} p_i$$

El tiempo de *setup* $s_{[0][1]}$ representa el *setup* inicial (antes de procesar el primer trabajo de la secuencia). Una heurística simple para resolver la programación para este problema es una heurística *greedy* del *Mejor Vecino (MV)* (Figura 1).

```
procedure Mejor Vecino
   Definir ListaTrabajos ordenada de 1 a n.
   while (ListaTrabajos no vacía) do
      Asignar primer trabajo de la lista como primer trabajo de la secuencia.
      while (Secuencia no completa) do
         Agregar a secuencia trabajo no asignado que origina menor setup.
      endwhile
       Calcular Cmax de secuencia.
      Eliminar primer trabajo de ListaTrabajos.
   endwhile
   Solución ← Secuencia de menor Cmax.
endprocedure
```

Figura 1: Pseudocódigo de heurística del Mejor Vecino (MV)

Optimización Basada en Colonia de Hormigas (ACO)

La optimización basada en colonia de hormigas (Dorigo *et al.,* 1991, 2006; Dorigo & Gambardella, 1997) resuelve problemas combinatorios como el del vendedor viajero, ruteo de vehículos, ruteo en redes de comunicaciones y otros, el cual basa su funcionamiento en el comportamiento de las hormigas en la colonia, específicamente en la capacidad que tienen para encontrar los caminos más cortos desde la colonia a la fuente de comida.

Por ejemplo, las primeras hormigas de una colonia toman diferentes rutas debido a las decisiones iniciales de carácter aleatorio. Conforme avanzan por el camino, van depositando una sustancia denominada feromona, que se evapora progresivamente en el tiempo. Al final, los caminos más cortos tendrán mayor probabilidad de ser recorridos por las siguientes hormigas, debido a que el rastro de feromona que encuentran será mayor. Con el tiempo, la feromona irá incrementándose en los más cortos con mayor rapidez, porque mayor cantidad de hormigas transitarán por ellos terminando por converger en el camino más corto.

El algoritmo ACO dota a las hormigas artificiales características de las hormigas reales reforzadas con otras capacidades que las hacen aún más efectivas y eficientes para resolver problemas de optimización. Por ejemplo, las dota de una memoria interna que contiene los estados visitados, de la capacidad de depositar cantidades de feromona en función a la calidad de la solución encontrada, y de la capacidad de depositar feromona después de haber construido un camino (Dorigo & Di Caro, 1999).

Algoritmos ACO como Ant System (AS), Ant Colony System (ACS) y el Max–Min Ant Sytem (MMAS) han sido comparados con otras heurísticas en problemas ATSP. Por ejemplo, variantes de AS han sido comparadas con un *algoritmo genético*, *simulated annealing* y *tabu search* (Dorigo *et al.,* 1996) obteniendo buenos resultados para AS, ACS se comparó con *branch and bound* combinado con heurísticas de búsqueda local y MMAS con otros algoritmos ACO, estableciendo al ACS y MMAS como las variantes más competitivas para problemas ATSP (Stützle & Hoos, 1997, Sun *et al.,* 2004, Li *et al*., 2009).

Algoritmo Max – Min Ant System para el Problema $1|s_{ij}|C_{max}$

La aplicación de un algoritmo ACO al problema $1|s_{ij}|C_{max}$ se realiza con dos modificaciones importantes a un algoritmo ACO aplicado al ATSP, la solución construida en el problema $1|s_{ij}|C_{max}$ no representa un tour (ciclo cerrado de ciudades a visitar por el vendedor viajero), y la definición de la regla heurística que se considera como el inverso de los *setup* dependientes de la secuencia.

Las metaheurísticas ACO están compuestas fundamentalmente de tres elementos: *inicialización de feromona, regla de transición de estado* y *actualización de feromona*, más *una acción daemon* (mejora local de la solución) de carácter opcional. En este estudio se utiliza un algoritmo ACO denominado MMAS (*Max–Min Ant System*) para resolver el problema $1|s_{ij}|C_{max}$, sin considerar acciones *daemon*.

Regla de transición

Cada hormiga construye una solución (secuencia de *n* trabajos), moviéndose probabilísticamente a través de una secuencia de arcos (i,j), los movimientos son seleccionados aplicando una regla de transición de estado en función del nivel de feromonas (τ_{ij}) y la información heurística $\eta_{ij} = 1/s_{ij}$ según la ecuación 2 propuesta por Dorigo *et al.* (1996).

ISSN 0717-9103
ISSN Online 0718-8307

$$p_{ij} = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{k \in S} [\tau_{ik}]^{\alpha} \cdot [\eta_{ik}]^{\beta}} & \text{si} \quad j \in S \\ 0 \quad \text{en otro caso} \end{cases} \qquad (1)$$

donde τ_{ij} es el nivel de feromona depositada en el arco (i,j), la que se actualiza de acuerdo a la estrategia de actualización del algoritmo. La información heurística $\eta_{ij} = 1/s_{ij}$ se interpreta como la *visibilidad* del arco (i,j), en este caso mientras menor sea el *setup* s_{ij} entre los trabajos i y j (arco (i,j)) más alto es el valor de η_{ij}. Los parámetros α y β determinan la influencia del rastro de feromona y la información heurística en la toma de decisiones. Según la ecuación, la hormiga prefiere moverse probabilísticamente a trabajos con bajo valor de *setup* y que estén conectadas por un arco con alto valor de feromona. En cada decisión *S* corresponde al conjunto de trabajos no secuenciados.

Cada hormiga repite los pasos hasta que finaliza su secuencia, y luego calcula el *makespan* de la solución generada gracias a la memoria interna representada por *S* que guarda la secuencia de trabajos.

Actualización de Feromona

Los arcos (i,j) que pertenecen a una solución (trabajo i precede inmediatamente a trabajo j en la secuencia) son actualizados en base a la ecuación 2 propuesta por Stützle & Hoos (1997).

$$\tau_{ij} = (1 - \rho)\, \tau_{ij} + \Delta\tau_{ij} \qquad (2)$$

En la ecuación 2, τ_{ij} es la feromona depositada actualmente en el arco (i,j), ρ es la fracción de evaporación de feromona y $\Delta\tau_{ij}$ es la cantidad de feromona depositada sobre el arco (i,j) como actualización del nivel de feromona del arco. $\Delta\tau_{ij} = 1/L_b$ si la hormiga transita por el arco (i,j) y $\Delta\tau_{ij} = 0$ si la hormiga no transita por el arco (i,j) con L_b igual al valor del *makespan* de la solución definida por la estrategia de actualización de feromona utilizada, esto es, la mejor solución global o la mejor solución en cada ciclo. Esto significa que sólo los arcos que pertenecen a la solución definida aumentarán su feromona, mientras que los arcos que no pertenecen a esta solución disminuirán su feromona.

La acción de evaporar y actualizar el rastro de feromona no es suficiente para incentivar la exploración de nuevas soluciones. Por eso MMAS define limites de feromona máximo, τ_{max}, y mínimo, τ_{min}, para todos los arcos del grafo. Luego que una hormiga construye la secuencia solución, se actualiza su feromona de acuerdo a la estrategia utilizada entre los límites de feromona mínimo y máximo.

Inicialización de Feromona

Al inicio todos los arcos se ajustan al valor τ_{max}. Entonces, los arcos que no reciben feromona adicional durante la ejecución del algoritmo reducen paulatinamente su nivel de feromona,

mientras que los arcos que reciben feromona mantienen un alto nivel. El valor τ_{max} se calcula según la ecuación 3 (Stützle & Hoos, 2000):

$$\tau_{max} = \frac{1}{\rho} \cdot \frac{1}{L^*} \tag{3}$$

donde L^* representa el *makespan* del secuenciamiento óptimo, sin embargo, inicialmente este se desconoce y es precisamente el objetivo de la aplicación del algoritmo, por lo que es estimado por una solución heurística simple. En este trabajo se obtiene como la mejor solución de entre 10 soluciones generadas en forma aleatoria, y ρ es la fracción de evaporación de feromona. La ecuación 3 resulta de suponer que luego de muchos ciclos la feromona acumulada en un arco (i,j) perteneciente a la solución óptima converge al valor τ_{max}, esto es:

$$\lim_{t \to \infty} \tau_{ij}(t) = \tau_{max}$$

De la ecuación 2, la feromona acumulada en uno de estos arcos luego de t ciclos es:

$$\tau_{ij}^{max}(t) = \sum_{k=1}^{t} (1-\rho)^{t-k} \cdot \frac{1}{L^*} + (1-\rho)^t \cdot \tau_{ij}(0) .$$

La suma parcial $\sum_{i=1}^{t} (1-\rho)^{t-i}$ converge a $\frac{1}{\rho}$ cuando t → ∞ (serie geométrica de razón (1–ρ)), y $(1-\rho)^t \cdot \tau_{ij}(0)$ converge a 0 cuando t → ∞ debido al valor de ρ < 1, dando lugar a la ecuación 3. El límite inferior de feromona τ_{min} se calcula en función a τ_{max}, mediante la ecuación 4 (Stützle & Hoos, 2000):

$$\tau_{min} = \frac{\tau_{max} \cdot \left(1 - \sqrt[n-1]{p_{best}} \right)}{avg \cdot \sqrt[n-1]{p_{best}}} \tag{4}$$

La ecuación 4 se obtiene a partir del supuesto de convergencia del algoritmo, situación en donde la mejor solución del problema se obtiene cuando los niveles de feromona de sus arcos son igual a τ_{max}, mientras que los demás arcos permanecen en un nivel τ_{min}, donde p_{best} es una estimación de la probabilidad de construir la mejor solución. Si p_{dec} es la probabilidad de seleccionar el arco correcto en cada paso; entonces:

$$p_{best} = p_{dec}^{n-1} \quad \text{de donde} \quad p_{dec} = \sqrt[n-1]{p_{best}}$$

Bajo el supuesto de que la relación entre los niveles de feromona es suficiente para guiar a las hormigas en la construcción de la mejor solución, la regla de transición (ecuación 1) al arco (i,j) es la probabilidad p_{dec} sin dependencia de la información heurística:

$$p_{dec} = \frac{\tau_{max}}{\tau_{max} + avg \cdot \tau_{min}}$$

donde *avg* representa el número medio de arcos posibles de elegir (adicionales al mejor arco) en cada decisión al construir una solución; al despejar τ_{min} se obtiene la ecuación 4. En base a referencias de la literatura (Stützle & Hoos, 1997, 2000) se consideró p_{best} = 0.05, *avg* = n/2.

Notar que τ_{max} y τ_{min} se recalculan en cada ciclo tomando L^* como el valor del *makespan* de la mejor solución conocida.

Estrategias de Actualización de Feromona

En este estudio se aplican tres estrategias de actualización de feromona para MMAS:

$MMAS_1$: Se actualiza la feromona en los arcos de la mejor solución global; es decir, el valor de L_b en la ecuación 2 corresponde al *makespan* de la mejor solución observada hasta el momento (Stützle & Hoos, 1997).

$MMAS_2$: Se actualiza la feromona en los arcos de la mejor solución local; es decir, el valor de L_b en la ecuación 2 corresponde al *makespan* de la mejor solución del ciclo actual (Sun *et al.,* 2004, Li *et al*., 2009, Stützle & Hoos, 2000).

$MMAS_3$: Se actualiza la feromona en los arcos de todas las soluciones construidas en cada ciclo, similar a la estrategia utilizada en el Ant System (Dorigo *et al*., 1996); es decir, el valor de L_b en la ecuación 2 corresponde sucesivamente al *makespan* de la solución construida por cada hormiga.

ESTUDIO EXPERIMENTAL

La evaluación de los algoritmos se realizó utilizando instancias generadas en forma aleatoria, de acuerdo a distribuciones de tiempos de proceso y de *setup* utilizadas en la literatura. Los tiempos de proceso de los trabajos se generaron de la distribución uniforme discreta entre 1 y 100 ($p_i \sim UD[1,100]$). Los tiempos de preparación dependientes de la secuencia de los trabajos se generaron de la distribución uniforme discreta entre 1 y 50 ($s_{ij} \sim UD[1,50]$).

En base a referencias de la literatura (Stützle & Hoos, 1997, 2000, Sun *et al,* 2004), se consideró α=1, y se realizó un análisis experimental sobre 5 problemas de tamaño 100 para ajustar los parámetros relevantes de MMAS β y ρ, utilizando 10 hormigas y 10 réplicas de 500 ciclos. Para el ajuste se consideraron valores de β = 1, 2, 3, 4 y 5, y ρ = 0.02, 0,05 y 0,10. Para el ajuste de parámetros se realizó un análisis de varianza determinando valores de parámetros para cada estrategia base: para $MMAS_1$, β= 5; ρ= 0.02; para $MMAS_2$ β= 4 ó 5; ρ= 0.05; y para $MMAS_3$ β= 5; ρ= 0,1.

Los métodos evaluados fueron las estrategias base con respecto de la heurística constructiva *greedy* del Mejor Vecino (MV). Posteriormente se realizan modificaciones a las estrategias base, con el objetivo de mejorar sus desempeños.

Para la comparación de algoritmos se generaron 30 instancias de problemas de 100, 50 y 30 trabajos, haciendo énfasis en los problemas de gran tamaño (100 trabajos). En el análisis de optimización se utilizó un total de 2500 ciclos. La evaluación de la heurística se realizó por medio de rutinas adaptadas del software SPS_Optimizer (Salazar, 2010), herramienta diseñada para la programación de operaciones.

Para evaluar el desempeño se utilizó el *makespan* (C_{max}) como medida de desempeño, el que se compara con una cota inferior *(CI)* utilizando la medida:

$$\%Dif = \frac{Sol_{Método} - CI}{CI} * 100$$

$Sol_{Método}$ es el valor del *makespan* obtenido con el respectivo método. La cota inferior se obtiene sumando los tiempos de proceso con el mínimo tiempo de *setup* inicial, más la suma por fila de los mínimos tiempos de *setup*, y restando el mayor de los mínimos tiempos de *setup* por fila:

$$CI = \sum_{i=1}^{n} p_i + \min_{i=1,..,n}\{s_{ii}\} + \sum_{i=1}^{n} \min_{j=1,..,n;j\neq i}\{s_{ij}\} - \max_{i=1,..,n}\{\min_{j=1,..,n;j\neq i}\{s_{ij}\}\}$$

Para poder comparar los algoritmos se define la medida *%pDif* que corresponde al promedio de las diferencias porcentuales de la solución de un método respecto de su cota inferior en todas las instancias analizadas (en la tabla 1, 0.82 es el promedio de las diferencias porcentuales observadas al aplicar el algoritmo $MMAS_1$ a las 30 instancias analizadas.

Comparación de Algoritmos MMAS

Los tres algoritmos MMAS se compararon en 30 problemas de tamaño 30, 50 y 100; sin embargo, se presentan aquí los resultados del estudio para los problemas de tamaño 100, dado que los resultados observados en los problemas de tamaño 30 y 50 trabajos son similares llegando a las mismas conclusiones. El valor de la cota inferior y los resultados de las diferencias porcentuales del *makespan* obtenido por cada algoritmo con respecto a la cota inferior en las 30 instancias de problemas de 100 trabajos consideradas se muestra en la tabla A1 del Anexo (columnas Cota, $\%MMAS_i$ y %MV).

En la figura 2 se grafican los resultados comparando los algoritmos $MMAS_1$, $MMAS_2$, $MMAS_3$ y MV, en el eje horizontal se tienen las instancias (para una mejor presentación no se incluye los rótulos para la identificación de las 30 instancias en el eje horizontal) y en el eje vertical se presenta el valor de *%Dif*.

De la tabla A1 y figura 2 se observa que los algoritmos MMAS obtienen mejor solución que la heurística MV en todas las instancias comparadas. La tabla 1 resume el porcentaje de veces que un algoritmo obtiene la mejor solución (%MS) y la diferencia promedio de los algoritmos respecto de la cota (*%pDif*), estableciendo un criterio de evaluación entre estos algoritmos, observando que $MMAS_1$ muestra mejor rendimiento promedio que $MMAS_2$ y $MMAS_3$, lo que se corrobora con el 76,67% de los casos en que obtuvo la mejor solución entre estos algoritmos (en un caso existió empate entre $MMAS_1$ y $MMAS_2$).

Tabla 1. Resumen de resultados de Algoritmos

Algoritmo	$MMAS_1$	$MMAS_2$	$MMAS_3$	MV
%MS	76,67	26,67	0,00	0,00
%pDif	0,82	0,90	1,16	2,14

ISSN 0717-9103
ISSN Online 0718-8307

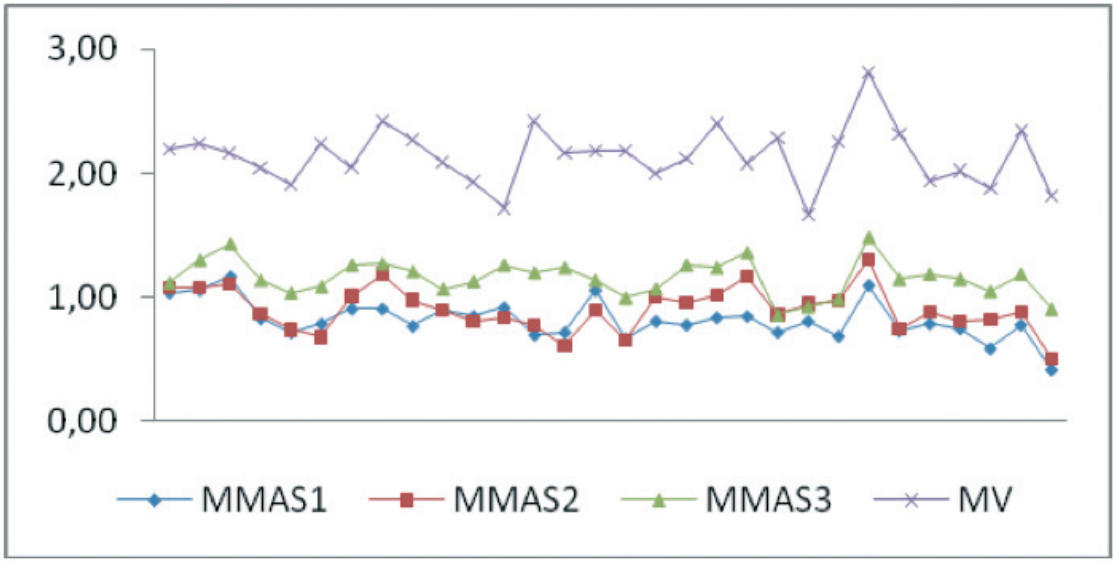

Figura 2. Comparación de Métodos – C_{max}

Comparación de Algoritmos MMAS

Se introducen modificaciones a las estrategias de actualización de feromona mediante la combinación de características entre si, a modo de influenciarlas en su rendimiento:

Modificación a $MMAS_1$: Después de cada ciclo, tanto los arcos de la mejor solución global como los arcos de la mejor solución local son considerados para la actualización de feromona (influencia ejercida por la estrategia 2 aplicada a la estrategia 1) → denominada estrategia $MMAS_4$.

Modificación a $MMAS_2$: Cada cierto número de ciclos se realiza una actualización global (influencia ejercida por la estrategia 1 aplicada a la estrategia 2) → denominada estrategia $MMAS_5$.

Modificación a $MMAS_3$: Cada cierto número de ciclos se realiza una actualización sobre los arcos de la mejor solución global (influencia ejercida por la estrategia 1 aplicada a la estrategia 3) → denominada estrategia $MMAS_6$.

La tabla 2 resume como resultado los efectos de las modificaciones realizadas, indicando el número de veces (nMS) y el porcentaje de veces (%MS) en que el respectivo algoritmo obtiene la mejor solución, y el porcentaje promedio de diferencia con respecto de la cota inferior (%pDif). La modificación de $MMAS_1$ dio como resultado la obtención de mejores soluciones en la mitad de los problemas (con 1 empate), esto hace de $MMAS_4$ una buena alternativa para resolver este problema. En cuanto al rendimiento promedio $MMAS_4$ es similar a $MMAS_1$, $MMAS_5$ tiene un rendimiento promedio superior a $MMAS_2$, y $MMAS_6$ también mejora el rendimiento promedio en relación al $MMAS_3$ para problemas de 100 trabajos.

En base a la complementariedad de los resultados obtenidos, especialmente entre el $MMAS_1$ y $MMAS_4$, se consideró el mejor resultado de ambos, así como también el mejor resultado entre $MMAS_2$ y $MMAS_5$ generando los algoritmos denominados $MMAS_{1\text{-}4}$, $MMAS_{2\text{-}5}$ y $MMAS_{3\text{-}6}$ respectivamente.

Tabla 2. Comparación de estrategias MMAS

Algoritmo	nMS	%MS	%pDif
$MMAS_1$	16	53,33	0,82
$MMAS_4$	15	50,00	0,83
$MMAS_2$	6	20,00	0,90
$MMAS_5$	26	86,67	0,84
$MMAS_3$	12	40,00	1,16
$MMAS_6$	20	66,67	1,11

De la tabla 3. se tiene que el algoritmo $MMAS_{1-4}$, resultó ser el mejor en la comparación efectuada, obteniendo mejores resultados en 23 de los 30 problemas de 100 trabajos seguido de $MMAS_{2-5}$ con 11 (se registraron 5 empates con $MMAS_{1-4}$) y $MMAS_{3-6}$ con 1. La comparación de las diferencias porcentuales respecto de la cota de los algoritmos $MMAS_{1-4}$, $MMAS_{2-5}$ y $MMAS_{3-6}$ se muestra en la figura 3.

Tabla 3. Algoritmos MMAS modificados

Algoritmo	nMS	%MS	%pDif
$MMAS_{1-4}$	23	76,67	0,78
$MMAS_{2-5}$	11	36,67	0,83
$MMAS_{3-6}$	1	3,33	1,08

De la tabla 3 se tienen las diferencias porcentuales promedio de los algoritmos $MMAS_{1-4}$, $MMAS_{2-5}$ y $MMAS_{3-6}$ con respecto a la cota, lo que se interpreta de la siguiente forma: en promedio las soluciones de estos algoritmos difieren (con poca variabilidad) a lo más en un 0,78%, 0,83% y 1,08% de la solución óptima.

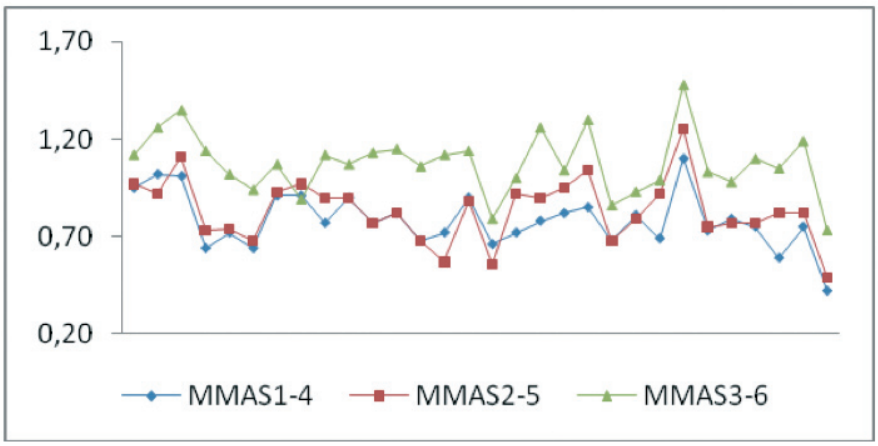

Figura 3. Comparación de Métodos – %Dif en problemas de tamaño 100

Para los problemas analizados, el uso de algoritmos MMAS produce soluciones que en promedio difieren a lo más en un 1% de la solución óptima, a diferencia de la heurística constructiva MV que en promedio está a lo más a 2,14% (Tablas 1, 2 y 3), pero esta última obtiene la solución en un tiempo computacional prácticamente despreciable.

El procesamiento se realizó en un computador Intel Core 2 Duo T7500 de 2.2 GHz de 2 GB de RAM. El orden de magnitud del tiempo CPU por réplica para la ejecución de los algoritmos MMAS fue de 135 segundos por réplica, mientras que para la heurística MV fue de 0,05 segundos.

CONCLUSIONES

Se utilizó el algoritmo MMAS (Max–Min Ant System) para resolver el problema de programación de una máquina con tiempos de preparación dependientes de la secuencia y minimización de *makespan*, evaluando tres estrategias de actualización de feromona.

El rendimiento del algoritmo $MMAS_1$, estrategia que en cada ciclo actualiza la feromona sobre los arcos de la mejor solución global, resultó superior a las otras 2 estrategias ($MMAS_2$ y $MMAS_3$) porque obtuvo los mejores resultados en la mayoría de los problemas y mejor diferencia promedio respecto de la cota inferior.

El buen rendimiento de $MMAS_1$ se logró mejorar actualizando en cada ciclo, tanto los arcos de la mejor solución global como los de la mejor solución del ciclo. El carácter complementario de $MMAS_1$ y su modificación ($MMAS_4$) hace del uso combinado de ambos ($MMAS_{1\text{-}4}$) una buena alternativa de solución del problema tratado.

Las modificaciones introducidas a los algoritmos $MMAS_2$ y $MMAS_3$ produjeron mejoras, tanto en el número de mejores soluciones encontradas como en la diferencia promedio respecto de la cota inferior; sin embargo, no superaron a la modificación de $MMAS_1$. De igual forma, el uso combinado de estos algoritmos con sus respectivas modificaciones mejora su rendimiento, pero sin superar al uso combinado de $MMAS_1$ con su modificación.

Se concluye, en general, que una modificación a la estrategia de actualización de feromona mejora notablemente la calidad de las soluciones obtenidas por un algoritmo MMAS, lo que constituye una buena alternativa de aplicación de MMAS en otros problemas y en otros algoritmos ACO.

El porcentaje promedio de diferencia con respecto de la cota inferior menor al 1% obtenido por prácticamente todos los algoritmos MMAS, hace plantear la hipótesis de que en los problemas evaluados estos algoritmos generan soluciones muy cercanas al óptimo.

REFERENCIAS

Allahverdi, A., Ng, C.T., Cheng, T.C.E & Kovalyov, M. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187(3), 985- 1032.

Bianco, L., Mingozzi, A. & Ricciardelli, S. (1993). The Traveling Salesman Problem with Cumulative Costs. *Networks*, 23(2), 81- 91.

Blazewicz, J., Ecker, K., Pesch, E., Schmidt, G. & Weglarz, J. (1996). Scheduling Computer and Manufacturing Processes. Springer.

Choi, I., Kim, S. & Kim, H. (2003). A Genetic Algorithm with a mixed region search for the asymmetric traveling salesman problem. *Computers & Operations Research*, 30, 773 – 786.

Dorigo, M., Birattari, M & Stützle, T. (2006). Ant Colony Optimization--Artificial Ants as a Computational Intelligence Technique. *IEEE Computational Intelligence Magazine*, 1, 28 – 39.

Dorigo, M. & Di, Caro G. (1999). The ant colony optimization meta-heuristic – New ideas in optimization. McGraw-Hill.

Dorigo, M. & Gambardella, L.M. (1997). Ant Colony System: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1, 53 – 66.

Dorigo, M., Maniezzo, V. & Colorni, A. (1996). The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems and Cybernetics – Part B*, 26, 29 – 41.

Dorigo, M., Colorni, A. & Maniezzo, V. (1991). Distributed Optimization by Ant Colonies. *Proceedings of the First European Conference on Artificial Life*, Paris, France, 134- 142.

Eren, T. & Guner, E. (2006). A bicriteria scheduling with sequence-dependent setup times. *Applied Mathematics and Computation*, 179, 378 – 385.

Feo, T.A., Sarathys, K. & McGahan, J. (1995). A GRASP for single machine scheduling with sequence dependent setup costs and linear delay penalties. *Computers & Operations Research*, 23, 881 – 895.

Freisleben, B. & Merz, P. (1996). New genetic Local search operators for the Traveling Salesman Problem. *LNCS*, Springer, 1141, pp. 890 – 899.

Graham, R.L., Lawler, E.L., Lenstra, J.K. & Rinnooy, K.A.H.G. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5, 287 – 326.

Gupta, S & Smith, J. (2006). Algorithms for single machine total tardiness scheduling with sequence dependent setups. *European Journal of Operational Research*, 175(2), 722 – 739.

Heizer J. & Render B. (2007). Dirección de la producción y de operaciones – Decisiones tácticas. Pearson/PrenticeHall.

Kolahan F. & Liang M. (2000). Optimization of holemaking operations: a Tabu - search approach. *International Journal of Machine Tools & Manufacture*, 40, 1735 – 1753.

Kwon S., Kim H. & Kang M. (2005). Determination of the candidate arc set for the asymmetric traveling salesman problem. *Computers & Operations Research*, 32, 1045 – 1057.

Laguna M., Barnes W.J. & Glover F.W. (1991). Tabu search methods for a single machine scheduling problem. *Journal of Intelligent Manufacturing*, 2, 63 – 74.

Lee S.M. & Asllani A.A. (2004). Job scheduling with dual criteria and sequence-dependent setups: mathematical versus genetic programming. *Omega*, 32, 145 – 153.

Li T., Chen W., Zheng X. & Zhang Z. (2009). An improvement of the ant colony optimization algorithm for solving Traveling Salesman Problem (TSP). *5th WiCom Conference*, 1 – 3.

Liao C. & Juan H. (2007). An ant colony optimization for single-machine tardiness scheduling with sequence-dependent setups. *Computers & Operations Research*, 34, 1899 – 1909.

Miller D.M., Chen H-C., Matson J. & Liu Q. (1999). A Hybrid Genetic Algorithm for the single machine scheduling problem. *Journal of Heuristics*, 5(4), 437 – 454.

Pinedo, M. (2008). Scheduling – Theory, Algorithms and Systems. Springer. 3th Edition.

Salazar, E. (2010). Programación de Sistemas de Producción con SPS_Optimizer. *Revista ICHIO*, 1(2), 33 – 46 (Digital).

Sun J., Xiong S.W. & Guo F.M. (2004). A new pheromone updating strategy in ant colony optimization. *Proceedings of the Third International Conference on Machine Learning y Cybernetics*, Shangai, 26 – 29.

Stützle T. & Hoos H. (2000). Max-Min Ant System. *Future Generation Computer Systems*, 16(8), 889 – 914.

Stützle T. & Hoos H. (1997). Improvements on the Ant System: Introducing MAX- MIN ant system. In *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms*. Springer-Verlag, 245 – 249.

Anexo

Tabla A1. Resultados de la Cota Inferior para C_{max} y Algoritmos (%Dif)

Instancia	Cota	%MMAS$_1$	%MMAS$_2$	%MMAS$_3$	%MMAS$_4$	%MMAS$_5$	%MMAS$_6$	%MV
1	5387	1,04	1,08	1,12	0,95	0,97	1,24	2,20
2	4921	1,06	1,08	1,30	1,02	0,92	1,26	2,24
3	5034	1,17	1,11	1,43	1,01	1,11	1,35	2,16
4	4692	0,83	0,87	1,14	0,64	0,73	1,14	2,04
5	5363	0,72	0,74	1,04	0,74	0,81	1,02	1,91
6	5063	0,79	0,68	1,09	0,64	0,83	0,94	2,24
7	5346	0,91	1,01	1,26	0,93	0,93	1,07	2,05
8	5009	0,91	1,19	1,27	1,01	0,97	0,89	2,42
9	4868	0,77	0,98	1,21	0,79	0,90	1,12	2,27
10	5214	0,90	0,90	1,07	1,00	0,90	1,21	2,09
11	5337	0,85	0,81	1,13	0,77	0,77	1,17	1,93
12	5444	0,92	0,84	1,26	0,82	0,82	1,15	1,72
13	5141	0,70	0,78	1,20	0,68	0,68	1,06	2,42
14	5444	0,72	0,61	1,24	0,74	0,57	1,12	2,17
15	4880	1,06	0,90	1,14	0,90	0,88	1,18	2,18
16	4983	0,67	0,66	1,00	0,66	0,56	0,79	2,18
17	4777	0,81	1,00	1,07	0,72	0,92	1,00	2,00
18	5075	0,78	0,96	1,26	1,08	0,90	1,26	2,12
19	4932	0,84	1,02	1,24	0,82	0,95	1,04	2,40
20	4738	0,85	1,17	1,36	0,93	1,04	1,30	2,08
21	4514	0,72	0,86	0,86	0,68	0,68	0,97	2,29
22	5233	0,81	0,95	0,93	0,81	0,79	1,06	1,67
23	4877	0,69	0,97	0,99	0,86	0,92	1,14	2,25
24	4599	1,10	1,31	1,48	1,27	1,25	1,58	2,81
25	5129	0,73	0,75	1,15	0,81	0,79	1,03	2,32
26	5298	0,79	0,88	1,19	0,85	0,77	0,98	1,94
27	5261	0,75	0,81	1,15	0,83	0,77	1,10	2,02
28	5117	0,59	0,82	1,05	0,79	0,89	1,11	1,88
29	5330	0,78	0,88	1,19	0,75	0,82	1,21	2,35
30	4950	0,42	0,51	0,91	0,51	0,49	0,73	1,82
	%pDif	0,82	0,90	1,16	0,83	0,84	1,11	2,14