Meta-heurística ACO (Ant Colony Optimization) para la resolución de problemas en líneas de producción[1]

Joaquín Bautista, Javier Bretón, José A. Fernández, Marcela de la Rosa

Laboratorio de Organización Industrial
Departamento de Organització de Empresas
Universidad Politécnica de Cataluña
e-mail:{bautista, delarosa}@oe.upc.es

Resumen

La meta-heurística ACO (Ant Colony Optimization) es un procedimiento heurístico para la resolución de problemas de optimización discreta basado en el comportamiento de las hormigas. Sus principales características son: (1) la utilización de "feed-back" positivo, (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 (ayuda a encontrar soluciones aceptables en las primeras etapas del proceso de exploración).
En este artículo se presenta la aplicación de dichas meta-heurísticas a la resolución de problemas de producción como equilibrado de líneas de montaje o la secuenciación de unidades en un sistema que no permite esperas.

Palabras clave: Heurísticas, ACO

1 Introducción

Unas de las mayores fuentes de inspiración para la elaboración de heurísticas en la investigación operativa han sido los fenómenos naturales. En ellos se intenta reproducir situaciones o comportamientos de la vida real, aplicándolos a los problemas de optimización combinatoria.

Entre los fenómenos que se han utilizado, se pueden encontrar leyes genéticas que dan lugar a los algoritmos con el mismo nombre, leyes físicas, resumidos en algoritmos como el recocido simulado, o incluso comportamientos sociales de algunos animales, como es el caso de los algoritmos de hormigas.

Todos los métodos basados en fenómenos naturales, realizan búsquedas heurísticas de las soluciones del problema; esto es, dado el espacio de soluciones exploran parte de él en busca de soluciones. Por esta razón, en algunos casos, las soluciones que se obtienen no corresponde con valores óptimos. Por ejemplo, los algoritmos genéticos parten de una familia de soluciones y realizan cruces y/o mutaciones, dando lugar a nuevas soluciones que no tienen porque ser mejores que las soluciones iniciales. Otro caso parecido es el del Recocido Simulado, en que a partir de una solución, se va mejorando hasta llegar a un óptimo local, donde, según cierta probabilidad, puede tender a empeorar, intencionadamente, la soluciones para intentar acceder a un nuevo espacio de secuencias posibles.

[1] Esta investigación ha sido parcialmente subvencionada por el proyecto TAP-0494

En general, en los algoritmos heurísticos, la búsqueda de la solución depende de varios parámetros o estrategias para la obtención de soluciones de calidad. La primera de las opciones es decidir si el algoritmo "desarrollará" la solución completa-algoritmos constructivos- o si intentará mejorar una solución inicial –algoritmos de mejora-. Otra de las opciones posibles, es el tamaño de los elementos de estudio para cada iteración; es decir, existen algoritmos que trabajan con una única soluciones cada vez, mientras que otros trabajan con familias de soluciones. De igual manera, también es posible que el algoritmos utilice la "memoria histórica" de la resolución o que inicien la búsqueda cada vez de cero. En el presente trabajo se presenta un algoritmo basado en el comportamiento de las hormigas que se caracteriza por ser constructivo, utilizar una subcolonia de soluciones (más de una solución) y recordar las decisiones tomadas, entre algunas de sus características más relevantes.

Dada la componente heurística de los algoritmos, es preciso la definición de unos parámetros de control que establezcan tanto la dirección que la búsqueda debe tomar como el criterio de fi. Dichos parámetros dependen del algoritmo y son los decisivos en la bondad de las soluciones por lo que es importante la definición correcta y precisa. Por ejemplo, en los algoritmos de Recocido simulado son los parámetros de temperatura y la componente de probabilidad la que establecen si el algoritmo debe empeorar, orientando así la búsqueda hacia otros sectores no explorados, o mejorar, profundizando en el vecindario escogido.

Los resultados obtenidos con los métodos heurísticos son bastante buenos aunque es importante tener en cuenta las siguientes limitaciones:

- En ningún caso estos métodos pueden garantizar la obtención de soluciones óptimas
- Existe una fuerte dependencia de la bondad de las soluciones con los parámetros de control.

En este trabajo se presenta un algoritmo para la resolución de problemas de secuenciación basado en el comportamiento de las hormigas. En la sección 2, se definen los algoritmo ACO y los parámetros de control más importantes. En la sección 3 se presenta el esquema básico del algoritmo utilizado. Dicho esquema ha sido utilizado para la resolución de un problema de equilibrado de líneas (SALPB-E) y para un problema de secuenciación de las tareas de un taller teniendo el cuenta que el flujo dentro de él es regular (caso flujo regular o *flow-shop*). Las particularidades de estos dos casos, así como los resultados obtenidos se presentan en los apartados 4 y 5, respectivamente. En las sección 6 se exponen las conclusiones más relevantes del estudio y las futuras líneas de trabajo.

El objetivo que se perseguía en este trabajo era la implementación de un algoritmo ACO de gran flexibilidad que permitiera resolver diferentes tipos de problemas realizando pocos ajustes. En este sentido, se puede decir que el algoritmo propuesto permite la resolución de diferentes tipos de problemas realizando únicamente las modificaciones referentes a la entrada de los datos, el cálculo de las cotas del problema y las funciones de evaluación.

2 Los algoritmos de hormigas

2.1 Presentación de los algoritmos de hormigas

Las hormigas reales son capaces de encontrar el camino más corto desde su nido hasta la fuente de alimentación sin disponer el sentido de la vista. Además presentan la capacidad de adaptarse al entorno cuando se producen cambios en él, volviendo a encontrar el mejor camino. Esta características se debe a una substancia denominada feromona.

Cuando la hormiga se desplaza deja un rastro que marca el camino que ha seguido de manera, así las siguientes hormigas lo detectaran junto con el del resto de predecesoras. Las hormigas que deban realizar el camino hasta el alimento, escogerán uno de los caminos según una función de probabilidad que dependerá de la intensidad del rastro. En el caso de los caminos utilizados por un gran número de hormigas, el rastro de feromona será más intenso.

Los algoritmos basados en el comportamiento de las colonias de hormigas, se han utilizado en los últimos años en disciplinas como la investigación operativa, las redes de comunicación, la robótica o la optimización combinatoria. En *Monmarché et al.(2000)* aparece una revisión muy completa de esta variedad en las aplicaciones.

Algunas de las implementaciones más importantes inspiradas en este fenómeno, y en particular las utilizadas en las aplicaciones para rutas y la optimización combinatoria, siguen una estructura similar que ha desembocado en la consideración de este planteamiento como una nueva meta-heurística, denominado ACO (*Ant Colony Optimizaton)*. Estos algoritmos, se caracterizan por utilizar las hormigas “virtuales” como los agentes que se desplazan por el grafo asociado al problema.

Para evitar una convergencia demasiado rápida hacia soluciones no tan deseables, también se mantiene en la versión digital la propiedad de evaporación de la feromona. En el caso real, al tratarse de una substancia orgánica, se produce una evaporación en función del tiempo. Para el caso de los algoritmos de hormigas, se ha establecido que ha medida que se realiza una iteración, el rastro disminuye para evitar que las soluciones iniciales, a pesar de poder ser peores, dispongan de un rastro proporcional a la calidad.

En estas condiciones, se han desarrollado desde el año 1991 algunos métodos ACO para casos de optimización combinatoria como el problema del viajante de comercio (TSP), la asignación cuadrática o la secuenciación de las operaciones de un taller según un flujo general.

Una de las propiedades más importantes de los algoritmos ACO se encuentra en que una colonia finita de hormigas busca, de forma colectiva, soluciones de gran calidad. Cada individuo, cuando construye una solución, realiza una aportación a la comunidad con el único objetivo de colaborar a que otras hormigas obtengan mejores soluciones a través de la memoria histórica almacenada en forma de rastro.

En el estado permanente, el camino que escoge la hormiga para avanzar depende, según una variable probabilística, del criterio asignado en su creación (regla heurística asociada a cada hormiga) y del rastro de feromona existente desde la posición actual hasta las tareas candidatas. Esta probabilidad es propia de la programación de la meta-heurística, así como también lo es la decisión sobre la

cantidad y el momento para dejar el rastro de feromona. Generalmente, la cantidad de substancia depositada es función de la calidad de la solución Esto es especialmente importante ya que cuanto mayor sea el rastro más atrayente será para las siguientes hormigas.

2.2 Propiedades

Parámetros de control

Como se ha explicado anteriormente, el camino que escoge la hormiga depende de las propiedades del algoritmo programado. Esta dependencia se muestra a través de los parámetros α y β que son los que controlan el peso de la información histórica y de la regla heurísticas, respectivamente. Es mediante éstos con los que se define si se desea proporcionar mayor importancia al instinto de la hormiga (heurística asociada) o si se prefiere priorizar el rastro dejado por la colonia (información histórica disponible por la cooperación del resto de hormigas).

Para evitar la convergencia rápida hacia soluciones no tan deseables se define el parámetro ρ como el porcentaje de evaporación en cada iteración del rastro. Esta evaporación tiene como objetivo "hacer olvidar" parte de la memoria histórica para no provocar una dependencia demasiado prematura de la resolución hacia la información del rastro.

Otros parámetros de control interesantes son los referentes a los rastros en "condiciones especiales" para las situaciones que precisan de un tratamiento diferente. Cuando se desea que el rastro quede amplificado, como por ejemplo para hacer más interesante el camino a la mejor solución conseguida, se debe establecer cuál es la cantidad extra de feromona. Así mismo, para la situación inicial del problema es preciso establecer el valor del rastro a priori.

Por último, no se pueden olvidar los parámetros referentes a las limitaciones del tamaño de la exploración. Como se apuntó en la introducción, las meta-heurísticas se caracterizan por obtener buenas soluciones, que no tienen que ser óptimas, con un coste temporal sensiblemente menor al de los algoritmos exactos. Esta duración se debe establecerse ya sea estableciendo directamente el tiempo de ejecución o limitando el número de soluciones; en este caso, hormigas.

Importancia del criterio heurístico asociado a la hormiga

Los algoritmos Greedy son aquellos que crean soluciones utilizando un único criterio de exploración (tiempo de proceso menor o mayor, número sucesoras,...) constante durante toda la búsqueda de solución. En estos casos, las reglas funcionan bien durante las primeras etapas pero empeora cuando llegan a las etapas finales al estar obligados a aplicar la misma regla escogida en el inicio.

Es por ese motivo que en los algoritmos ACO se utilizan, durante los instantes iniciales, en mayor medida las reglas heurísticas y a medida que hay más experiencia, éstas perderán peso frente a la memoria del rastro. No obstante, hay que recordar que depende de una variable de probabilidad.

3 Resolución de problemas de secuencias mediante ACO

3.1 Consideraciones iniciales

La meta-heurística propuesta se basa en gran medida en el sistema ACS (*Ant Colony System)* propuesto por *Dorigo et al.* (1997) para el problema del viajante de comercio (TSP). En este trabajo, se resuelven dos problemas de secuencias presentando las siguientes analogías para poder aplicar el algoritmo descrito por Dorigo:

- Cada tarea de la secuencia equivale a una ciudad del TSP.
- Las restricciones de precedencia entre tareas se representan por medio de los arcos del grafo teniendo en cuenta los sentidos. Como existen estas restricciones el número de candidatas por etapa no es k! (siendo k las tareas no secuenciadas) sino que es un número menor.
- Una trayecto del TSP (camino cerrado que recorre todas las ciudades sin repetir ninguna) tiene su analogía en la secuencia completa de tareas donde no existen repeticiones de operaciones y donde se respetan las restricciones de precedencia.
- La construcción de la solución se realiza paso a paso. Cuando una hormiga l se encuentra en un nodo i debe escoger el siguiente nodo a que desplazarse. La lista de candidatas depende del nodo i y de los vértices secuenciados para garantizar la factibilidad de la solución final. Como en el problema TSP la elección se realiza mediante una regla probabilística de transición.
- Cuando se actualiza el rastro del TSP, se deposita un rastro de feromona en los arcos del grafo según el orden de la solución. En los problemas de secuenciación el rastro se depositará entre parejas de tareas según la solución encontrada.
- La actualización del rastro se realiza para incrementar los valores de la memoria de los componentes de soluciones de mejor calidad; por lo tanto, la cantidad de rastro depositada dependerá del inverso de la calidad de la función objetivo alcanzada en la solución.
- Mientras que en el problema de TSP se utiliza una única regla heurística, en la resolución presentada se utilizan 14 criterios diferentes.

3.2 Esquema de funcionamiento

Es necesario establecer cual será el criterio de finalización para el momento en que se considere que no es posible mejorar la solución obtenida hasta el momento. Los criterios más conocidos y utilizados hacen referencia a la desviación de la solución aunque otros más simples como el tiempo de ejecución o el número de subcolonias también son utilizados. Para la resolución de los problemas de este trabajo se ha optado por este último criterio.

El esquema de funcionamiento general de la heurística se puede resumir según el diagrama.

Paso 1: Inicialización
- Paso 1.0: Entrada de datos
- Paso 1.1: Cálculo de cotas del problema
- Paso 1.2: Cálculo de la matriz de pesos
- Paso 1.3: Inicialización del rastro

Paso 2: **mientras** no se alcance la condición de fin
- Paso 2.1: Creación de la subcolonia
- Paso 2.2: Actualización de los rastros
- Paso 2.3: **si se alcanza mejor solución**
  - Paso 2.3.1: Añadir rastro extra
  - Paso 2.3.2: Consolida la matriz de rastros. **Volver a paso 2**
- Paso 2.4: **si el rastro es muy débil**
  - Paso 2.4.1: Recuperar última matriz consolidada. **Volver a Paso 2**

Esquema de funcionamiento

3.3 Descripción de rastros

Para poder establecer el grafo único de un problema cuando existen varios productos con operaciones independientes, es preciso definir dos tareas ficticias que marquen el inicio y final de las operaciones del problema completo. Los arcos desde estos nuevos vértices hasta las primeras tareas "reales" también deben contener la información referente a la elección de estas operaciones. Así los rastro deberán almacenarse en una matriz $(n+2)\cdot(n+2)$ donde cada valor τ_{ij} representa la cantidad de feromona si después de la tarea i se secuencia la tarea j.

La inicialización de esta matriz se puede realizar siguiendo diferentes estrategias. En el caso de este estudio, se ha decidido fijar el rastro inicial de feromona en valores elevados ya que se ha comprobado que esta decisión utiliza en menor medida las reglas heurísticas cuando se encuentran soluciones de calidad.

Por otro lado, la actualización de la matriz de rastros se realiza después de que cada subcolonia haya finalizado la construcción y evaluación de las secuencias utilizando únicamente aquellas soluciones con mejor función objetivo. La cantidad de rastro depositado (ec. 1) depende del rastro anterior y de la discrepancia de la función objetivo de la secuencia con relación a la mejor cota encontrada (d(%)).

En caso que alguna solución obtenga un resultado de más calidad que la mejor solución lograda hasta el momento, se vuelve a actualizar la matriz depositando una cantidad extra de feromona en la nueva secuencia encontrada. La cantidad extra dependerá de cuánto mejor es la nueva mejor solución (ec. 2). La idea es que se almacene con mayor intensidad aquellos caminos que mejores soluciones proporcionan para que las hormigas sucesoras exploren ese mismo vecindario en busca de óptimos.

El rastro depositado queda definido por:

$$\tau_{ij} = (1-\rho)?\tau_{ij} + \frac{1}{(d(\%)^{mejor})^2} \qquad (1)$$

$$\tau_{ij} = Q\max_{\forall k \neq j}\{\tau_{ik}\} \quad \forall i,j \in \text{mejor solución} \qquad (2)$$

Después de un cierto número de etapas sin mejora, para evitar que por efecto de la evaporación se pierda la información almacenada en la matriz de rastros, cada vez que se obtiene una solución de mejor calidad se realiza una consolidación de la matriz. Así, si durante el proceso se cumple que el rastro es menos intenso que un determinado valor, la matriz se reemplaza por los últimos valores almacenados en la consolidación. En el caso que se presenta, se ha considerado como criterio para reemplazar la matriz de rastro que no se produzca mejora en 35 ocasiones de 100 subcolonias.

3.4 Funcionamiento de la subcolonia

Las subcolonias están formadas por tantas hormigas como criterios heurísticos se desee. Así, la unidad funcional de la subcolonia es la hormiga. Para la búsqueda, cada hormiga es independiente del resto y es capaz de crear una secuencia de tareas propia. Para ello utiliza tanto la información histórica recogida por sus compañeras, como el criterio heurístico intrínseco en ella (ec.4), defiendo un índice (ec. 3). Éste será el que se utilizará para el cálculo de la probabilidad de escoger la tarea (ec. 5).

El índice de decisión u_{ij} se define como

$$u_{ij} = \left[\tau_{ij}\right]^{\alpha}\left[\eta_j\right]^{\beta} \qquad (3)$$

$$\eta_j = \frac{v(j)}{\sum_{k=1}^{nhormigas} v(k)} \qquad (4)$$

donde τ_{ij} es el rastro, η_{ij} es el peso de la decisión de tomar la tarea j según el criterio heurístico y v(j) es el peso del criterio heurístico (ver anexo)

Así la probabilidad se define:

$$p_{ij} = \frac{u_{ij}}{\sum_{k \in candidatas} u_{ik}} \qquad (5)$$

El funcionamiento de cada hormiga se puede formalizar:

mientras no sea la última hormiga de la subcolonia
Paso 1: Creación de las hormigas según el criterio heurístico
Paso 2: Definición de las tareas candidatas
Criterio: Número de precedentes asignadas igual al número total de precedentes.
Paso 3: Cálculo de los índices de decisión de la lista de candidatas
Paso 4: Elección de la siguiente tarea asignada
Paso 5: Adición de la nueva tarea a la ristra o secuencia en curso
Paso 6: **si** la tarea asignada no está en la última posición posible, **volver Paso 2.**
Paso 7: Evaluación de la Ristra y asignación a la hormiga en curso

Esquema representación

3.5 Mejora local

A pesar de las grandes posibilidades que presentan los algoritmos ACO, estos se ven limitados entre otras cosas por el tamaño de las subcolonias. Esto hace que en algunas ocasiones, el método de resolución no logre obtener la solución óptima aunque la distancia a la que se queda es muy pequeña.

Para solucionar este problema, se ha añadido a ciertas hormigas una mejora local con el objetivo de mejorar la calidad de las soluciones (y por lo tanto el rastro) acelerando así el proceso de búsqueda.

Dada que aplicar esta mejora a todas las hormigas de una subcolonia que tiene resultados muy prometedores (función objetivo igual a la mejor solución encontrada) implica un alto coste de tiempo y recursos, se ha aplicado ésta solamente a las hormigas que igualan o mejoran la solución de mayor calidad. Además, para evitar un estancamiento en óptimos locales, cada cierto número de subcolonias se escoge al azar una hormiga para aplicarle la mejora.

Paso 1: Inicialización
- Paso 1.0: Entrada de datos
- Paso 1.1: Cálculo de cotas del problema
- Paso 1.2: Cálculo de la matriz de pesos
- Paso 1.3: Inicialización del rastro

Paso 2: **mientras** no se alcance la condición de fin
- Paso 2.1: Creación de la subcolonia
- Paso 2.2: Actualización de los rastros
- Paso 2.3: **si se iguala la mejor solución o la hormiga es escogida al azar**
  - Paso 2.3.1: Realizar mejora local en la hormiga
  - Paso 2.3.2: Actualización de rastros
- Paso 2.4: si se alcanza mejor solución
  - Paso 2.4.1: Añadir rastro extra
  - Paso 2.4.2: Consolida la matriz de rastros. **Volver a paso 2**
- Paso 2.5: si el rastro es muy débil
  - Paso 2.5.1: Recuperar última matriz consolidada. **Volver a Paso 2**

Esquema del algoritmo de mejora

El funcionamiento de la mejora consiste en intercambios de piezas consecutivas de la secuencia encontrada, dando por válidos solo aquellos cambios que cumplen las reglas de precedencia del problema. Los intercambios se producen sin vuelta a empezar; es decir, que aunque se obtiene una solución mejor continúa los intercambios sin volver a la posición inicial. Se escogerá la nueva solución la mejor secuencia producida a raíz de los estos intercambios.

4 El problema de equilibrado de línea

4.1 Introducción al problema

Una característica importante de la industria manufacturera es el diseño de los procesos producción. La distribución en planta orientada al producto se adopta en los sistemas cuando la producción de los elementos es en masa o en grandes series y la organización es continua o repetitiva. El caso más característico es el de las cadenas de montaje.

Una línea de producción consiste en una secuencia ordenada de tareas o elementos de trabajo. Cada tarea es un conjunto de operaciones elementales que deben hacerse a la vez (precisan de la misma herramienta, utilizan la misma guía, etc...). Por lo tanto las tareas no pueden subdividirse y deben ser asignadas a una misma estación, atendida por un operario, un equipo o, incluso, un robot. Cada estación, por tanto, tendrá asignado un cierto subconjunto de operaciones.

Las unidades de producto, a medida que se elaboran, se transfieren de una estación a otra de forma ordenada. La trayectoria de los materiales puede tener formas diversas, pero lo esencial es que no hay retrocesos. El flujo de productos en fase de elaboración se desplazan por la cadena recibiendo de cada estación una cierta cantidad de trabajo.

El problema general del equilibrado de una línea de montaje se suele plantear después del diseño y montaje de la propia línea. Este diseño fijará varios parámetros del problema como pueden ser los tiempos de las tareas y el diagrama de precedencias.

Las decisiones propias del equilibrado se refieren a la posibilidad de dividir el flujo de trabajo lo suficiente y asignar las diferentes tareas elementales a las distintas estaciones de trabajo. Estas tareas se deben ejecutar sin interrupción y, por consiguiente, en una misma estación para que el personal y los equipos sean utilizados de la forma más eficiente posible a lo largo de toda la línea.

En el caso más frecuente, una de las operaciones requiere más tiempo para ser ejecutada, convirtiéndose en el cuello de botella, que restringirá la capacidad de todo el proceso. El equilibrado de líneas se realiza para ajustar y equilibrar las cargas de todas las estaciones de tal forma que se eviten estos picos de carga en las estaciones, esto se muestra con una menor tenencia de tiempos muertos. Para lograr esto, las estrategias para el equilibrado de una línea son minimizar el número de estaciones y/o minimizar el tiempo de ciclo.

Si se fija, o se conoce, el tiempo que dispone cada estación para realizar las tareas que se le han asignado (tiempo de ciclo) se trabaja para minimizar el número de estaciones. La relación entre el tiempo requerido para la fabricación y el tiempo consumido para ello definirá la eficiencia de la cadena. Otra opción es establecer cual debe ser el mínimo tiempo de ciclo, si se conoce el número de estaciones.

4.2 El problema de SALPB-E

El problema de equilibrado de líneas de montaje ALBP (*Assembly Line Balancing Problem*), según la clasificación propuesta por Baybars en 1986, puede agruparse en dos grandes categorías: SALBP (*Simple Assembly Line Balancing Problem*) y GALBP (*General Assembly Line Balancing Problem*).

El primer problema consiste en diseñar un conjunto de estaciones de trabajo, con idéntico tiempo de ciclo o tasa de producción, cada tarea sólo es asignable a una estación y éstas deben respetar las relaciones de precedencia; la función objetivo propuesta admite tres variantes:

(1) Minimizar el número de estaciones para una tasa de producción prefijada; problema conocido como SALBP-1.

(2) Minimizar el tiempo de ciclo asignado a cada operario para un número de estaciones establecido: SALBP-2.

(3) Minimizar el tiempo muerto tiendo en cuenta las diferentes combinaciones posibles del tiempos de ciclo y del número de estaciones: SALBP-E.

La segunda categoría, GALBP, incluye el resto de problemas. En ellos cabe considerar estaciones en paralelo, agrupaciones de tareas, incompatibilidades entre tareas, otras funciones objetivo, etc.

La resolución exacta de SALBP se ha abordado mediante diversos procedimientos, entre ellos los basados en *branch and bound*, como en *Hoffmann (1992)*, aunque su aplicación sólo es viable, en general, para ejemplares de dimensiones reducidas debido al carácter NP-hard del problema. La resolución de ejemplares con las dimensiones requeridas por la industria de automoción es conveniente enfocarla a través de procedimientos heurísticos, tanto para el problema SALB-2*, Ugurdag et al. (1997)*, como para el SALBP-1, *Boctor (1995).*

Para la resolución de este último problema, destacan las heurísticas *greedy* basadas en el empleo de reglas de prioridad que condicionan el orden de asignación de las tareas a las estaciones de trabajo. Las reglas se refieren o combinan aspectos tales como el tiempo de proceso de cada tarea, número de tareas siguientes a una concreta, cotas sobre el número mínimo de estaciones necesario para completar la asignación de las tareas, etc. Dichas reglas sirven para establecer una ordenación de las tareas en cada iteración, con el propósito de seleccionar la más apropiada (según la regla) entre un conjunto de candidatas compatibles con la parte de la solución ya construida; las tareas candidatas son, evidentemente, aquéllas cuyas precedentes han sido asignadas y su tiempo de proceso no supera el tiempo disponible en la estación receptora. Usualmente, cada heurística de este tipo tiene asociada una sola regla, y la regla determina (salvo que se emplee el sorteo) la tarea que debe asignarse en cada paso. La aplicación de este tipo de algoritmos heurísticos suele ofrecer soluciones aceptables, en promedio, tanto más cuanto mayor sea el número de aspectos que combina la regla; no obstante, no puede concluirse que exista una única regla que supere a todas las demás ante cualquier ejemplar de problema; por otra parte, salvo que se incorpore el azar al procedimiento, siempre ofrecerán las mismas soluciones.

Una segunda clase de heurísticas, constituida por los algoritmos GRASP (*Greedy Randomized Adaptative Search Procedure*), permite generar distintas soluciones gracias a la incorporación del azar al procedimiento. En efecto, en cada iteración se establece una selección de tareas, que está condicionada por una regla de prioridad, entre el conjunto de tareas candidatas; las tareas seleccionadas se someten a un sorteo que puede depender o no de una regla de prioridad. El procedimiento se ha empleado con éxito en diversas aplicaciones de la Organización Industrial, tal como aparece en la recopilación de *González en Díaz et all.* (1996); para el problema que nos ocupa, se han obtenido resultados satisfactorios en *Bautista et al.* (2000).

Finalmente, una tercera clase de heurísticas es los métodos de búsqueda local o procedimientos de exploración de entornos, *Díaz et al.* (1996), entre ellos se encuentran: los procesos de escalado (HC), recocido simulado (SA), búsqueda tabú (TS) y los algoritmos genéticos (GA). Esta clase de heurísticas proporciona vías alternativas para buscar soluciones en un espacio delimitado por la definición de un vecindario. No obstante, si la definición del vecindario es general y válida para cualquier problema combinatorio, se pierde entonces el conocimiento sobre el problema específico que sí es tenido en cuenta por las heurísticas *greedy*.

En *Bautista et al.* (2000) se proponen dos procedimientos que consideran simultáneamente el principal aspecto positivo de la primera familia de heurísticas y el principal aspecto interesante de las otras dos: 1) el conocimiento sobre el SALBP que ofrecen las reglas de prioridad específicas del problema y 2) la posibilidad deseable en todo problema combinatorio de generar soluciones en el espacio de búsqueda. Las aplicaciones concretas en dicho trabajo son: un algoritmo GRWASP (*Greedy Randomized Weighted Adaptative Search Procedure*) y un algoritmo genético que genera y explora soluciones en el espacio de las reglas heurísticas.

4.3 Particularidades del ACO asociado

Dado el esquema general del ACO desarrollado se puede ver que hay ciertos aspectos o etapas que deben desarrollarse de manera diferente según el problema que se desea tratar.

El primero de los aspectos propios que es necesario definir para el caso del SALBP-E es el cálculo de las cotas. La cota superior del problema pueden ser cualquier solución factible, mientras que para el cálculo de las inferiores se ha llevado a cabo de 4 maneras diferentes escogiendo luego la mejor de ellas:

- Cota1: el valor corresponde a considerar el tiempo total de ejecución de las tareas; es decir, se considera que las operaciones se realizan sucesivamente en una única estación.
- Cota2: Toma por valor el resultado de multiplicar el número mínimo de estaciones por la operación más larga.
- Cota3: Se trata de establecer conjuntos para varias combinaciones de estaciones, donde se mezclen operaciones con tiempos de operación diferentes. La asignación de las tareas a un conjunto se realiza por orden decreciente de tiempos, haciendo que para aquellos conjuntos con tiempos grandes haya menos elementos. El mayor tiempo de ciclo de estos conjuntos multiplicado por el número de estaciones, es la cota. (para más información *Scholl (1999)*).
- Cota4: Para cada tarea, se calculan las estaciones necesarias para realizar las operaciones predecesoras y las sucesoras. La cota se obtiene calculando el tiempo máximo de estas estaciones y multiplicando por el número total de estaciones. (*Scholl (1999)*)

4.4 Experiencia computacional

El estudio del comportamiento del SALBP-E se ha dividido en dos partes. Por una lado se ha realizada la experiencia computacional fijando el número de secuencias a evaluar y por el otro se ha establecido el tiempo máximo de las heurísticas para estudiar el problema.

Se han estudiado 256 juegos de datos con diferentes grafos de precedencia y/o diferentes combinaciones del número mínimo y máximo de estaciones. Cada uno de los juegos de datos se ha solucionado 10 veces y cada solución ha sido considerada como un problema diferente.

De los 256 juegos de datos se han establecido 4 grupos de estudio, en función del número de tareas. En el grupo I están los grafos de entre 7 y 30 tareas, en el grupo

II están los que oscilan entre 32 y 58 tareas, el grupo III lo forman los de 70 a 89 tareas, el grupo IV entre 94 y 148 tareas y en el grupo V 297 tareas.

Los limites se han fijado en 84,000 hormigas (600 subcolonias) y el límite de tiempo en 1 minuto para el grupo I, 2 para el grupo II, 4 para el grupo III y 8 para el grupo iV y V.

Para la comparación de los resultados se ha utilizado un procedimiento denominado *multi-start*. Básicamente, consiste en construir secuencias de tareas de forma aleatoria a las cuales se le aplica una mejora local mediante intercambios lineales.

Los parámetros calculados son el porcentaje de óptimos, la discrepancia mínima, media y máxima y el tiempo medio para obtener el resultado.

Grupo	Tipo de prueba	Nº óptimos (%)	% discrepancia al óptimo			Tiempo medio (s)
			Min	Media	Máx	
I	Multi-start	70,7	0,14	0,31	0,73	23
	Hormigas	72,4	0,06	0,32	0,55	26
	Hormigas + mejora	79,3	0,06	0,20	0,42	23
II	Multi-start	45,4	0,85	1,24	1,72	40
	Hormigas	44,6	0,76	1,35	1,82	68
	Hormigas + mejora	54,1	0,49	0,91	1,43	59
III	Multi-start	13,6	1,31	1,91	2,46	67
	Hormigas	16,7	1,34	2,07	2,74	159
	Hormigas + mejora	21,0	0,83	1,28	1,81	109
IV	Multi-start	0,8	2,71	3,32	3,90	119
	Hormigas	2,2	2,64	3,49	4,24	392
	Hormigas + mejora	2,4	1,78	2,54	3,27	208
V	Multi-start	0,0	2,04	2,46	2,82	371
	Hormigas	0,0	1,83	2,30	2,78	884
	Hormigas + mejora	0,0	1,36	1,96	2,46	675

Resultados con limitación en el número de secuencias

Grupo	Tipo de prueba	Nº óptimos (%)	% discrepancia al óptimo			Tiempo medio (s)
			Min	Media	Máx	
I	Multi-start	78,1	0,11	0,17	0,30	44
	Hormigas	70,5	0,19	0,35	0,47	20
	Hormigas + mejora	76,4	0,06	0,24	0,42	48
II	Multi-start	51,3	0,76	1,04	1,35	97
	Hormigas	45,9	0,69	1,31	1,86	68
	Hormigas + mejora	53,1	0,48	0,88	1,36	106
III	Multi-start	19,2	1,14	1,49	1,97	216
	Hormigas	17,0	1,26	2,02	2,93	208
	Hormigas + mejora	20,7	0,82	1,21	1,70	238
IV	Multi-start	1,0	2,46	2,98	3,42	475
	Hormigas	2,0	2,67	3,48	4,26	472
	Hormigas + mejora	1,4	1,65	2,41	3,10	479
V	Multi-start	0,0	1,72	2,14	2,48	480
	Hormigas	0,0	1,68	2,08	2,48	480
	Hormigas + mejora	0,0	1,30	1,79	2,27	480

Resultados con limitación en el tiempo

5 El caso de flow-shop sin pulmones

5.1 Introducción al problema

El problema de taller mecánico consiste en considerar un flujo de n piezas o elementos que deben ser procesados según unas rutas a través de m máquinas o recursos. La resolución del problema consiste en encontrar la secuencia de producción que optimice una función objetivo referida a la característica principal que se desea que cumpla el sistema.

Los problemas de taller mecánico admiten un gran número de clasificaciones. Una de ellas se refiere al movimiento de las piezas dentro del sistema. Cuando el flujo de las piezas es regular; es decir, las piezas siguen todas la misma ruta, el problema se conoce como *flow-shop*, mientras que si cada pieza sigue una ruta diferente el problema es de tipo general o *job-shop*.

El número de soluciones, el tamaño del espacio de exploración, varía según el tipo de flujo. En el caso de flow-shop el número de secuencias posibles se encuentra limitado a *n!*; en el caso de *job-shop* está en $(n!)^m$, por lo que se puede intuir la dificultad de resolución de estos problemas por métodos de exploración de todas las soluciones. En este tipo de problemas también se utilizan los métodos de búsqueda local descritos para los problemas de equilibrado de líneas.

El caso concreto que se ha estudiado con el algoritmo ACO no solo incluye la consideración presentada sobre el flujo de los materiales dentro del sistema sino que también añade la restricción referente al espacio existente entre dos máquinas o recursos. Este espacio representa la cantidad de piezas que pueden estar a la espera de ser procesada. En concreto los resultados obtenidos son para el caso de *flow-shop* sin espacio intermáquina o sin pulmones.

De manera resumida, las características del problema son:

- Se trata de un problema de flow-shop por lo que todas las piezas siguen la misma secuencia o ruta. Además el tipo tratado tiene la particularidad que la permutación o el orden de producción se mantiene para todas las máquinas. Es decir, si en la primera máquina procesa las piezas en un determinado orden, éste debe mantenerse para todo el resto de máquina.
- El número de piezas y máquinas del sistema está disponible en el estado inicial y es conocido.
- Las máquinas tienen una capacidad de una pieza y las operaciones deben realizarse de manera consecutiva; no se permiten dos operaciones en paralelo para una misma pieza. Además el número de máquinas de cada tipo es uno.
- Si cuando una máquina ha finalizado su operación, la máquina siguiente aún esta trabajando, la primera queda bloqueada hasta que la pieza pueda desplazarse. Esto indica que no existe espacio posible de almacenaje entre dos máquinas.

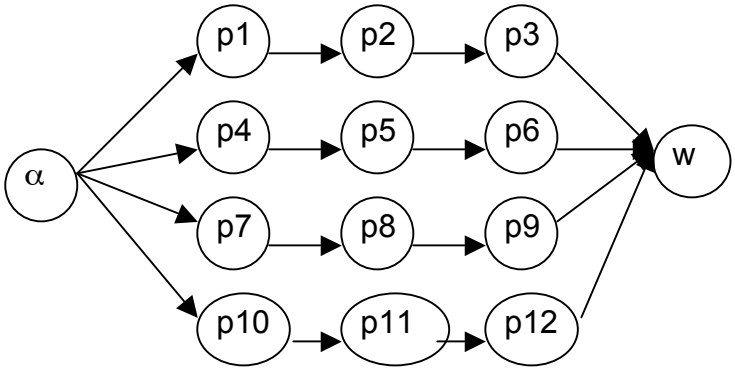
5.2 Particularidades del ACO asociado

El primer aspecto que se ha desarrollado con la heurística ACO aplicada en este problema concreto es el tratamiento de las soluciones. Dadas las características del problema se presentaban dos opciones de planteamiento. Como el orden de las

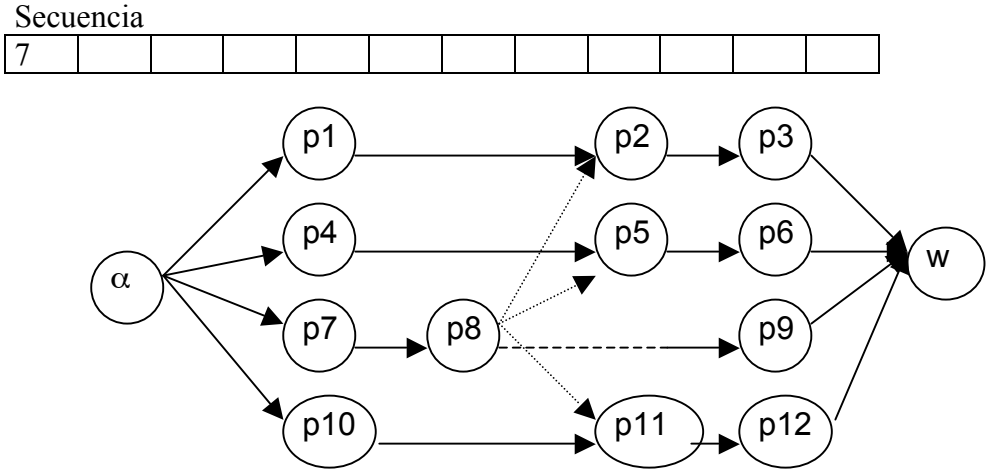
piezas se mantienen constante para todas las máquinas, se podría considerar únicamente el problema equivalente a encontrar una única secuencia de n piezas que tuviera en cuenta los aspectos de las m máquinas, únicamente a la hora de realizar la evaluación. Por otro lado, la otra opción posible era establecer una secuencia de n·m elementos donde se tuviera en cuenta el mantenimiento del orden según las reglas de precedencia. A pesar que la primera opción era mucho más simple, se creyó más conveniente utilizar el segundo planteamiento ya que con menos reajusten en el algoritmo se puede utilizar el mismo algoritmo para problemas más generales.

En la formulación propuesta, 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 unan el problema se resuelve según el grafo representado donde ahora se consideran 12 tareas en lugar de 4.

	M1	M2	M3
P1	*p1*	*p2*	*p3*
P2	*p4*	*p5*	*p6*
P3	*p7*	*p8*	*p9*
P4	*p10*	*p11*	*p12*



Para el caso en que las secuencias se mantienen, se debe introducir además la modificación dinámica de la matriz de precedencias. Esto significa que 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. Para el ejemplo anterior:

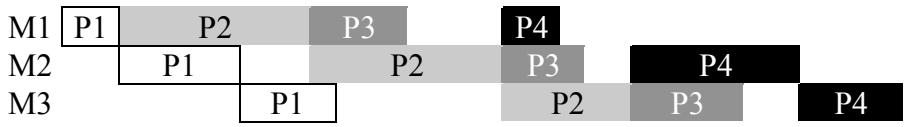


De esta manera se garantiza que en la máquina 2 la primera operación que se debe realizar es la p8, que corresponde con la primera operación en la máquina 1.

Otra de las modificaciones que deben hacerse para adaptar el algoritmo ACO al problema de flow-shop es la formulación de las cotas propias del problema. En este caso se ha considerado para la cota superior, igual que en el caso del equilibrado de líneas, el valor de la función objetivo de una solución factible. Para el caso de las cotas inferiores se ha tomado:

- Cota1: El valor promedio de los tiempos de utilización de las máquinas sin tener en cuenta las duraciones de los periodos en que la máquina está desocupada; esto es sumar todas las duraciones de las piezas y dividirlo por el número de máquinas.
- Cota2: Considerar la cota de la primera máquina como la suma de las duraciones en ésta añadiendo la duración de la pieza que menos tarda en realizar las operaciones en el resto de máquinas
- Cota3: Considerar la cota de la última máquina calculándola como la cota2 pero aplicando a la última máquina.
- Cota4: Considerar la cota a la máquina situada en la posición media del total de las máquinas.

La evaluación de la solución se lleva a cabo considerando las propiedades del problema en que a los tiempos en que las máquinas hay que añadir la duración de los períodos en que la máquina está en espera o bloqueada por el efecto de las máquinas vecinas. De manera gráfica:



Ejemplo de evaluación de la función objetivo

5.3 Experiencia computacional

La experiencia computacional que se ha llevado a término es la continuación de la aparecida en Companys et al. (1999) y en De la Rosa (2000). En ambos casos 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, el error medio y el máximo error para cada una de las colecciones.

Los algoritmos que se utilizaran para la comparación se hayan descritos en De la Rosa (2000), aunque se han cogido los métodos con mejores resultados, además de los métodos Greedy clásicos.

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

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%

Comparación según el error medio

Piezas	10		12		15	
Máquinas	3	5	4	5	3	4
Palmer	53,79%	52,47%	49,73%	48,74%	50,77%	42,42%
Teixido	31,97%	25,12%	36,19%	28,64%	34,83%	28,19%
Gupta	28,68%	39,90%	37,20%	35,00%	32,99%	30,29%
Trapecios	30,56%	21,80%	24,62%	21,93%	37,84%	26,36%
Palmer + RAES	19,73%	13,19%	11,92%	13,76%	11,92%	11,93%
Teixido + RAES	13,07%	9,28%	9,05%	9,52%	11,06%	11,26%
Gupta + RAES	13,07%	9,79%	14,02%	9,96%	10,81%	10,36%
Trapecios + RAES	10,83%	9,90%	9,95%	8,96%	12,83%	11,26%
Búsqueda Tabú	50,00%	9,50%	10,00%	8,72%	8,37%	7,97%
ACO	14,77%	15,51%	12,05%	11,59%	15,05%	12,55%
Aco + mejora	2,70%	0,00%	0,00%	0,00%	4,62%	2,22%

Comparación según el error máximo

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 de el ACO con la mejora incorporada, puede apreciarse como ésta se destaca de los resultados. Este métodos obtiene, en el peor de los casos un 96% de óptimos.

Otro factor que no se ha añadido ya que no se dispone de los datos exactos es el tiempo empleado para la resolución. En el caso del algoritmo ACO y el ACO con mejora puede verse como el tiempo de resolución se reduce espectacularmente en el segundo caso. En la tabla se presenta el tiempo medio de resolución por problema (cada colección está formada por 1000 problemas)

Piezas	10		12		15	
Máquinas	3	5	4	5	3	4
ACO	6,837	18,448	23,26	35,019	23,909	48,814
ACO + mejora	0,772	2,222	2,29	3,396	3,947	4,171

6 Conclusiones

Los algoritmos ACO (*Ant Colony System*) se presentan como un campo de investigación novedoso y prometedor. Como se ha comentado, esta meta-heurística pertenece al grupo de las de búsqueda local, como pueden ser los algoritmos genéticos, el recocido simulado, la búsqueda Tabú o las redes neuronales que se caracterizan por utilizar analogías o principios básicos de fenómenos naturales.

Este nuevo método de búsqueda se basas en un proceso autocatalítico y distribuido. La idea general de este tipo de algoritmos es guiar a una población de agentes mediante un proceso dirigido por un criterio heurístico. Si los individuos no reaccionan entre ellos, el proceso autocatalítico y el criterio heurístico hacen que el individuo converja hacia soluciones subóptimas en tiempo exponencial. Cuando los agentes interaccionan entre ellos, el criterio heurístico encamina el proceso y facilita la convergencia hacia soluciones de gran calidad evitando los óptimos locales. Ninguna solución se excluye completamente, aunque la probabilidad de explorar vecindarios de poca calidad se reduce.

Las principales contribuciones de estos algoritmos está en el efecto del *feed-back* positivo como herramienta para resolver problemas de optimización explotando la capacidad de utilizar la información obtenida en el pasado. Además, una de las principales ventajas de este tipo de algoritmos, es la facilidad para la implementación en diferentes problemas de optimización ya que se basa en una idea general y sencilla y la adaptación se precisa en aspectos muy localizados.

De la resolución presentada se puede decir que para los problemas del SALBP-E no es sencillo encontrar métodos heurísticos para poder contrastar las soluciones. Existen algunos métodos constructivos y heurísticos como en *Scholl (1999)* o *Bautista et al.* (2000) que resuelven los problemas de equilibrado de líneas, pero no se ha podido disponer de ningún estudio para el SALBP-E. Por esta razón se ha optado por la implementación de otro método sencillo de implementar y rápido como es el multi-start.

Sin embargo, para el caso de *flow-shop sin pulmones* no ocurre lo mismo. En este caso se disponen de un gran número de métodos de resolución. En este trabajo se han escogido los métodos Greedy clásicos, los métodos Greedy con mejora y un muestra de las heurísticas con la Búsqueda Tabú procedente de Bernadó (1999).

Dado que los algoritmos de hormigas al tipo de meta-heurísticas en que la fijación de los parámetros influye de manera notable en el funcionamiento, de manera que hay que ajustar los parámetros para garantizar un comportamiento adecuado. Para ajustar estos parámetros se han realizado pruebas con más de 500 combinaciones.

El objetivo no era buscar la combinación óptima, sino el rango de valores donde las hormigas se comportaran de manera robusta y eficiente independientemente del juego de datos. Este comportamiento se ha encontrado para la combinación de datos:

alfa	beta	evaporación	rastro Inicial	rastro Extra
0.75	0.25	0.01	100	20

Entrando en el análisis de los problemas, para el caso de las pruebas con el número de secuencias fijadas se puede observar que el algoritmo para el SALBP-E sin mejora local obtiene, de forma general, unos resultados simulares a los de la heurística *multi-start* de referencia.

Comparando estos resultados con los obtenidos en otros trabajos sobre algoritmos de hormigas a problemas combinatorios, y donde se ha utilizado de referencia el mismo algoritmo (*Gambardella et al (1999))* se puede concluir que el algoritmo de hormigas general obtiene buenos resultados. Estudiando el resultado obtenido por el mismo algoritmo aplicando la mejora, puede verse como éste ha obtenido mejores resultados en todos los parámetros de comparación.

En relación a la evolución de las soluciones, como era de esperar el tiempo crece de forma exponencial al número de tareas del juego de datos. Además, fijado el número de secuencias el método más rápido es siempre el multi-start.

En las pruebas realizadas para el equilibrado de líneas con el tiempo limitado puede verse como los tres métodos presentan mejores resultados al tener más capacidad para explorar soluciones (el tiempo es mayor). La heurística que presenta mayor descenso en sus discrepancias al óptimo es el multi-start. Esto se debe a que las hormigas han llegado al límite de exploración mientras que en el otro caso, debido al alto componente aleatorio puede explorar otros espacios de soluciones. Aún así, el algoritmo de hormigas con mejora local sigue siendo superior al resto en todos los resultados. Concretamente, su discrepancia media al óptimo es un 16% mejor a la obtenida por el multi-start.

En relación al otro problema de estudio, el caso de flow-shop sin pulmones, puede observarse que las soluciones del algoritmo de hormigas sin mejora, son algo peores a las obtenidas mediante la Búsqueda Tabú, aunque se encuentre en el mismo orden de magnitud para el número de óptimos. Para el algoritmo con la mejora local, puede verse que los resultados obtenidos son bastante espectaculares ya que de 1000 problemas por colección, en 3 de ellas los encuentra mientras que en el resto no encuentra un máximo de 36 óptimos.

Es curioso el comportamiento de la heurística propuesta ya que si se observan los resultados para un número de piezas fijo, en general son algo mejores los resultados de los problemas con mayor número de máquinas. La lógica sugeriría que cuanto menor es el número de tareas de la secuencia (recordemos que en este caso la secuencia consta de $n \cdot m$ tareas, siendo n el número de piezas y m el número de máquinas) mejor serán los resultados al ser menor el espacio de exploración. No obstante, en el problema concreto que se resuelve el número de máquinas no implica una mayor dificultad ya que las precedencias “dinámicas” ya marcan las secuencias de todas las máquinas, una vez fijada la secuencia en la primera.

Por otro lado puede verse como, lógicamente, el incremento en el número de piezas para un número de máquinas constante implica un empeoramiento de las soluciones. En este caso no existe ninguna relación entre las piezas por lo que la lógica se puede aplicar sin problemas.

7 Referencias

BAUTISTA J., COMPANYS R. y COROMINAS A. (1995). “Sequenciació d’unitats en context JIT”. Colección TOE nº 9, Edicions UPC.

BAUTISTA J., MATEO M., FERRER R., PEREIRA J. y COMPANYS R. (2000). “The assembly line balancing problem solved by hybrid heuristic procedures and driven exploration”.POMS, Sevilla 2000.

BAUTISTA J., SUÁREZ R., MATEO M. y COMPANYS R. (2000). “Local Search Heuristics for the Assembly Line Balancing Problem with Incompatibilities Between Tasks”. Proceedings of the 2000 IEEE International Conference on Robotics and Automation, ICRA 2000, San Francisco, CA, USA, April 24-28, 2000, pp. 2404-2409.

BERNADÓ, D. (1999) Resolució del problema de flow-shop sense pulmons intermàquina mitjançant cerca tabú. PFC- ETSEIB-UPC, director R. Companys

BRETÓN J., FERNÁNDEZ J.A. y BAUTISTA J. (2000). “Resolución del problema SALBP-E mediante procedimientos heurísticos y la programación lineal entera”. DIT-OE-ETSEIB-UPC.

BRETÓN, J.; FERNÁNDEZ, J. (2000) “Aplicación de los algoritmos de hormigas para la resolución del problema de equilibrado de líneas de montaje SALBP-E y aplicación a un caso real” PFC-ETSEIB-UPC, director J. Bautista.

BULLNHEIMER B., HARTL R. F. y STRAUSS C. (1997). “A new Ranked Based Version of the Ant System-Computational Study”. Technical report, University of Viena, Institute of Management Science.

CARRERAS-CANDI M. Y DOMINGO E. (1997). “Una aplicación de los algoritmos GRASP y genéticos al problema de equilibrado de líneas de montaje. Resolución de un caso de empresa”. PFC ETSEIB-UPC director J. Bautista.

COLORNI A., DORIGO M. y MANIEZZO V. (1992a). "Distributed Optimization by Ant Colonies". Proceedings of the First European Conference on Artificial Life, Paris, France, F.Varela and P.Bourgine (Eds.), Elsevier Publishing, 134-142.

COLORNI A., DORIGO M. y MANIEZZO V. (1992b). "An Investigation of Some Properties of an Ant Algorithm. Proceedings of the Parallel Problem Solving from Nature Conference" (PPSN 92), Brussels, Belgium, R.Männer and B.Manderick (Eds.), Elsevier Publishing, 509-520.

COLORNI A., DORIGO M., MAFFIOLI F., MANIEZZO V., RIGHINI G., TRUBIAN M. (1996). "Heuristics from Nature for Hard Combinatorial Problems". International Transactions in Operational Research, 3(1):1-21.

COMPANYS R., COROMINAS A. (1994). “Organización de la producción II. Dirección de operaciones 4”. Edicions UPC, colección Aula ETSEIB.

COMPANYS, R; MATEO, M; BAUTISTA, J. (1999) Flow-shop sin pulmones. *III Jornadas de Ingeniería de Organización*

De la ROSA, M. (2000) “Flow-shop sin pulmones: Síntesis y comparación de métodos de resolución” PFC- ETSEIB-UPC, director J. Bautista

DI CARO G. y DORIGO M. (1997). "Ant Net: A mobile agents approach to adaptative routing”.Technical Report 97-12, IRIDIA, Université Libre de Bruxelles.

DOMINGUEZ J.A., ALVAREZ J.M, GARCIA S., DOMINGUEZ M.A. y RUIZ A. (1995). “Dirección de operaciones. Aspectos estratégicos en la producción y los servicios”. Ediciones McGrawHill.

DORIGO M. y 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.

DORIGO M., DI CARO G. y GAMBARDELLA L.M. (1999). "Ant Algorithms for Discrete Optimization". Artificial Life, 5(2):137-172

DORIGO M., MANIEZZO V. y COLORNI A. (1996). "The Ant System: Optimization by a Colony of Cooperating Agents". IEEE Transactions on Systems, Man, y Cybernetics-Part B, 26(1):29-4.

FERRER R. (2000). “Resolución del problema de equilibrado de líneas de montaje mediante procedimientos híbridos con algoritmos heurísticos y de exploración dirigida”. PFC. ETSEIB-UPC, director J. Bautista.

FERRER R., BAUTISTA J. (1999). “Modelización y resolución de problemas de equilibrado de líneas de montaje con PLE”. Document Intern de Treball DT-I-1999/02-, IOC, ETSEIB-UPC.

GAMBARDELLA L.C. y DORIGO M. (1999). “An ant colony system hybridized with a new local search for the sequential ordering problem”. Aceptado para publicación en Informs Journal on Computing, marzo 2000.

MONMARCHÉ, N; VENTURINI, G; SLIMANE, M. (2000) “On how *Pachycondyla apicalis* ants suggest a new research algorithm” Future Generation Computer System 16, pág. 937-946.

PLANS J. (1999). ”Classificació, modelització i resolució dels problemes de disseny i asssignació de tasques en linies de producció”. Tesis doctoral DOE-UPC

SCHOLL A. (1999). “Balancing and Sequencing of Assembly Lines”. Heidelberg Physica-Verlag cop.

STÜTZLE T. y DORIGO M. (1999). "ACO Algorithms for the Traveling Salesman Problem". In K. Miettinen, M. Makela, P. Neittaanmaki, J. Periaux, editors, Evolutionary Algorithms in Engineering and Computer Science, Wiley, 1999.

STÜTZLE T. y HOOS H. (1997). "Improvements on the ant system: Introducing *MAX-MIN* ant system”. In Proceding of the International Conference on Artificial Neural Networks and Genetic Algorithms, pag. 245-249. Springer Verlag, Wien.

STÜTZLE T. y HOOS H. (1997). "The *MAX-MIN* ant system and local search for the Traveling Salesman Problem". In T. Baeck, Z. Michalewicz, y X. Yao, editors, Proceedings of IEEE-ICEC-EPS’97, IEEE International Conference on Evolutionary Computation and Programming Conference, pag. 309-314. IEEEE Press.

VALERO J. (1991). “Modelos y algoritmos de PLE para el diseño y equilibrado de líneas de producción y montaje”. PFC- ETSEIB-UPC, director A. Corominas.

Anexo I: Nomenclatura ACO

τ_{ij}: cantidad de rastro desde la tarea i hasta la j.

η_j: peso de la tarea j según el criterio heurístico.

u_{ij}: peso teniendo en cuenta tanto la componente heurística como histórica de la tarea j para realizarse después de i.

v(j): Peso del criterio heurístico para la tarea j.

p_{ij}: probabilidad de escoger la tarea j después de la i.

α: parámetro para determinar el peso de la memoria histórica en el peso de la decisión.

β: parámetro para determinar el peso de la heurística en el peso de la decisión.

Q: número de veces que la nueva solución es mejor que la anterior.

d(%): discrepancia de la función objetivo respecto a la mejor cota del problema.

n: número de productos.

m: número de máquinas o recursos.

i: última tarea secuenciada.

ANEXO II: Nomenclatura y reglas de los problemas SALB

i, j	Índice de tarea
N	Número de tareas
C	Tiempo de ciclo
T_i	Duración de la tarea i
IS_i	Conjunto de tareas siguientes inmediatas a la tarea i
S_i	Conjunto de tareas siguientes a la tarea i
TP_i	Conjunto de tareas precedentes a la tarea i
L_i	Nivel de la tarea i en el grafo de precedencias

Nombre	***Peso***
1. Longest Processing Time	$v(i)=t_i$
2. Greatest number of Immediate Successors	$v(i)=\|IS_i\|$
3. Greatest number of Successors	$v(i)=\|S_i\|$
4. Greatest Ranked Positional Weight v	$v(i)=t_i \pm \Sigma\, t_j\ (j \in S_i)$
5. Greatest Average Ranked Positional Weight	$v(i)= (t_i \pm \Sigma\, t_j\ (j \in S_i))/(\|S_i\|+1)$
6. Smallest Upper Bound	$v(i)=-UB_i=-n-1+[(t_i \pm \Sigma\, t_j\ (j \in S_i))/C]^+$
7. Smallest Upper Bound / Number of Successors	$v(i)= -UB_i/(\|S_i\|+1)$
8. Greatest Processing Time / Upper Bound	$v(i)=t_i/ UB_i$
9. Smallest Lower Bound	$v(i)= -LB_i=-[(t_i \pm \Sigma\, t_j\ (j \in TP_i))/C]^+$
10. Minimum Slack	$v(i)=-(UB_i- LB_i)$
11. Maximum Number of Successors / Slack	$v(i)= \|S_i\|/(UB_i- LB_i)$
12. Bhattcharjee & Sahu	$v(i)= t_i \pm \|S_i\|$
13. Etiquetas Kilbridge & Webster	$v(i)=t-L_i$