Una aplicación de Ant Colony Optimization para la resolución de problemas de Flow Shop.*

Joaquín Bautista[1] , Marcela de la Rosa[2], Jordi Pereira[3], Ramón Companys[4]
[1]ETSEIB-DOE-UPC, Diagonal 647, Planta 7, 08028, Barcelona, bautista@oe.upc.es
[2]ETSEIB-DOE-UPC, Diagonal 647, Planta 7, 08028, Barcelona,, delarosa@oe.upc.es
[3]ETSEIB-DOE-UPC, Diagonal 647, Planta 7, 08028, Barcelona,, pereira@oe.upc.es
[4]ETSEIB-DOE-UPC, Diagonal 647, Planta 7, 08028, Barcelona, companys@oe.upc.es

RESUMEN

ACO, acrónimo de "Ant Colony Optimization", es una meta-heurística que puede emplearse en la resolución de problemas de optimización combinatoria; se inspira en el comportamiento de guía observado en las hormigas, y se caracteriza por la utilización de "feed-back" positivo, su estructura paralelizable y el empleo de heurísticas constructivas específicas del problema a resolver. En este trabajo se presenta la aplicación de dicha meta-heurística a la resolución de problemas de Flow Shop y se comparan los resultados obtenidos con los ofrecidos por otros procedimientos heurísticos y exactos.
***Palabras clave**: Flow Shop, Secuenciación, Soft Computing, Meta-Heurística ACO.*

1 Introducción.

El problema de Flow-Shop sin buffers intermedios [1] es un problema de optimización combinatoria que se puede encontrar en diversas industrias: producción de lotes de productos químicos, fabricación en línea de autobastidores, etc. Sin pérdida de generalidad, el problema consiste en secuenciar *n* piezas (órdenes de trabajo) en *m* máquinas (etapas productivas) dispuestas en serie. El flujo del producto a través del sistema es regular, es decir, todas las piezas siguen la misma ruta de máquinas y se conocen los tiempos de proceso (pieza-máquina) y los instantes de disponibilidad de todas las piezas y de todas las máquinas. A lo anterior, debemos añadir las siguientes condiciones: (1) entre todo par de máquinas consecutivas no hay espacio para almacenar piezas, (2) una pieza debe ser tratada por todas las máquinas siguiendo una secuencia de operaciones, denominada ruta, que es idéntica para todas las piezas, (3) una máquina no puede procesar más de una pieza a la vez y (4) una pieza no puede recibir más de una operación al mismo tiempo.

El objetivo perseguido, teniendo en cuenta las condiciones anteriores, es hallar un orden de entrada de piezas al sistema de forma que el tiempo requerido para completar el trabajo en todas las piezas sea mínimo.

El problema puede ser resuelto mediante diversos métodos generales (para problemas de optimización combinatoria), tales como las heurísticas greedy basadas en reglas de prioridad o por algún algoritmo de exploración de entornos.

Algunos algoritmos de exploración de entornos - como el recocido simulado, la búsqueda tabú y los algoritmos genéticos [2]- han mostrado su validez en la resolución de problemas combinatorios obteniendo resultados satisfactorios; aunque recientemente han aparecido otros métodos como los algoritmos meméticos (algoritmos genéticos en que se combinan las ideas evolutivas Darwinistas con las Lamarkianas, y por lo tanto se profundiza en las mejoras locales dentro del proceso genético), los algoritmos de inteligencia de masas "Swarm

* El presente estudio se ha realizado en el marco del proyecto de investigación TAP98-0494 financiado por la CICYT.

Intelligence" [3], basados en los comportamientos de colmena de ciertos insectos, o la evolución diferencial, que al igual que en con los anteriores están obteniendo buenos resultados.

En el presente trabajo se describe la resolución del problema de Flow-Shop sin buffers intermedios mediante un algoritmo de optimización por colonias de hormigas (ACO *Ant Colony Optimization*). Para ello, se presenta primero el problema de Flow-Shop sin buffers. A continuación, se describe la meta heurística utilizada y su implementación. Finalmente, se concluye comparando los resultados obtenidos por evolución diferencial con otros procedimientos.

2 El problema de Flow-Shop sin buffers intermedios.

2.1 Introducción.

El problema de flow-shop consiste en la secuenciación de un número de órdenes que deben pasar por un número de procesos con un orden fijado para todas ellas. El caso particular sin buffers intermedios, sin stocks intermedios, proviene de la suposición implícita de existencia de colas frente a las máquinas en la formulación tradicional, caso que no se cumple en ciertas circunstancias, como en la producción química por lotes, lo que lleva a la resolución de un problema distinto.

Normalmente, la resolución del problema mediante métodos heurísiticos consiste en generar permutaciones que representan la secuencia a resolver, pudiendo aplicar a posteriori o durante el proceso de generación las heurísticas de intercambio aplicables en estos problemas. El problema a su vez puede ser resulto de forma exacta mediante diversos algoritmos como el Lomnicki y el Lomnicki Pendular (LOMPEN), aprovechando la inversibilidad del problema, pero con peores resultados, en general, que en el caso de Flow-Shop debido a que las cotas utilizadas son las adaptadas al problema general de Flow-Shop, y por tanto no explotan todo el conocimiento del problema.

2.2 Nomenclatura.

El problema de flow-shop sin buffers intermedios consiste en secuenciar n lotes o piezas, índice i, que deben de ser procesadas por m máquinas o procesos, con un índice de etapa j, con un tiempo de proceso p_{ij} dependiente de cada pieza y cada máquina, con un orden fijado para todas ellas (hasta aquí es la definición de un problema de Flow-Shop convencional) y que entre máquina y máquina no existen colas donde los lotes o piezas puedan permanecer antes de ser procesados. Esta hipótesis conlleva implícitamente el bloqueo de las máquinas, ya que si una pieza termina su operación en una máquina, bloqueará la máquina hasta que la máquina siguiente esté libre.

Al igual que en los problemas clásicos de flow-shop el índice de eficiencia de la secuencia puede ser cambiante. El índice más habitual, y el utilizado en este caso es el de minimización de Fmax, tiempo de finalización del tratamiento de la última pieza secuenciada y siguiendo la notación utilizado para los problemas de flow-shop, el problema se expresaría como n/m/Fzb/Fmax , donde el subíndice zb la inexistencia de colas (zero buffer).

Las restricciones del problema pueden formularse de la siguiente forma:

$$e_{j,k} + p_{j,[k]} \leq f_{j,k} \quad (1)$$
$$e_{j,k} \geq f_{j,k-1} \quad (2)$$
$$e_{j,k} \geq f_{j-1,k} \quad (3)$$
$$f_{j,k} \geq f_{j+1,k-1} \quad (4)$$

con la siguiente nomenclatura adicional:

[k] : la pieza situada en la posición k-ésima de la permutación (k=1,2,…,n).
$e_{j,k}$ instante en que la pieza [k] entra en la etapa j desde la etapa j-1
$f_{j,k}$: instante en que sale la pieza [k]hacia la etapa j+1

Las tres primeras restricciones (1),(2),(3) son las típicas de la estructura del problema de flow-shop y corresponden a (1) tiempo de proceso en la máquina de cada pieza, (2) la máquina sólo puede procesar una pieza a la vez y (3) la pieza no puede ser procesada en dos máquinas a la vez. La nueva restricción (4) obliga a que el tiempo de salida de una pieza en la máquina que está en proceso deba de ser como mínimo el tiempo de salida de la pieza anterior en la secuencia de la máquina siguiente, lo que conlleva implícitamente el bloqueo de las máquinas.

Para calcular el tiempo de inicio de una operación en una máquina sólo es necesario resolver la siguiente operación:

$$e_{j,k} = \max(e_{j,k-1} + p_{j,k}; e_{j-1,k}) \quad (5)$$

2.3 Un ejemplo.

Para ilustrar el problema y el método de cálculo de la función objetivo, se muestra un pequeño problema a continuación.

Una empresa del sector químico debe elaborar tres lotes de producto a través de tres procesos de forma ordenada (cocción, fermentado y deshidratado). Debido a las características de los mercados a los que se destinará cada producto, el tiempo de proceso en las máquinas responsables de realizar la tarea tienen duraciones diferentes. Además, debido a las condiciones del sistema productivo, los lotes de producción no pueden almacenarse entre proceso y proceso, ocupando la máquina anterior en caso de no encontrar libre la máquina siguiente para ser procesado. El interés de la empresa es finalizar estos lotes de exportación en el mínimo tiempo para poder continuar utilizando sus máquinas para la elaboración de otros lotes de producción.

El tiempo de proceso para cada lote y cada máquina en la siguiente tabla:

	Cocción	Fermentado	Deshidratado
Lote 1	3	7	6
Lote 2	5	2	4
Lote 3	1	5	5

Existen n! secuencias posibles, Una posible secuencia para la realización de la producción sería ejecutar primero el lote 1, a posteriori el lote 3 y finalmente el lote 2. A continuación se

muestra el tiempo de entrada y finalización de proceso para cada proceso y lote de dicha secuencia.

	Cocción	Fermentado	Deshidratado
Lote 1	0/3	3/10	10/16
Lote 3	3/8	10/12	16/20
Lote 2	10/15	15/20	20/25

Como puede verse, la falta de almacenes o colas entre procesos provoca tiempos de muertos en las máquinas debido a la ocupación de la máquina por un lote al no poder ser procesado en la máquina siguiente. El lote 3 termina su proceso de cocción en el instante 8, pero no libera ese recurso hasta el instante 10, cuando el proceso de fermentado queda libre del lote 1, y por lo tanto el lote 3 puede empezar a ser fermentado, dejando el horno de cocción disponible para el lote 2.

3 La meta heurística ACO.

La meta-heurística ACO, Ant Colony Optimization, es un procedimiento heurístico evolutivo utilizado en la resolución de problemas de optimización discreta inspirado en el comportamiento de las hormigas. Concretamente, utilizan una analogía de su sistema de comunicación indirecto, que les permite resolver problemas mediante la colaboración multi-agente. Sus principales características son: (1) la utilización de "feed-back" positivo (acelera el descubrimiento de soluciones de gran calidad), (2) computación distribuida (la estructura de estos algoritmos permite su paralelización de forma muy simple y natural), y (3) el uso de heurísticas Greedy constructivas.

En la meta-heurística ACO [4], una colonia de "hormigas artificiales" busca, de forma colectiva, soluciones de calidad para el problema de optimización considerado. Cada hormiga construye una solución mediante sucesivas decisiones. En cada punto de decisión, se dispone de una lista de posibles candidatos cuya elección se realiza mediante una regla de transición probabilística, basada en información heurística sobre el problema e histórica de las soluciones construidas hasta el momento. Cada hormiga construye una solución con información propia (heurística) e información histórica (rastros de "feromona") de las soluciones construidas hasta el momento.

La información histórica que las hormigas utilizan durante la construcción de soluciones consiste en un vector de índices de decisión para decidir su política de exploración. Los valores de este vector son función de los rastros de feromona disponibles en un instante determinado y de los valores heurísticos.

Una vez una hormiga ha construido una solución, deposita (en alguno de los atributos de su solución) una cantidad de feromona proporcional a la calidad de la solución que la hormiga ha construido.

Desde el momento en que se propuso esta meta-heurística, han existido diversas implementaciones y variaciones de ella, destacando MINMAX-ACO, [5] y ACS [6]. La versión propuesta en este trabajo se basa el ACS (Ant Colony System) para resolver el problema del viajante de comercio, cuya aportación principal sobre la metaheurística ACO es la introducción de hormigas elitistas, lo que indica que no todas las hormigas depositan rastro de feromonas, sino que sólo aquellas que presentan mejores soluciones lo hacen.

El tratamiento de las soluciones, dadas las características del problema, puede presentarse de dos formas, establecer una única secuencia de n piezas que tuviera en cuenta los aspectos de las m máquinas únicamente durante la evaluación, o establecer una secuencia de n·m elementos, donde se tuviera en cuenta el mantenimiento del orden según reglas de precedencia. La opción adoptada en esta implementación es la segunda al ser más general y poder resolver otros problemas más generales.

En la formulación propuesta [7], las diferentes operaciones de una pieza se presentan como operaciones que presentan lazos de dependencia, sin tener en cuenta la máquina en la que deben ser procesadas. Esta asignación a una máquina en concreto se formaliza cuando se realiza la evaluación de la solución. Así si se disponen de 4 piezas con 3 operaciones cada una, se consideran 12 tareas en lugar del orden de 4 piezas. Adicionalmente se debe mantener el orden en que las máquinas procesan las piezas. Para ello se realiza una modificación dinámica de la matriz de precedencias, después de introducir una nueva tarea en la secuencia se debe establecer que las siguientes tareas de todas las candidatas no secuenciadas, deben tener como predecesora la tarea siguiente a la nueva incorporación. A través de este enfoque, el rastro se depositará entre las parejas de tareas que se secuencian correlativamente.

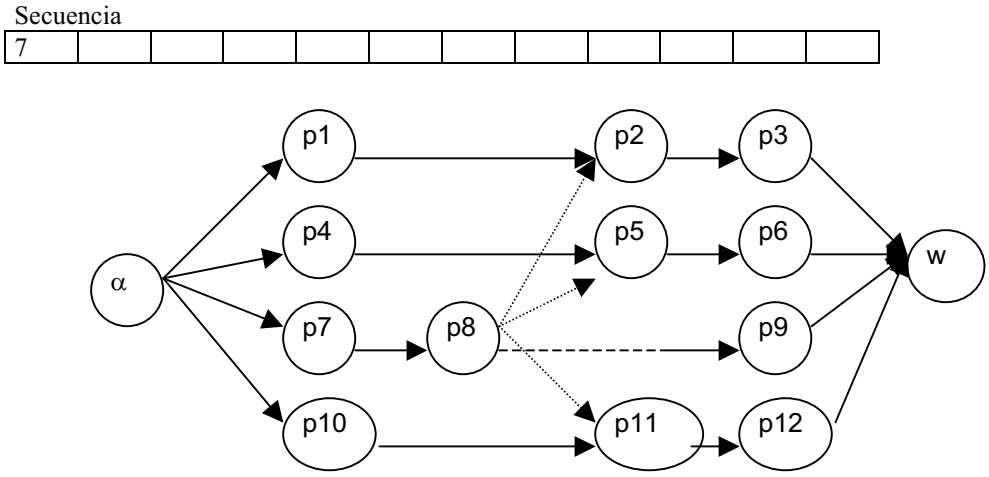

Figura 1: Modificación grafo de precedencias entre actividades para un problema de 4 piezas y 3 máquinas, se amplía para mantener el orden de la secuencia.

Dada la representación de una solución, el rastro entre soluciones y la forma heurística de construcción, podemos plantear el esquema del algoritmo:

1 Inicializar la matriz de rastros.
2 Crear subcolonia de hormigas
3 Las subcolonias crean una solución, utilizando la información del rastro y de la heurística propia de cada hormiga.
4 Actualizar el rastro con las nuevas secuencias de tareas
5 Si se cumple la condición de final, finalizar. Si no, volver al paso 2

Cada subcolonia (punto 2 y 3 del algoritmo anterior) es un conjunto independiente, formado por unidades funcionales denominadas hormigas. Cada hormiga es independiente de las demás y es capaz de crear una secuencia de tareas propia. El funcionamiento de cada hormiga se describe a continuación:

1 Crear una hormiga con un criterio heurístico, en este caso el valor de la cota de Nabeshima.
2 Crear una lista de tareas candidatas para ser añadidas a la secuencia de tareas.
3 Calcular los índices de decisión utilizando el criterio heurístico propio y el rastro entre la última tarea asignada y las candidatas
4 Asignar la tarea en función de los índices
5 Actualizar las tareas candidatas. Ir al paso 2 hasta que ya no queden candidatas.
6 Evaluar la lista de tareas.

Cada vez que una hormiga asigna una nueva tarea a la secuencia en curso, actualiza la lista de candidatas, aquellas tareas que cumplen las restricciones de precedencia, y calcula para cada una de ellas un índice de probabilidad que le servirá para escoger la siguiente tarea. Éste índice combina el conocimiento acumulado en la matriz de rastros y la inteligencia propia de la hormiga debido su criterio heurístico intrínseco. Se calculará un índice para cada tarea j del conjunto de tareas candidatas C, siendo i la última tarea asignada a la secuencia de tareas:

$$u_{ij} = \left[\tau_{ij}\right]^{\alpha} \cdot \left[\eta_{j}\right]^{\beta} \quad \forall j \in C \tag{6}$$

donde τij es el rastro histórico entre la tarea i y la j, ηj corresponde al peso de la tarea j según el criterio heurístico intrínseco de la hormiga y C es el conjunto de candidatas.

Los parámetros α y β controlan la importancia del rastro y la inteligencia heurística. Por lo tanto, la probabilidad de transición es un compromiso entre ambos. En el presente trabajo se han adoptado unos valores de α y β iguales a 0,75 y 0,25 respectivamente.

La probabilidad de escoger cada tarea se obtiene de la ponderación entre todos los índices de las tareas candidatas:

$$p_{ij} = \frac{u_{ij}}{\sum_{k \in J} u_{ik}} \quad \forall j \in C \tag{7}$$

La actualización de la matriz de rastros se realiza después de que cada subcolonia de hormigas haya construido y evaluado todas las secuencias construidas por sus hormigas, utilizándose solamente la secuencia de tareas con mejor función objetivo (OBJmejor) para dejar rastro, basándose en el concepto de hormiga elitista.

La actualización del rastro se realiza incrementando los valores de rastro entre las parejas de vértices adjacentes (τij) siguiendo la siguiente expresión:

$$\tau_{ij} \leftarrow (1-\rho) \cdot \tau_{ij} + \frac{1}{(d(\%)^{mejor})^2} \tag{8}$$

donde d(%) es la discrepancia de la función objetivo respecto a la mejor cota hallada para el problema mediante la cota de nabeshima.

Además de esta actualización cíclica, cuando alguna subcolonia mejora la mejor función objetivo encontrada hasta en momento, se vuelve a actualizar la matriz de rastros, depositando una cantidad extra de feromona, con el objetivo de garantizar que siempre que se encuentre una buena solución el resto de hormigas pueda seguirla:

4 Experiencia Computacional y conclusiones.

La experiencia computacional que se ha llevado a término es la continuación de la aparecida en [1]. Se dispone de 6 colecciones de 1000 problemas cada uno en los que se combinan varios valores de piezas (10, 12 y 15) y de máquinas (3, 4, 5). En todos los problemas se dispone de la solución óptima de manera que es posible establecer la bondad de las soluciones y compararlo con otros métodos anteriores.

Los parámetros utilizados para realizar las comparaciones de los métodos corresponden al número de veces que se obtiene la solución óptima y el error medio para cada una de las colecciones.

Se ha resuelto este problema fijando el número de subcolonias a 60 (840 soluciones) y para cada problema se ha permitido un máximo de 70 mejoras.

Piezas	10		12		15	
Máquinas	3	5	4	5	3	4
Palmer	0	1	0	0	0	0
Teixido	18	7	2	0	1	0
Gupta	14	1	1	0	1	0
Trapecios	8	7	4	1	0	0
Palmer + RAES	180	159	55	56	63	22
Teixido + RAES	235	193	81	82	79	32
Gupta + RAES	232	165	106	76	81	40
Trapecios + RAES	219	187	90	100	76	29
Búsqueda Tabú	6	652	398	415	174	142
ACO	368	395	267	304	91	199
Aco + mejora	989	1000	1000	1000	964	993

Tabla 1: Comparación según el número de óptimos

Piezas	10		12		15	
Máquinas	3	5	4	5	3	4
Palmer	20,47%	15,06%	20,23%	17,34%	24,81%	22,86%
Teixido	10,54%	8,75%	11,73%	10,69%	15,81%	14,45%
Gupta	9,13%	11,10%	11,67%	12,64%	14,23%	14,20%
Trapecios	10,61%	7,41%	10,28%	9,02%	15,75%	12,93%
Palmer + RAES	2,57%	2,53%	3,04%	2,99%	2,82%	3,61%
Teixido + RAES	2,01%	2,01%	2,60%	2,54%	2,66%	3,07%
Gupta + RAES	2,08%	2,22%	2,64%	2,67%	2,58%	2,95%
Trapecios + RAES	2,03%	1,92%	2,38%	2,31%	2,50%	2,92%
Búsqueda Tabú	17,29%	0,48%	0,85%	0,82%	1,24%	1,29%
ACO	2,76%	2,51%	2,97%	2,78%	4,30%	3,63%
Aco + mejora	2,70%	0,00%	0,00%	0,00%	4,62%	2,22%

Tabla 2: Comparación según el error medio

De los resultados presentados puede decirse que de los métodos propuestos la Búsqueda Tabú es mejor que el ACO presentado. No obstante, si se observan los resultados del ACO con una mejora local en las mejores soluciones de cada subcolonia, puede apreciarse como ésta

obtiene los mejores resultados. Este método obtiene, en el peor de los casos un 96% de óptimos.

Referencias

[1] Companys, R; Mateo, M; Bautista, J. (1999) "Flow-shop sin pulmones", *Actas III Jornadas de Ingeniería de Organización, Volumen II*, Barcelona 16 y 17 de septiembre de 1999, pp. 515-521. ISBN:84-95355-01-9, Barcelona, septiembre de 1999.

[2] Díaz, A.; Glover, F.; Ghaziri, H; González, J.M.; Laguna, M.; Moscato, P.; Tseng, F. (1996). *Optimización heurística y redes neuronales*. Paraninfo.

[3] Bonabeau E., Dorigo M., Theraulaz G., (1999) *Swarm Intelligence From Natural to Artifical Systems*. Oxford University Press.

[4] Colorni A., Dorigo M. , Maniezzo V. (1992) "Distributed Optimization by Ant Colonies", *Proceedings of the First European Conference on Artificial L ife*, Paris, France, F.Varela and P.Bourgine (Eds.), Elsevier Publishing, pp. 134-142.

[5] Stützle T. , Hoos H. (1997). "Improvements on the ant system: Introducing MAX-MIN ant system", *In Proceding of the International Conference on Artificial Neural Netwo rks and Genetic Algorithms*, pp. 245-249. Springer Verlag, Wien.

[6] Dorigo M. , Gambardella L.M. (1997) "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem", *IEEE Transactions on Evolutionary Computation*, 1(1):53-66.

[7] Bautista J., Bretón, J. Fernández J. De la Rosa M. (2001) "Meta-heurística ACO (Ant Colony Optimization) para la resolución de problemas en líneas de producción", *Actas Congreso sobre Técnicas de Ayuda a la Decisión en la Defensa*, Madrid 12 al 15 diciembre 2000, pp. 267-287. Edita Secretaría General Técnica Ministerio de Defensa, ISBN:84-7823-831-X, Madrid, abril 2001.