Universidad Central “Martha Abreu” de Las Villas

Facultad de Matemática, Física y Computación

Trabajo de diploma en Ciencia de la Computación

Aplicación de la Metaheurística ACO en II Etapas al Problema de Planeamiento de las Fuerzas de Trabajo

Autor: Adniel Simó Pimienta

Tutor: Msc. Amílkar Puris Cáceres

Santa Clara 2008

Declaración de autoría

Hago constar que el presente trabajo fue realizado en la Universidad Central "Martha Abreu" de Las Villas como parte de la culminación de los estudios de la especialidad de Ciencias de la Computación, autorizando a que el mismo sea utilizado por la institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos ni publicado sin la autorización de la Universidad.

_____

Firma del autor

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdos de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____ _____

Firma del tutor **Firma del Jefe del Seminario**

Dedicatoria

A mis padres.

Agradecimientos

A mi tutor, Amilkar, que siempre tuvo tiempo para atenderme cuando más difícil se me tornaba todo.

A la familia Arcelo Gutiérrez, que me ayudaron en todo lo que estuvo a su alcance para confeccionar este trabajo.

A Yai, que me ha soportado durante todo este tiempo.

A toda mi familia, muchas gracias por confiar siempre en mí.

A todos mis amigos y compañeros, que me apoyaron en el trascurso de la carrera, dentro y fuera de la UCLV, son tantos que tengo miedo dejar de mencionar alguno.

Resumen

En el presente trabajo se hace un estudio de todos los aspectos relacionados con el Problema de las Fuerzas de Trabajo (WPP) para modelar el mismo de forma tal que sea posible aplicarle la metaheurística “Optimización Basada en Colonias de Hormigas, ACO”, además se investigó el efecto de aplicar el modelo “Opimización Basada en Colonias de Hormigas en II Etapas, TS-ACO”. Este enfoque difiere del modelo original en que presenta una nueva estrategia de búsqueda basada en II etapas, el proceso de búsqueda desarrollado por la hormigas se divide en dos, en la primera etapa se encuentran soluciones parciales que se comportan como estados iniciales en el proceso de búsqueda desarrollado en la etapa siguiente. Se escogió el algoritmo Max-Min Ant System (MMAS) para comparar el desempeño de ambos modelos en cuanto a la calidad de las soluciones obtenidas para el WPP.

Abstract

In this work a comparison is established between two different metaheuristics. One of them is the traditional Ant Colony Optimization (ACO). The other one is a new variant of the traditional and is named Two-Step ACO (TS-ACO). In order to establish the comparison we choose the Max-Min Ant System (MMAS), it was applied to the Workforce Planning Problem (WPP) in both variants of the metaheuristic. The goal is compare the acting of both models as for the quality of the solutions obtained for this problem.

Índice

Introducción

Al buscar en el diccionario el significado de la palabra optimizar, se encuentra con que optimizar, no es más que buscar el mejor modo de realizar una actividad, sin embargo, en términos científicos optimizar se define como el proceso de tratar de encontrar la mejor solución posible para un determinado problema.

En un *problema de optimización* existen diferentes soluciones y un criterio para discriminar entre ellas. De forma más precisa, estos problemas se pueden expresar en cómo encontrar el valor de unas *variables de decisión* para los que una determinada *función objetivo* alcanza su valor máximo o mínimo. El valor de las variables en ocasiones está sujeto a unas *restricciones*.

Una clasificación usual de los problemas de optimización es en Optimización Estocástica y Optimización Determinista, en este trabajo se trataran solo problemas de optimización combinatoria que caen dentro de la categoría de los problemas deterministas.

Estos problemas tienen como objetivo encontrar el máximo (o el mínimo) de la función objetivo sobre el conjunto finito de soluciones S. No se exige ninguna condición o propiedad sobre la función objetivo o la definición del conjunto S. Es importante notar que dada la finitud de S, las variables han de ser discretas, restringiendo su dominio a una serie finita de valores. Usualmente, el número de elementos de S es muy elevado, lo que hace impracticable la evaluación de todas sus soluciones para determinar el óptimo.

La idea intuitiva de problema “difícil de resolver” queda reflejada en el término científico NP-hard (Johnson 1979) utilizado en el contexto de la complejidad algorítmica. En términos coloquiales se puede decir que un problema de optimización difícil es aquel para el cual no se puede garantizar encontrar la mejor solución posible en un tiempo razonable. La existencia de una gran cantidad y variedad de problemas difíciles, que aparecen en la práctica y que necesitan ser resueltos de forma eficiente impulsó el desarrollo de procedimientos eficientes para encontrar buenas soluciones aunque no fueran óptimas. Estos métodos, en los que la rapidez

del proceso es tan importante como la calidad de la solución obtenida, se denominan heurísticos o aproximados. En (A. Díaz and F.T. 1996) y en otras literaturas existen diversas definiciones diferentes de algoritmo heurístico, entre las que destacamos la siguiente:

Un método heurístico es un procedimiento para resolver un problema de optimización bien definido mediante una aproximación intuitiva, en la que la estructura del problema se utiliza de forma inteligente para obtener una buena solución.

En contraposición a los *métodos exactos* que proporcionan una solución óptima del problema, los *métodos heurísticos* se limitan a proporcionar una buena solución no necesariamente óptima. Lógicamente, el tiempo invertido por un método exacto para encontrar la solución óptima de un problema difícil, si es que existe tal método, es de un orden de magnitud muy superior al del heurístico (pudiendo llegar a ser tan grande en muchos casos, que sea inaplicable).
Existen muchos métodos heurísticos de naturaleza muy diferente, por lo que es complicado dar una clasificación completa. Además, muchos de ellos han sido diseñados para un problema específico sin posibilidad de generalización o aplicación a otros problemas similares. Las heurísticas más conocidas pudieran ubicarse como sigue:

- Método de Descomposición
- Métodos Inductivos
- Método de Reducción
- Métodos Constructivos
- Métodos de Búsqueda Local

En este caso el algoritmo que se utilizará es un método constructivo que consisten en construir literalmente paso a paso una solución del problema. Usualmente son métodos deterministas y suelen estar basados en la mejor elección en cada iteración.

Estos métodos han sido muy utilizados en problemas clásicos como el del viajante de comercio.

Los métodos constructivos y los de búsqueda local constituyen la base de los procedimientos metaheurísticos. En los últimos años han aparecido una serie de métodos bajo el nombre de Metaheurísticos con el propósito de obtener mejores resultados que los alcanzados por los heurísticos tradicionales. El término metaheurístico fue introducido por Fred Glover en 1986, después de este se han introducido diversas definiciones de metaheurística, los profesores Osman y Kelly en (Kelly 1996) introducen la siguiente definición:

Los procedimientos Metaheurísticos son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son efectivos. Los Metaheurísticos proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los mecanismos estadísticos.

Los procedimientos Metaheurísticos se sitúan conceptualmente “por encima” de los heurísticos en el sentido que guían el diseño de estos. Así, al enfrentarnos a un problema de optimización, podemos escoger cualquiera de estos métodos para diseñar un algoritmo específico que lo resuelva aproximadamente. Dentro de esta variedad de métodos podemos destacar la Optimización Basada en Enjambre de Partículas (Particle Swarm Optimization, PSO) y Optimización Basada en Colonias de Hormigas (Ant Colony Optimization, ACO), metaheurísticas que han tomado gran auge de investigación en el campo de la inteligencia artificial. Estas dos Metaheurísticas caen en la categoría de algoritmos bioinspirados o de vida artificial y de inteligencia colectiva, ya que la potencialidad de estos modelos para resolver problemas está dada por la cooperación entre individuos de una forma directa o indirecta. Específicamente en la Metaheurística ACO la cooperación se realiza entre los agentes de una forma indirecta.

Es creciente el interés científico en este nuevo modelo computacional ACO por su alta aplicabilidad para resolver diferentes problemas de optimización discreta. En los últimos años la investigación sobre ACO a estado encaminada a mejorar la calidad de las soluciones encontradas, todas basadas en el estudio de su convergencia y en el estudio de sus parámetros, solo en los resultados alcanzados en (Cáceres, Bello et al. 2006, Bello, Puris et al. 2006) se ha profundizado en como disminuir el tiempo de ejecución para su versión secuencial, manteniendo una alta calidad en las soluciones. La nueva estrategia de cooperación se conoce como "Optimización Basada en Colonias de Hormigas en II Etapas" (TS-ACO). Esta estrategia propone hacer una división en el espacio de búsqueda en dos, en la primera etapa solo una parte de las hormigas van a solucionar un sub-problema de tamaño inferior al problema original, estas soluciones parciales encontradas servirán de estado inicial para que las hormigas de la colonia durante la segunda etapa busquen en el espacio de búsqueda restante, para completar las soluciones al problema en general, de esta forma se hace mucho más cooperativo el trabajo de estos agentes. Esta nueva estrategia ha sido probada para problemas como el Viajante de Comercio (TSP), Secuenciación de Tareas (JSSP), Selección de Rasgos (FSP), Asignación Cuadrática (QAP) y Cubrimiento de Conjuntos (SCP), en todos los casos queda probado que esta estrategia de búsqueda puede ser fácilmente aplicada a cualquier algoritmo del modelo ACO tradicional y conjuntamente con esto es capaz de disminuir el tiempo de ejecución en una razón del 40% al 60%.

El Problema de Panificación de las Fuerzas de Trabajo(Enrique Alba 2007a) ("Workforce Planning Problem", en inglés, WPP) es un problema que contiene grandes niveles de complejidad debido no solo al gran tamaño de sus instancias reales, sino también al proceso de modelación en sí por la cantidad de restricciones que presenta.

Por estas razones se desea aplicar ACO a este problema, pues nunca ha sido modelado utilizando dicha metaheurística. En la revisión bibliográfica internacional desarrollada para este trabajo se ha detectado que solo ha sido resuelto utilizando

Algoritmos Genéticos, es por eso que se plantea dicha tarea para comparar los resultados obtenidos y llegar a conclusiones acerca de la factibilidad del uso de este nuevo modelo en este problema.

Para darle respuesta a esta problemática se ha determinado desarrollar un **trabajo investigativo de tipo exploratorio** cuyas **preguntas de investigación** son:

1. ¿Cómo debe ser modelado el WPP para que sea posible aplicarle los algoritmos de la metaheurística ACO?
2. ¿Cómo debe ser aplicado la nueva estrategia de exploración ACO en II etapas al WPP de forma tal que se encuentren soluciones parecidas a las encontradas por ACO y se logre disminuir el tiempo de ejecución?

La **hipótesis** del presente trabajo es, en esencia, el uso de la metaheurística ACO, la cual ha sido aplicada a otros problemas, como el TSP, JSSP, QAP y al SCP, y ha brindado buenos dividendos, por lo cual deben obtenerse para el problema WPP buenos resultados, no solo en cuanto al tiempo de procesamiento de la información, sino además en cuanto a la calidad de los resultados obtenidos.

Teniendo en cuenta lo antes referido se plantea el **objetivo general** siguiente:

Aplicar la Metaheurística ACO al Problema de Planeamiento de las Fuerzas de Trabajo.

Y los **objetivos específicos** que se enuncian a continuación:

- Modelar el WPP de forma tal que se le pueda aplicar la Metaheurística ACO.
- Implementar computacionalmente dos módulos, uno con el algoritmo Sistema de Hormigas Max-Min (Max-Min Ant System, MMAS) de ACO puro o tradicional y otro con el algoritmo Max-Min Ant System en II Etapas (TS-MMAS), ambos aplicados al WPP.
- Validar los resultados obtenidos a través de técnicas estadísticas y llegar a conclusiones sobre la efectividad.

El presente trabajo de diploma se estructura en tres Capítulos.

En el Capítulo 1 se abordará en detalles el problema a resolver y la metaheurística ACO de forma general.

En el Capítulo 2 se expondrá genéricamente la nueva metaheurística ACO en II Etapas y se explicará cómo fueron atacados ambos modelos en todos sus puntos esenciales para su aplicación al WPP.

Finalmente, en el Capítulo 3 se presentarán los resultados experimentales alcanzados por la implementación de ambas metaheurísticas para este problema.

Tras el análisis de estos tres capítulos aparecerán las Conclusiones y las Recomendaciones.

Capítulo 1. Marco Teórico:

En el presente capítulo será expuesto el problemas a tratar y su modelación matemática. Además se hará un esbozo general de la teoría de Optimización Basada en Colonias de Hormigas (ACO).

1.1 El Problema de Planeamiento de las Fuerzas de Trabajo (WPP)

Los problemas de optimización del tipo NP-completos(Johnson 1979) (las siglas NP vienen de Non Polynomial, o sea, son aquellos problemas para los cuales no se ha encontrado ningún algoritmo que los resuelva de forma exacta con complejidad polinomial y sólo se conocen algoritmos con complejidad exponencial o factorial), que son particularmente complejos, se resuelven con buenos resultados aplicándoles Optimización basada en Colonias de Hormigas (ACO son sus siglas en inglés). El WPP cae dentro de este tipo de problemas y consiste a groso modo en asignar un número de trabajadores a realizar una cantidad determinada de tareas minimizando el costo de la asignación.

En WPP(Enrique Alba 2007a) es dado un conjunto de trabajos $J = \{1, \ldots, m\}$ que deben completarse durante el próximo período de la planificación (por ejemplo, una semana). Cada trabajo j requiere un número de horas dj durante el período de la planificación. Dado el conjunto $I = \{1, \ldots, n\}$ de obreros disponibles. La disponibilidad de cada obrero i durante el período de la planificación es si horas. Por las razones de eficiencia, un obrero debe realizar un número mínimo de horas ($hmin$) de cualquier trabajo al que se asigne y, al mismo tiempo, ningún obrero puede ser asignado a más de un número de trabajos ($jmax$) durante el período de la planificación. Los trabajadores tienen diferentes habilidades, por lo tanto, sea Ai el conjunto de trabajos para los cueles el trabajador i está calificado para realizar. No más de t trabajadores pueden ser asignados durante el período de planeamiento. En

otras palabras a lo sumo t trabajadores pueden ser seleccionados del conjunto I de n trabajadores y el subconjunto seleccionado debe ser capaz de completar todos los trabajos. El objetivo es encontrar una solución factible que optimice la función objetivo.

Se usa cij como el costo de asignarle al trabajador i el trabajo j para formular el problema de optimización asociado dicho problema. Luego matemáticamente el problema sería:

$$x_{ij} = \begin{cases} 1, si\ el\ trabajador\ i\ es\ asignado\ al\ trabajo\ j \\ 0, en\ otro\ caso \end{cases}$$

$$y_i = \begin{cases} 1, si\ el\ trabajador\ i\ es\ seleccionado \\ 0, en\ otro\ caso \end{cases}$$

$z_{ij} = número\ de\ horas\ que\ el\ trabajador\ (i)\ es\ asignado\ al\ trabajo\ (j)$

$A_i = Conjunto\ de\ trabajos\ para\ los\ cuales\ el\ trabajador\ i\ está\ calificado$

$Q_j = Conjunto\ de\ trabajadores\ calificados\ para\ realizar\ el\ trabajo\ j$

$$Minimizar \sum_{i \in I} \sum_{j \in A_i} c_{ij} * x_{ij} \qquad (1)$$

$Sujeto\ a:$

$$\sum_{j \in A_i} z_{ij} \le s_i * y_i \qquad \forall i \in I \qquad (2)$$

$$\sum_{i \in Q_j} z_{ij} \ge d_j \qquad \forall j \in J \qquad (3)$$

$$\sum_{j \in A_i} x_{ij} \le j_{\max} * y_i \qquad \forall i \in I \qquad (4)$$

$$h_{\min} * x_{ij} \le z_{ij} \le s_i * x_{ij} \qquad \forall i \in I,\ j \in A_i \qquad (5)$$

$$\sum_{i \in I} y_i \le t \qquad (6)$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I,\ j \in A_i$$

$$y_i \in \{0,1\} \qquad \forall i \in I$$

$$z_{ij} \ge 0 \qquad \forall i \in I,\ j \in A_i$$

En el modelo, la función objetivo (1) minimiza el costo de la asignación total. La restricción (2) limita el número de horas para cada trabajador seleccionado. Si el obrero no es escogido, entonces esta restricción no permite ninguna asignación de horas al trabajador. La restricción (3) enfatiza en los requisitos del trabajo, como ha sido especificado el número de horas necesitado completar cada trabajo durante el período de la planificación. La restricción (4) pone límite al número de veces que un trabajador puede ser asignado a realizar trabajos. La restricción (5) limita el mínimo de horas que un trabajador puede ser asignado a laborar en un trabajo, o sea, una vez asignado el trabajador debe realizar un número mínimo de horas; también esta restricción no permite que se le asignen horas de trabajo a un trabajador que no ha sido seleccionado para realizar un trabajo. La restricción (6) limita el número de obreros escogido durante el período planeando.

1.2 Optimización Basada en Colonias de Hormigas

1.2.1 Las colonias de hormigas naturales.

Las hormigas son insectos sociales que viven en colonias y que, debido a su colaboración mutua, son capaces de mostrar comportamientos complejos y realizar tareas difíciles si se compara con un insecto individual. Un aspecto interesante del comportamiento de muchas especies de hormigas es su habilidad para encontrar los caminos más cortos entre su hormiguero y las fuentes de alimento. Este hecho es especialmente interesante si se tiene en cuenta que muchas de las especies de hormigas son casi ciegas, lo que evita el uso de pistas visuales (Sergio Alonso 2003).

Mientras se mueven entre el hormiguero y la fuente de alimento, algunas especies de hormigas depositan una sustancia química denominada feromona. Las feromonas son un sistema indirecto de comunicación química entre animales de una misma especie, que transmiten información acerca del estado fisiológico, reproductivo y social, así como la edad, el sexo y el parentesco del animal emisor, las cuales son recibidas en el sistema olfativo del animal receptor, quien interpreta esas señales, jugando un papel importante en la organización y la supervivencia de muchas

especies (Barán and Almirón 2000). Si no se encuentran ningún rastro de feromona, las hormigas se mueven de manera básicamente aleatoria, pero cuando existe feromona depositada, tienen mayor tendencia a seguir el rastro. De hecho, los experimentos realizados por biólogos han demostrado que las hormigas prefieren de manera probabilística los caminos marcados con una concentración superior de feromona (Pasteels, Deneubourg et al. 1987).

En la práctica, la elección entre distintos caminos toma lugar cuando varios caminos se cruzan. Entonces, las hormigas eligen el camino a seguir con una decisión probabilística sesgada por la cantidad de feromona: cuanto más fuerte es el rastro de feromona, mayor es la probabilidad de elegirlo. Esto lleva a un proceso de auto refuerzo que concluye con la formación de rastros señalados por una concentración de feromona elevada. Este comportamiento debe permitir que con el transcurso de cierto tiempo las hormigas encuentren los caminos más cortos entre su hormiguero y la fuente del alimento (Goss, Aron et al. 1989). Las hormigas solo se comunican de manera indirecta, a través de modificaciones del espacio físico que perciben. Esta forma de comunicación se denomina Estimergia Artificial.

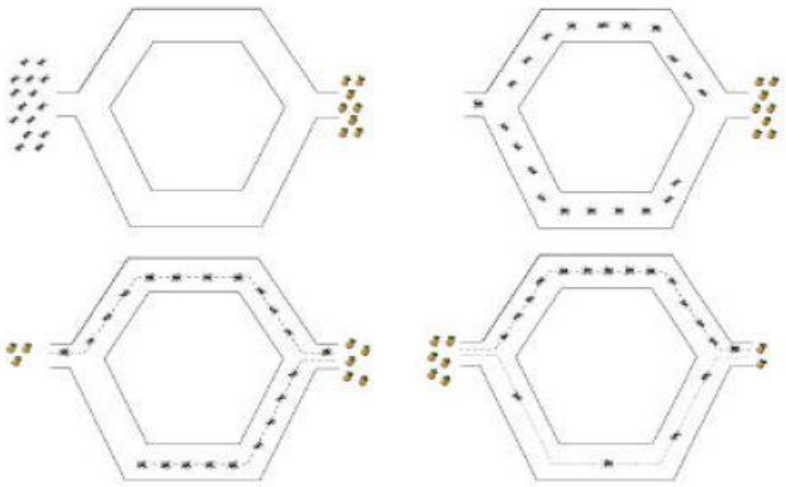


Figura 1. El comportamiento de la colonia termina por obtener el camino más corto entre dos puntos (Bonabeau, Dorigo et al. 1999).

En la Figura 1 se ilustra cómo este mecanismo permite a las hormigas encontrar el camino más corto. Inicialmente no existe ningún rastro de feromona en el medio y cuando una hormiga llega a una intersección, elige de manera aleatoria una de las bifurcaciones posibles. Según transcurre el tiempo y las hormigas están recorriendo los caminos más prometedores, estos van recibiendo una cantidad superior de feromona. Esto ocurre gracias a que, al ser los caminos más cortos, las hormigas que los siguen consiguen encontrar la comida más rápidamente, por lo que comienzan su viaje de retorno antes. Entonces, en el camino más corto habrá un rastro de feromona ligeramente superior y, por lo tanto, las decisiones de las siguientes hormigas estarán dirigidas en mayor medida a dicho camino. Además, este camino recibirá una proporción mayor de feromona por las hormigas que vuelven por él que por las que vuelven por el camino más largo. Este proceso finaliza haciendo que la probabilidad de que una hormiga escoja el camino más corto aumente progresivamente y que al final el recorrido de la colonia converja al más corto de todos los caminos posibles.

Esta convergencia se complementa con la acción del entorno natural que provoca que la feromona se evapore una vez transcurrido un cierto tiempo. Así, los caminos menos prometedores pierden progresivamente feromona porque son visitados cada vez por menos hormigas. Sin embargo, algunos estudios biológicos han demostrado que los rastros de feromona son muy persistentes -la feromona puede permanecer desde unas pocas horas hasta varios meses dependiendo de varios aspectos distintos, como la especie de las hormigas, el tipo de suelo, etcétera. (Bonabeau, Dorigo et al. 1999), lo que provoca una menor influencia del efecto de la evaporación en el proceso de búsqueda del camino más corto.

Numerosos experimentos muestran que, debido a la gran persistencia de feromona, es difícil que las hormigas “olviden” un camino que tiene un alto nivel de feromona aunque hayan encontrado un camino aún más corto (Bonabeau, Dorigo et al. 1999). Hay que tener en cuenta que si se traslada este comportamiento directamente al ordenador para diseñar un algoritmo de búsqueda podemos encontrarnos con que se

quede rápidamente estancado en un óptimo local.

1.2.2 De las hormigas naturales a la metaheurística de Optimización Basada en Colonias de Hormigas.

Los algoritmos de ACO se inspiran directamente en el comportamiento de las colonias reales de hormigas para solucionar problemas de optimización combinatoria (Dorigo and Stützle 2004).

Se basan en una colonia de hormigas artificiales, esto es, unos agentes computacionales simples que trabajan de manera cooperativa y se comunican mediante rastros de feromona artificiales.

Los algoritmos de ACO son esencialmente algoritmos constructivos: en cada iteración del algoritmo, cada hormiga construye una solución al problema recorriendo un grafo de construcción. Cada arista o nodo del grafo, en dependencia del problema que se trate, que representa los posibles pasos que la hormiga puede dar, tiene asociada dos tipos de información que guían el movimiento de la hormiga (Sergio Alonso 2003):

- Información heurística, que mide la preferencia heurística de moverse desde el nodo *r* hasta el nodo *s*, o sea, de recorrer la arista a_{rs}. Se nota por η_{rs}. Las hormigas no modifican esta información durante la ejecución del algoritmo.
- Información de los rastros de feromona artificiales, que mide la "deseabilidad aprendida" del movimiento de *r* a *s*. Imita a la feromona real que depositan las hormigas naturales. Esta información se modifica durante la ejecución del algoritmo dependiendo de las soluciones encontradas por las hormigas. Se denota por τ_{rs}.

En esta sección se presentan los pasos que llevan desde las hormigas reales a la ACO. A partir de ahora debe tenerse en cuenta que los algoritmos ACO presentan una doble perspectiva:

- son una abstracción de algunos patrones de comportamiento naturales

relacionados con el comportamiento que permite encontrar el camino más corto.

- incluyen algunas características que no tienen una contrapartida natural, pero que permiten que se desarrollen algoritmos para obtener buenas soluciones al problema que se pretende resolver (por ejemplo, el uso de información heurística que guíe el movimiento de las hormigas).

El tipo de problemas que se pueden resolver por medio de hormigas artificiales pertenece al grupo restringido de problemas de camino mínimo que se pueden caracterizar por los siguientes aspectos (Dorigo and Caro 1999, Dorigo and Stützle 2003):

- Existe un conjunto de restricciones Ω definido por el problema a solucionar.
- Existe un conjunto finito de componentes $N = \{n_1, n_2, ..., n_k\}$.
- El problema presenta diversos estados que se definen según secuencias ordenadas de componentes $\delta = \langle n_r, n_s, ..., n_u, .. \rangle$ ($\langle r, s, ..., u, .. \rangle$ para simplificar) sobre los elementos de *N*. Si *Δ* es el conjunto de todas las secuencias posibles, llamaremos *Δ'* al conjunto de todas las subsecuencias que respetan las restricciones Ω. Los elementos en *Δ'* definen los estados posibles. Sea $|\delta|$ la longitud de una secuencia δ, esto es, el número de componentes en la secuencia.
- Existe una estructura de vecindario definida como sigue: δ_2 es un vecino de δ_1 si:
    - tanto δ_1 como δ_2 pertenecen a *Δ*,
    - el estado δ_2 puede alcanzarse desde δ_1 en un paso lógico, es decir, si *r* es la última componente de la secuencia δ_1, debe existir una componente *s* tal que $\delta_2 = \langle \delta_1, s \rangle$, o sea, debe existir una transición

válida entre *r* y *s*. El vecindario alcanzable de δ_1 es el conjunto que contiene todas las secuencias $\delta_2 \in \Delta'$. Si $\delta_2 \notin \Delta'$, diremos que δ_2 está en el vecindario no alcanzable de δ_1.

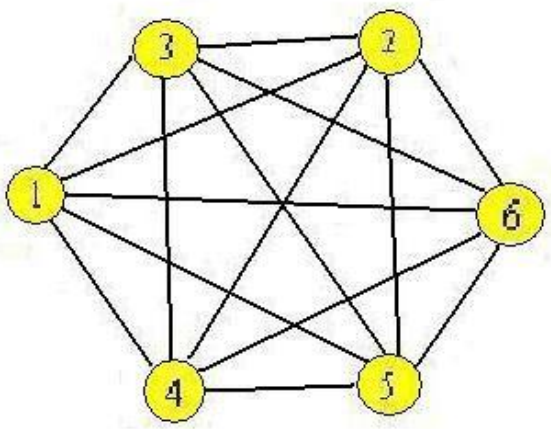
- Una solución *S* es un elemento de *Δ'* que verifica todos los requisitos del problema.
- Existe un costo *C(S)* asociado a cada solución *S*.
- En algunos casos se puede asociar un costo a los estados o una estimación del mismo puede estar asociado a estos.

Como hemos comentado todas las características anteriores se dan en problemas de optimización combinatoria que pueden representarse en forma de grafo ponderado $G = (N, A)$, donde *A* es el conjunto de aristas que conectan el conjunto de componentes *N*. El grafo *G* se denomina grafo de construcción *G*. Por tanto, tenemos que:

- las componentes *n* y *r* son los nodos del grafo,
- los estados δ (y por tanto las soluciones *S*) se corresponden con caminos en el grafo, esto es, secuencias de nodos o aristas,
- las aristas del grafo (a_{rs}) son conexiones/transiciones que definen la estructura del vecindario. $\delta_2 = \langle \delta_1, s \rangle$ será vecino de δ_1 si el nodo *r* es la última componente de δ_1 y la arista a_{rs} existe en el grafo,
- deben existir los costos explícitos c_{rs} asociados con cada arista, y
- las componentes o las conexiones deben tener asociados rastros de feromona τ, que representan un tipo de memoria indirecta y a largo plazo del proceso de búsqueda, así como valores heurísticos η, que representan la información heurística disponible en el problema a resolver.

En la siguiente figura se muestra un ejemplo de un grafo que pudiera corresponder a

un problema de optimización:



$N = \{1,2,3,4,5,6\}$

El conjunto de componentes

$\delta_x = \langle 1,3,4,2,6,5 \rangle$

$\delta_k = \langle 1,3,4,1,6,2,5 \rangle$

Dos secuencias ordenadas de componentes

$n_j * x_j = 1 \qquad \forall j = 1..6$

Una restricción que pertenece a Ω que fuerza que cada nodo solo aparezca una vez en la solución

En base a esto tenemos:

Δ **es el conjunto de todas las secuencias posibles**

$\delta_x, \delta_k \in \Delta$

Δ' **es el conjunto de todas las (sub)secuencias que respetan las restricciones Ω**

$\delta_x \in \Delta' \quad y \quad \delta_k \notin \Delta'$

Figura 2. Grafo correspondiente a un problema de optimización.

1.2.3 Similitudes y diferencias entre las hormigas naturales y artificiales.

Las colonias de hormigas naturales y artificiales comparten una serie de características. Las más importantes pueden resumirse a continuación(Dorigo, Caro et al. 1999):

- Uso de una colonia de individuos que interaccionan y colaboran para solucionar una tarea dada.
- Tanto las hormigas naturales como las artificiales modifican su “entorno” a través de una comunicación estimérgica basada en la feromona. En el caso de las hormigas artificiales, los rastros de feromona artificiales son valores numéricos que están disponibles únicamente de manera local.

- Ambas, las hormigas naturales y las artificiales, comparten una tarea común: la búsqueda del camino más corto (construcción iterativa del una solución de costo mínimo) desde un origen, el hormiguero (decisión inicial), hasta un estado final, la comida (última decisión).
- Las hormigas artificiales construyen las soluciones iterativamente aplicando una estrategia de transición local estocástica para moverse entre estados adyacentes, tal como hacen las hormigas naturales.

Sin embargo, estas características por sí solas no permiten desarrollar algoritmos eficientes para problemas combinatorios difíciles. De hecho, las hormigas artificiales viven en un mundo discreto y tienen algunas capacidades adicionales:

- Las hormigas artificiales pueden hacer uso de la información heurística, no solo de los rastros locales de feromona, en la política estocástica de transición que apliquen.
- Tienen una memoria que almacena el camino seguido por la hormiga.
- La cantidad de feromona depositada por la hormiga artificial es función de la calidad de la solución encontrada. Otra gran diferencia se refiere al momento de depositar la feromona. Las hormigas artificiales normalmente sólo depositan feromona después de generar una solución completa.
- La evaporación de feromona en los algoritmos de ACO es diferente a como se presenta en la naturaleza ya que la inclusión del mecanismo de evaporación es una cuestión fundamental para evitar que el algoritmo se quede estancado en óptimos locales. La evaporación de feromona permite a la colonia de hormigas artificiales olvidar lentamente su historia pasada para redireccionar su búsqueda hacia nuevas regiones del espacio. Esto evita una convergencia prematura del algoritmo hacia óptimos locales.
- Para mejorar la eficiencia y eficacia del sistema, los algoritmos de ACO pueden enriquecerse con habilidades adicionales. Ejemplos típicos son la

capacidad de mirar más allá de la siguiente transición o *"lookhead"* (Michel and Middendorf 1998), la *optimización local* (Stützle and Hoos 1997, Dorigo and Gambardella 1997) y *"backtracking",* cuyo uso no está muy extendido, que persiguen mejorar la eficacia, o la llamada *lista de candidatos* que contiene un conjunto de los estados vecinos más prometedores para mejorar la eficiencia del algoritmo(Dorigo and Gambardella 1997, Dorigo and Caro 1999).

1.2.4 Modo de funcionamiento y estructura genérica de la Metaheurística ACO.

El modo de operación básico de un algoritmo de ACO es como sigue: las *m* hormigas (artificiales) de la colonia se mueven, concurrentemente y de manera asíncrona, a través de los estados adyacentes del problema (que puede representarse en forma de grafo). Este movimiento se realiza siguiendo una regla de transición que está basada en la información local disponible en las componentes (nodos). Esta información local incluye la información heurística (dependiente del problema) y memorística (rastros de feromona) para guiar la búsqueda. Al moverse por el representativo del problema, las hormigas construyen incrementalmente soluciones. Opcionalmente, las hormigas pueden depositar feromona cada vez que crucen un arco (conexión) mientras que construyen la solución (*actualización en línea paso a paso de los rastros de feromona)*.

Una vez que cada hormiga ha generado una solución se evalúa ésta y puede depositar una cantidad de feromona que es función de la calidad de su solución (*actualización en línea de los rastros de feromona*). Esta información guiará la búsqueda de las otras hormigas de la colonia en el futuro.

Es bueno aclarar que la estructura que almacena la feromona esta en dependencia del tipo del problema a resolver: para problemas de secuenciación o asignación, la feromona es asociada a los arcos de grafo representativo, por lo que se hablas de una matriz de feromona, como se presenta en la mayoría de los problemas. Para problemas donde el orden de los elementos de la solución no es importante se le

asocia la feromona a los nodos del grafo, por lo que se utiliza como estructura de almacenamiento un vector de feromona. En lo adelante, para explicar el funcionamiento genérico de la Metaheurística ACO y el comportamiento de sus algoritmos se utilizara el termino de matriz de feromona, sin que se refiera a un tipo de problema en especifico.

Los valores iniciales de feromona pueden ser de la siguiente forma:

- Inicialización aleatoria: donde cada elemento de la matriz es un elemento escogido aleatoriamente con distribución uniforme en el intervalo (0,1).
- Con un valor constante: cada elemento de la matriz es inicializado con el mismo valor, este valor debe de ser en el intervalo (0,1).
- Inicialización dependiendo de una solución (S) calculada por algún otro algoritmo con menor costo computacional; donde cada elemento de la matriz es inicializado con el valor 1/S.

Además, el modo de operación genérico de un algoritmo de ACO incluye dos procedimientos adicionales, la evaporación de los rastros de feromona y las acciones del demonio. La evaporación de feromona la lleva a cabo el entorno y se usa como un mecanismo que evita el estancamiento en la búsqueda y permite que las hormigas busquen y exploren nuevas regiones del espacio. Las acciones del demonio son acciones opcionales -que no tienen un contrapunto natural- para implementar tareas desde una perspectiva global que no pueden llevar a cabo las hormigas por la perspectiva local que ofrecen. Las capacidades adicionales que desarrolla las acciones del demonio están por ejemplos: observar la calidad de todas las soluciones generadas y depositar una nueva cantidad de feromona adicional sólo en las transiciones/componentes asociadas a algunas soluciones, o aplicar un procedimiento de búsqueda local a las soluciones generadas por las hormigas antes de actualizar los rastros de feromona. En ambos casos, el demonio reemplaza la actualización en línea a posteriori de feromona y el proceso pasa a llamarse *actualización fuera de línea de rastros de feromona*.

La estructura de un algoritmo de ACO genérico es como sigue (Caro 1999, M. Dorigo 1999):

Procedimiento MetaheurísticaACO;

Actividades Programadas

Construir Soluciones de las Hormigas

Actualizar Feromonas

Evaporación de la feromona

Acciones del Demonio (opcional)

Fin de las Actividades Fijas

Fin del procedimiento

Este procedimiento se anida en el siguiente procedimiento iterativo:

***P1**: Inicializar los valores de feromona*

nciclo=1

***P2**: Repetir*

Procedimiento MetaheurísticaACO

nciclo= nciclo+1

Hasta que: criterio de parada

Los *criterios de parada* de este proceso iterativo son, entre otros:

- Se alcanza un número máximo de ciclos.
- Se alcanza una solución con la calidad deseada.
- Convergencia a la misma solución (estancamiento o stagnation).
- Se alcanza un tiempo límite o predeterminado de procesamiento.

En cualquier caso, la Metaheurística ACO no es lo suficientemente general como para cubrir la familia completa de algoritmos de hormigas, que pueden definirse

(aunque no de manera muy precisa) como métodos aproximados para solucionar problemas combinatorios basados en características del comportamiento genérico de las hormigas naturales. Algunos ejemplos de algoritmos de hormigas que no están incluidos en la Metaheurística ACO son el Sistema de Hormiga Rápido ("Fast Ant System") (Gambardella 1997b) y el Sistema de Hormiga Híbrido ("Hybrid Ant System") (L. M. Gambardella 1999). El primero es un algoritmo de construcción que se basa en el modo de operación de una única hormiga y que no utiliza de manera explícita la evaporación de feromona. El segundo consiste en un procedimiento de búsqueda local que hace uso de la información de los rastros de feromona para generar soluciones vecinas.

1.2.5 Pasos a seguir para resolver un problema mediante ACO.

Observando las aplicaciones actuales de la ACO, podemos identificar algunas directivas sobre como atacar problemas utilizando esta metaheurística. Estas directivas se pueden resumir en las seis tareas de diseño que se enumeran a continuación:

1. Representar el problema como un conjunto de componentes y transiciones o a través de un grafo de construcción que será recorrido por las hormigas para construir soluciones.

2. Definir de manera apropiada el significado de los rastros de feromona τ_{rs}, esto es, el tipo de decisión que inducen. Este es un paso crucial en la implementación de un algoritmo de ACO y, a menudo, una buena definición de los rastros de feromona no es una tarea trivial. De hecho, normalmente implica tener un buen conocimiento del problema que se quiere solucionar.

3. Definir de manera apropiada la preferencia heurística de cada decisión que debe tomar una hormiga mientras que construye una solución, es decir, definir la información heurística η_{rs} asociada a cada componente o transición. Hay que remarcar que la información heurística es crucial para un buen

rendimiento si no existen o no pueden ser aplicados algoritmos de búsqueda local.

4. Si es posible, implementar una búsqueda local eficiente para el problema que se desea solucionar, porque los resultados de muchas aplicaciones de la ACO a problemas de optimización combinatoria NP-completos demuestran que el mejor rendimiento se alcanza cuando se complementa con optimizaciones locales (Dorigo and Caro 1999, Dorigo and Stützle 2003).
5. Escoger un algoritmo de ACO específico y aplicarlo al problema que hay que solucionar teniendo en cuenta las características propias de cada uno.
6. Refinar los parámetros del algoritmo de ACO. Un buen punto de partida para la especificación de los valores de los parámetros es usar configuraciones de parámetros que han demostrado ser buenas cuando se aplicaban en el algoritmo de ACO a problemas similares o a una gran variedad de problemas distintos. Otra posible alternativa para evitar el esfuerzo y tiempo necesarios para esta tarea es utilizar procedimientos automáticos de refinamiento de parámetros (Birattari, Stützle et al. 2002).

1.3 Algoritmos de la Metaheurística ACO

En la literatura se han propuesto diversos algoritmos que siguen la Metaheurística ACO. Entre los algoritmos de ACO disponibles para problemas de optimización combinatoria NP-duros, se encuentran el Sistema de Hormigas (Ant System (AS))(Dorigo, Maniezzo et al. 1996) ,el Sistema de Colonia de Hormigas (Ant Colony System (ACS)) (Gambardella 1997c), el Sistema de Hormigas Max-Min (Max-Min Ant System) (Hoos 2000), el AS con Ordenación (Rank-Based Ant System) (B. Bullnheimer 1999), el Sistema de la Mejor-Peor Hormiga (Best-Worst Ant System)(O. Cordón 2000), entre otros.

Muchos investigadores interesados por la originalidad y el rendimiento del modelo ACO, aplicaron la técnica con excelentes resultados a problemas tan diversos como:

- El paradigma del cajero viajante *(Travelling Salesman Problem*) (Dorigo M. 1996, Gambardella 1997a);
- El problema del ordenamiento secuencial *(Sequential Ordering Problem)*(M 1997);
- El problema de ruteo de vehículos *(Vehicle Routing Problem)*(Bullnheimer B. 1999);
- El problema de asignación cuadrática *(Quadratic Assignment Problem)* (A 1999);
- Redes de telecomunicaciones *(Telecommunications Networks)* (Dorigo 1998).

Esta técnica comienza a tener la madurez tecnológica adecuada para su utilización en problemas reales, como puede verse por la publicación del libro (E. Bonabeau 1999).

El ***Sistema de Hormigas*** (*Ant System*, *AS*) (M. Dorigo 1996), desarrollado por Dorigo, Maniezzo y Colorni en 1991, fue el primer algoritmo de ACO. Inicialmente, se presentaron 3 variantes distintas: Hormiga-Densidad (Ant-Density), Hormiga-Cantidad (Ant-Quality) y Hormiga-Ciclo (Ant-Cycle), que se diferenciaban en la manera en que se actualizaban los rastros de feromona. En los dos primeros, las hormigas depositaban feromona mientras que construían sus soluciones (esto es, aplicaban una actualización en-línea paso a paso de feromona), con la diferencia de que la cantidad de feromona depositada en él hormiga-densidad es constante, mientras que la depositada en hormiga-cantidad dependía directamente de la deseabilidad heurística de la transición hrs. Por último, en hormiga-ciclo, la deposición de feromona se lleva a cabo una vez que la solución está completa (actualización en línea a posteriori de feromona). Esta última variante era la que obtenía unos mejores resultados y es por tanto la que se conoce como AS en la literatura y en el resto de este trabajo.

El AS se caracteriza por el hecho de que la actualización de feromona se realiza una vez que todas las hormigas han completado sus soluciones, y se lleva a cabo como

sigue: primero, todos los rastros de feromona se reducen en un factor constante, implementándose de esta manera la evaporación de feromona según la expresión:

$$t_{ij} \leftarrow (1 - p) * t_{ij},\ p \in (0,1]$$

donde *p* se conoce como constante de evaporación y es la encargada de reducir los rastros de feromona para evitar el estancamiento de las soluciones y t_{ij} la cantidad de feromona asociada al arco *(i, j)*.

A continuación cada hormiga de la colonia deposita una cantidad de feromona que es función de la calidad de su solución, según la siguiente expresión:

$$t_{ij} \leftarrow t_{ij} + \Delta t^k \quad \forall a_{ij} \in S^k$$

donde $\Delta t^k = f(C(S^k))$, es decir que la cantidad de feromona a depositar en cada arco (a_{ij}) que pertenece a la solución (S^k) de la hormiga *k* depende totalmente de la calidad ($C(S^k)$) de la solución encontrada por dicha hormiga.

Inicialmente, el AS no usaba ninguna acción del demonio, pero es relativamente fácil, por ejemplo, añadir un procedimiento de búsqueda local para refinar las soluciones generadas por las hormigas, esto se verá más adelante. Las soluciones en el AS se construyen como sigue. En cada paso de construcción, una hormiga k escoge ir al siguiente nodo con una probabilidad que se calcula como:

$$p_{ij}^k = \frac{(t_{ij})^\alpha * (\eta_{ij})^\beta}{\sum_{j \in N_i^k} (t_{ij})^\alpha * (\eta_{ij})^\beta} \qquad si \quad j \in N_i^k$$

donde N_i^k es el vecindario alcanzable por la hormiga k cuando se encuentra en el nodo i, los parámetros alfa (α) y beta (β) controlan el proceso de búsqueda. Para α=0 se tiene una búsqueda heurística estocástica clásica, mientras que para $\beta = 0$ solo el valor de la feromona tiene efecto. Un valor de $\alpha < 1$ lleva a una rápida situación de convergencia (stagnation), P_{ij}^k es un vector de probabilidades de visitar

cada uno de los nodos de la vecindad (N_i^k) por la hormiga *k*, τ_{ij} el valor del elemento i, j en la matriz de feromona y η_{ij} , se denomina función de visibilidad o función heurística que depende totalmente de las características del problema que se va a resolver.

El ***Sistema de Hormigas Max-Min*** (Max-Min Ant System)(Stützle and Hoos 2000), desarrollado por Stützle y Hoos en 1996, es una extensión del AS tradicional y muestra resultados satisfactorios en su aplicación. Max-Min extiende al AS tradicional en los siguientes aspectos:

Se aplica una actualización de los rastros de feromona fuera de línea. Después que todas las hormigas hayan construido su solución cada rastro sufre una evaporación:

$$t_{rs} = (1 - \text{constEvap}) * t_{rs} \tag{7}$$

Y a continuación la feromona se deposita siguiendo la siguiente fórmula:

$$t_{rs} = t_{rs} + f(C(S_{mejor})) \qquad \forall a_{rs} \in S_{mejor} \tag{8}$$

La *mejor hormiga* a la que se le permite añadir feromona puede ser la que tiene una solución *mejor de la iteración* o la solución *mejor global*. Los resultados experimentales demuestran que el mejor rendimiento se obtiene incrementando gradualmente la frecuencia de escoger la mejor global para la actualización de feromona (Stützle and Hoos 2000).

Además, en el Max-Min las soluciones que ofrecen las hormigas suelen ser mejoradas usando optimizadores locales antes de la actualización de feromona.

- Los valores posibles para los rastros de feromona están limitados al rango *[tmin,tmax]*. Por lo tanto, la probabilidad de un estancamiento del algoritmo disminuye al darle a cada conexión existente una probabilidad, aunque bastante pequeña, de ser escogida. En la práctica, existen heurísticas para

fijar los valores de *tmin* y *tmax*. Se puede ver que, a causa de la evaporación de la feromona, el nivel máximo de feromona en los rastros está limitado a $t^*max = 1/(r \times C(S^*))$, donde S^* es la solución óptima. Basándonos en este resultado, la mejor solución global puede usarse para estimar *tmax* sustituyendo S^* por *Smejor-global* en la ecuación de t^*max. Para *tmin*, normalmente sólo es necesario escoger su valor de tal manera que sea un factor constante menor que *tmax*.

Para poder incrementar la exploración de nuevas soluciones, el SHMM utiliza en ocasiones re-inicializaciones de los rastros de feromona(Stützle and Hoos 2000) (Stützle and Hoos 1997).

- En vez de inicializar los rastros de feromona a una cantidad pequeña, el Max-Min los inicializa a una estimación del máximo permitido para un rastro (la estimación puede obtenerse generando una solución $S´$ con una heurística voraz y reemplazando dicha solución $S´$ en la ecuación de t^*max). Esto lleva a una componente adicional de diversificación en el algoritmo, ya que al comienzo las diferencias relativas entre los rastros de feromona no serán muy acusadas, lo que no ocurre cuando los rastros de feromona se inicializan a un valor muy pequeño.

1.4 Conclusiones.

A partir de la revisión bibliográfica desarrollada para presentar este capítulo se pueden extraer las conclusiones siguientes:

Los problemas de optimización combinatoria caen regularmente en la categoría de problemas en la que se tienen algoritmos para resolverlos pero cuando su dimensión crece estos se vuelven inaplicables computacionalmente; de modo que la aplicación de los métodos heurísticos representa una buena alternativa.

Los algoritmos bioinspirados y en particular la Metaheurística ACO es un modelo muy poderoso para dar solución a esta clase de problemas ya que propone formas

muy precisas de emplear la cooperación entre agentes para la solución de problemas de optimización combinatoria.

No se encontró ninguna modelación de ACO para resolver el WPP.
De modo que resulta conveniente explorar la forma de modelación de este problema para aplicarle ACO, así como algunas modificaciones al modelo ACO que propongan mejores resultados.

Capítulo 2. ACO aplicado al WPP

En el presente capítulo se describirá el Modelo ACO, así como una nueva variante, el modelo ACO en II Etapas. Además se expondrán las adaptaciones hechas al problema y a los modelos para la aplicación de ambos al WPP.

2.1 Modelo ACO en II Etapas.

En esta sección se introduce una nueva estrategia de trabajo con la metaheurística ACO, la cual aborda el problema de la determinación del estado inicial de la búsqueda.

En los procesos de búsqueda heurística la determinación del estado inicial desde donde comenzar la misma, ha sido una problemática de interés, pues se ha mostrado que tiene un efecto importante en la solución final. El propósito es poder acercar lo más posible el estado inicial al estado final u objetivo. Por supuesto es necesario considerar un balance adecuado entre el costo computacional de lograr ese acercamiento y el costo total, no sea que la suma del costo de aproximar el estado inicial al final más el costo de hallar la solución desde ese estado inicial "mejorado", sea mayor que el costo de buscar la solución desde un estado inicial original.

De modo que el propósito es el siguiente: sea E_i un estado inicial generado aleatoriamente, u obtenido por cualquier otro método sin un costo computacional significativo, E_i^* un estado inicial generado por algún método M que lo acerca al estado final con un costo $C_M(E_i^*)$, y sea $CC_{ABH}(x)$ el costo computacional de encontrar una solución desde el estado x usando el algoritmo ABH (Algoritmo de Búsqueda Heurística). Entonces el objetivo es que:

$C_M(E_i^*)+ CC_{ABH}(E_i^*) < CC_{ABH}(E_i)$.

La nueva forma de trabajo con la metaheurística ACO, llamada ACO en II etapas (Two-Step ACO, TS-ACO), propone hacer una división de la exploración del espacio de búsqueda en dos, de modo que en la primera etapa se encuentren soluciones

preliminares (soluciones parciales) que sirven de estado inicial para el proceso de búsqueda desarrollado en la segunda etapa.

Para realizar esta división, primero se determinaron los parámetros que influyen en el tamaño de la exploración del espacio de búsqueda. Se detectó que el número de hormigas (nh), la cantidad de veces que este iterando el algoritmo, representada por su condicion de parada (nc) y el tamaño de las soluciones al problema (nn) ó lo que es lo mismo la cantidad de nodos que aparencan en la solución.

Una vez determinados los parámetros que definen la dimensión de la exploración es necesario determinar hasta donde llega la primera etapa del proceso, o sea, determinar el punto exacto donde termina la primera etapa y comienza la segunda. Para establecer el tamaño de cada etapa se le introduce al modelo un factor de proporcionalidad (r), el cual indica cuánto abarca cada etapa dentro del proceso de búsqueda completo. Este valor debe estar en el intervalo (0,1), o sea, 0<r<1.

Por ejemplo, si r=0.3, esto significa que la primera etapa cubrirá el 30% del espacio de búsqueda completo. En otras palabras, el 30% de la cantidad total de hormigas recorrerán el 30% de los nodos del grafo, en un 30% de la cantidad de ciclos. Después que se termine el proceso de búsqueda de la primera etapa, sólo una cantidad de las mejores soluciones parciales encontradas en la primera etapa denotadas por (cs) servirán de punto de partida de las hormigas restantes (0.7*nh) de la segunda etapa, para completar las soluciones con las soluciones del subproblema restante (0.7*nn), en la cantidad de ciclos restante (0.7*nc). La cantidad de soluciones parciales que pasan de la primera etapa a la segunda son un subconjunto de la cantidad de soluciones encontradas, donde $cs \leq$ ◖*nh‾.

Nótese que esta estrategia es aplicable a cualquier algoritmo de la metaheurística ACO, además es muy flexible si se desea trabajar mezclando algoritmos de dicha metaheurística, o si se desea trabajar con el mismo algoritmo pero con parámetros diferentes para lograr un diferente comportamiento en cada etapa.

2.2 Implementación del algoritmo Max-Min Ant System al WPP.

2.2.1 Representación del problema como un grafo de construcción.

Para representar el problema como un conjunto de nodos y transiciones que representen un grafo de construcción que pueda ser recorrido por las hormigas se determinó que cada nodo represente una combinación de trabajador- trabajo, o sea, cada nodo del grafo representaría un trabajador i que está realizando un trabajo j para el cual el está capacitado. De esta forma cada hormiga, una vez generado el nodo inicial, puede "saltar" de un trabajador a otro y de un trabajo a otro e ir incorporando estos nodos al vector solución y de esta forma encontrar un camino que constituya las asignaciones a realizar para completar todos los trabajos. Los pesos o valores de los nodos del grafo, en este caso trabajador-trabajo, no son más que los costos asociados a la asignación de ese trabajador a ese trabajo. A su vez, las transiciones entre los nodos vienen determinadas por la vecindad alcanzable desde cada nodo, o sea, el conjunto de trabajos con sus respectivos trabajadores calificados para realizarlos que cumplen las restricciones del problema. Veamos un ejemplo de cómo se comportaría este proceso.

$$A(trabajador, trabajo) = \begin{pmatrix} 15 & 26 & 0 & 34 \\ 22 & 0 & 32 & 18 \end{pmatrix}$$

Los valores de la matriz A significan el costo de asignar un trabajador i a un trabajo j, en caso de que el elemento (i, j) tenga valor cero significa que el trabajador i no está calificado para realizar el trabajo j. Luego cada elemento de la matriz tiene un valor asociado que simboliza la combinación trabajador-trabajo, por ejemplo, el valor 1 significa que el trabajador cero está realizando el trabajo 1, este proceso no es más que dividir el número del nodo por el total de elementos que contiene una fila, la división entera daría el trabajador y el resto el trabajo. Para el ejemplo de la matriz A la lista de nodos sería la siguiente:

0, 1, 2, 3, 4, 5, 6, 7

Y el grafo asociado con todos los posibles caminos seria:

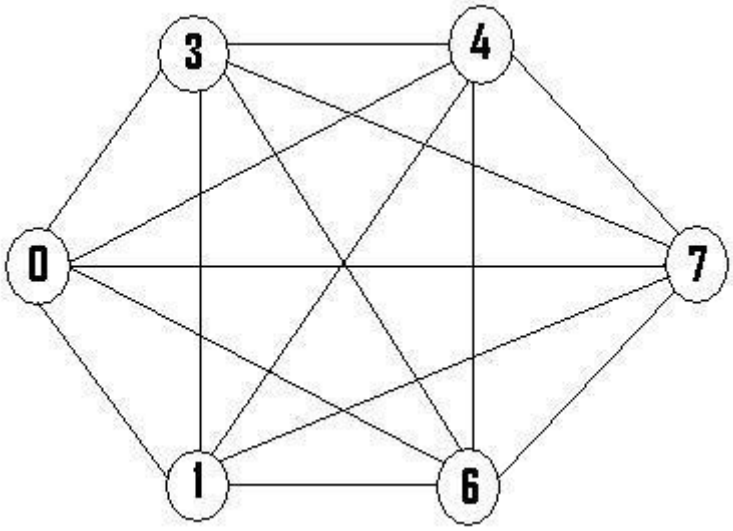


Figura 3. Grafo con todos los caminos posibles correspondiente a la matriz A.

Donde el nodo 2 y el nodo 5 son nodos no factibles pues el trabajador no está calificado para realizar el trabajo que le corresponde, por esta razón es que no aparecen en el grafo. Para el caso del nodo 2 significa que el trabajador 0 no está calificado para realizar el trabajo 2, y en el caso del nodo 5 que el trabajador 1 no lo está para el trabajo 1.

El siguiente ejemplo muestra parte del recorrido que puede realizar una hormiga para la matriz de costos A citada anteriormente, por algún método se determino que para el ejemplo anterior se debía comenzar el nodo 1, algunos de los posibles caminos que pudiesen encontrar las hormigas durante su recorrido serian:

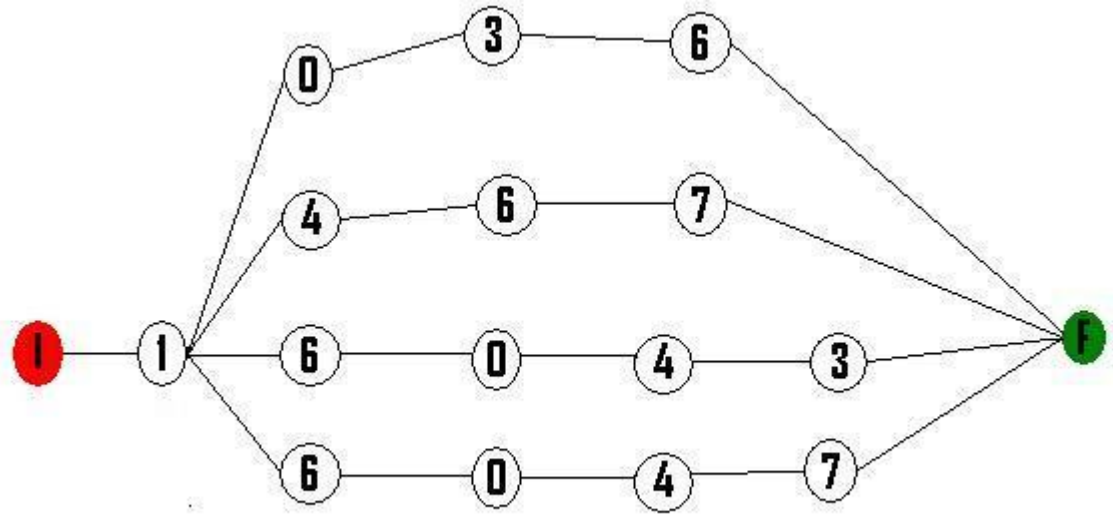


Figura 4. Ejemplo de grafo con posibles soluciones.

En la Figura 4 el grafo obtenido no tiene en los nodos el peso o costo, sino que el número del nodo en sí que representa que trabajador está realizando que trabajo. En el mismo se muestra algunas de las posibles soluciones del problema, no necesariamente tiene que ser así, pues la hormiga se mueve en base a las restricciones del problema, con las cuales se determina que trabajador es asignado a un trabajo. Estas restricciones son:

1. El trabajador está capacitado para realizar el trabajo que se le desea a asignar.
2. El trabajador no ha excedido el límite de veces que puede ser asignado.
3. El trabajador aun tiene horas disponibles para trabajar.
4. La solución contiene un número de trabajadores menor que al valor permisible.

Hay que resaltar que en este problema las hormigas terminan su recorrido cuando terminan de realizar todos los trabajos, y puede un trabajo ser asignado varias veces a diferentes trabajadores hasta que sea completado, pues como se restringe el

número de trabajadores que puede contener una solución, se hace necesario que varios trabajadores laboren en un mismo trabajo para que este sea completado.

2.2.2 Definición de los rastros de feromona.

Para el trabajo con los rastros de feromona en este problema se hizo necesario un análisis de cómo debía ser conformada la estructura de datos que debía guardar la información memorística asociada a la feromona. Considerando que no era importante el orden en que fueran añadidos los nodos a la solución, se decidió construir una matriz de feromonas que guardara en cada elemento la feromona asociado a cada combinación de trabajador- trabajo dado. De esta forma mientras mayor fuera el valor numérico depositado, mayor seria la deseabilidad sobre todo del trabajador correspondiente para integrar futuras soluciones.

Una vez decidida la estructura de datos a utilizar para almacenar los rastros de feromonas se procedió a analizar cómo se actualiza dicha matriz en el Sistema de Hormigas Max-Min.

Como se menciono en el epígrafe 1.3 en este sistema de hormigas se aplica una actualización de los rastros de feromona fuera de línea donde los rastros de feromona sufren una evaporación según la fórmula (7). A continuación es depositada la feromona, para este problema en específico la formula quedaría:

$$MatrixPheromone_{ij} = MatrixPheromone_{ij} + constEvap * 1/\cos te$$

Donde *coste* es el costo asociado a la solución encontrada por la *mejor hormiga.*

Los valores de t_{min} y t_{max}, en este caso en particular se calcularon de la siguiente forma:

$$t_{\max} = 1/(1 - constEvap) * 1/\cos te$$

$$t_{\min} = t_{\max} / 10.0 * numero\_trabajadores$$

El pseudocódigo siguiente muestra como se regulan los valores de feromonas de la matriz de feromonas:

1. Procedimiento **regulate_PheromoneMaxMin()**
2. *mientras* **no se cumple condición_de_parada**
3. Si **matriz_feromona[i][j] es mayor que tmax**
4. **matriz_feromona[i][j] = tmax**
5. Sino
6. Si **matriz_feromona[i][j] es menor que tmin**
7. **matriz_feromona[i][j] = tmin**
8. Fin Sino
9. Fin Mientras
10. Fin

Según bibliografía consultada (Stützle and Hoos 1997, Stützle and Hoos 2000) es conveniente actualizar los rastros de feromona cada cierto número de iteraciones por la mejor solución local o por la mejor solución global, y cuando existe un número grande de iteraciones se recomienda aplanar la feromona. En este problema constantemente se actualiza por la mejor solución local, cada 100 iteraciones se actualiza por la mejor solución global, cada 300 iteraciones se aplana toda la matriz de feromona.

Este proceso de aplanar o reinicializar los valores de feromona de realiza de la siguiente manera:

1. Procedimiento **aplanarMatriz_Feromona()**
2. *mientras* **no se cumple condición_de_parada**
3. **matriz_feromona[i][j] = 0.5 * matriz_feromona[i][j]**
4. **Fin** mientras
5. **Fin** procedimiento

2.2.3 Definición de la heurística.

Una heurística no es más que una función que determina la calidad de un nodo, o sea, que calcula en qué medida es factible incorporar un nodo o arco según sea el caso dependiendo del problema tratado, a la solución que está siendo construida. Esto implica que es muy importante en todos los problemas de búsqueda que se seleccione e implemente una buena heurística de forma tal que la selección de cada componente de la solución sea lo más cercana posible al óptimo en cada paso en aras de obtener buenos resultados generales.

En el WPP es evidente que entre dos nodos que no han sido incorporados a la solución, uno es más "deseable" que otro si tiene un menor costo (mayor que cero) correspondiente, por la repercusión que tiene el mismo en la calidad final de la solución que se encuentre. Luego, la primera heurística probada fue $f(x) = \frac{1}{C(x)}$.

Nótese que el costo es inversamente proporcional a la calidad, o sea, que mientras menor sea el costo de un nodo más deseable es el mismo para ser incorporado. Esta heurística, luego de las pruebas practicadas, no arrojó muy buenos resultados. Utilizándola fue posible diferenciar entre nodos potencialmente buenos y nodos cuya incorporación a la solución no era nada aconsejable por su elevado peso o costo, pero se hizo evidente que la función heurística debía ser mejorada incorporándole más parámetros para que la selección fuera más específica y discriminatoria.

De esta forma se procedió a tener en cuenta otros factores asociados a los trabajos y los trabajadores, como en el WPP es conveniente que un trabajador sea asignado la mayor cantidad de veces posibles para reducir el número de trabajadores que contenga una solución, entonces se tomo en cuenta el tiempo que requería cada trabajo y el tiempo disponible que poseía el trabajador correspondiente inicialmente se probó usando $f(x) = \left| \frac{T_{trabajadori} - T_{trabajoj}}{C_{ij}} \right|$ donde él $T_{trabajadori}$ es el tiempo disponible que posee el trabajador seleccionado y $T_{trabajoj}$ es el tiempo que requiere

dicho trabajo. Vale resaltar que la relación entre el tiempo que requiere el trabajo y el tiempo disponible de cada trabajador hace más rigurosa la selección del próximo nodo pues de esta forma se va chequeando desde ya que el trabajador seleccionado posea horas disponibles y el trabajo seleccionado no se ha finalizado. Usando esta heurística los resultados obtenidos mejoraron ostensiblemente. Sin embargo, aun no podían ser considerados como buenos pues era necesario aun darle más peso a los trabajadores que ya hubiesen sido asignados.

De esta forma fue constatado que se les debían hacer más modificaciones a la función heurística, y se determinó que era conveniente darle peso a los valores más pequeños de la anterior función heurística por lo tanto la nueva función quedaría de la siguiente forma: $$f(x) = 1 - \left| \frac{T_{trabajadori} - T_{trabajoj}}{C_{ij}} \right|$$

De esta manera mientras menor fuese la relación $T_{trabajadori}$ - $T_{trabajoj}$ más peso tendría este nodo por la posibilidad de que ya el trabajador pudiese haber sido asignado a realizar otro trabajo previamente. Finalmente con esta heurística los resultados obtenidos fueron de la calidad necesaria.

2.2.4 Refinación de los parámetros.

Como ya se ha mencionado antes, el algoritmo requiere de varios parámetros a la hora de su ejecución, a saber:

- cantidad de hormigas (ch): la cantidad de agentes que se desea recorra el grafo de construcción del problema. Cada uno construirá su propia solución, eligiéndose en cada ciclo la mejor hormiga y comparando con la mejor solución global encontrada hasta el momento para si es mejor, sustituirla.
- cantidad de ciclos (cc): el número de veces que se repetirá el proceso. En cada ciclo todas las hormigas construyen cada una su propia solución y al final se elije la mejor solución encontrada en ese ciclo para compararla con la

mejor solución global encontrada hasta el momento y, en caso de que sea mejor, sustituirla.

- nivel de feromona inicial: valor numérico que llevarán al comienzo del proceso todos los nodos de la matriz de feromonas. Se busca con esto que al principio todas las hormigas tengan igual probabilidad por la parte referente a la información memorística de viajar a cualquiera de los nodos y este valor influye en el comportamiento del algoritmo, por lo cual es un parámetro a determinar para ver con cual se obtienen los mejores resultados.
- constante de evaporación: corresponde al por ciento del valor numérico correspondiente a un nodo de la matriz de feromonas en que será disminuido el mismo cuando se lleve a cabo el proceso de evaporación.
- alfa (α): potencia el uso de la información memorística para la determinación del próximo nodo a visitar.
- beta (β): potencia el uso de la información heurística para la determinación del próximo nodo a visitar.

Todos estos parámetros pueden alcanzar un rango de valores variable y el algoritmo depende estrechamente para una buena ejecución y la obtención de resultados favorables de que estos valores se ajusten y se combinen de forma que todos contribuyan a que la convergencia sea rápida y buena. Determinar los mejores valores de estos parámetros es una tarea paciente y que requiere de una constante observación del efecto que causa en la calidad de las soluciones obtenidas la más mínima variación de cualquiera de ellos. También pudieran usarse métodos de refinamiento de parámetros que funcionaran de forma automática y que hallaran los mejores valores de estos, pero en este caso específico los parámetros fueron determinados manualmente tras un proceso largo de selección y pruebas. Los mismos se muestran en el Capítulo 3.

En el caso del algoritmo Max-Min AS en II tapas requería además de un parámetro adicional, el factor de proporcionalidad r, del cual también dependía la calidad de las soluciones alcanzadas por ese algoritmo.

2.2.5 Métodos principales del MMAS.

Teniendo en cuenta todos los aspectos vistos anteriormente se implementó el algoritmo MMAS aplicado al WPP. En esta sección se entrará brevemente en detalles acerca de la estructura de los dos principales métodos del algoritmo para dar una idea final de cómo resultaron. Los métodos en consecuencia son wayAS y executeWPP.

A continuación se muestra un pseudocódigo del método executeWPP, el cual se encarga de todo el proceso general de conducción del hilo del algoritmo.

1. Procedimiento **executeWPP()**
2. **Mejor_solución_global = +∞**
3. *mientras* **no se cumple condición_de_parada**
4. *para* **cada una de las hormigas hacer**
5. **Crear_nueva_hormiga(nodo_inicial)**
6. **fin** *para*
7. *mientras* **queden hormigas que no han terminado**
8. *para* **cada hormiga que no ha completado su solución**
9. **Calcular_nuevo_nodo()**
10. **fin** *para*
11. **fin** *mientras*
12. **Calcular_mejor_solución_del_ciclo()**
13. *si* **Mejor_solución_del_ciclo < Mejor_solución_global**
14. **Mejor_solución_global = Mejor_solución_del_ciclo**
15. **fin** *si*
16. **Actuilizar feromona**
17. **fin** *mientras*

18. **fin** Procedimiento

Como se puede apreciar, el algoritmo se mantiene iterando hasta que se cumple una condición de parada. Acerca de la misma se abordará con más detalle en el Capítulo 3.

Un aspecto a señalar aquí lo constituye lo referente a la determinación del nodo inicial, o sea, el primer elemento de la solución. Este proceso es muy importante, porque una solución que comienza con un nodo de mala calidad es poco probable que converja luego a una buena solución. En este algoritmo ese procedimiento se lleva a cabo como parte del método Crear_nueva_hormiga del paso 3 del pseudocódigo anterior. En este método se crea el arreglo que contiene todos los datos referentes a las hormigas participantes, y además se determina para cada una de ellas el primer nodo de su solución. Este nodo es seleccionado de una lista de nodos factibles que se crea con todos los nodos donde los elementos de la matriz de costo son diferentes de cero.

El otro método importante del algoritmo es el wayAS, que es el mismo que fue llamado por el executeWPP con el nombre de Calcular_nuevo_nodo. El mismo consiste en hacer uso tanto de la información memorística como de la heurística para que una hormiga, en un estado cualquiera de la construcción de su solución particular, determine de todos los nodos accesibles de su vecindad que se ajusten a las restricciones del problema, uno de ellos hacia el cual avanzar e incorporarlo como un componente más de la solución que está construyendo. A continuación se muestra el pseudocódigo correspondiente a este método:

1. Procedimiento **wayAS ó Calcular_nuevo_nodo (** α, β **)**
2. **Crear arreglo TempArray con un número de localizaciones equivalente a la cantidad de elementos que posee la matriz de costo**
3. **SumArray = 0**
4. *para* **cada localización del arreglo**

5. **b = 0**
6. *si* **nodo** *j* **cumple las restricciones del problema**
7. $$b = (rastro\_feromona)^{\alpha} * (heuristica)^{\beta}$$

   SumArray += b
8. **fin** *si*
9. **TempArray[** *j* **] = b**
10. **fin** *para*
11. para **cada localización del arreglo**
12. **TempArray[** *j* **] /= SumArray**
13. **Fin** para
14. **Seleccionar_nodo()**
15. **Actulizar_restricciones()**
16. **Agregar_nodo_a_solución(columna)**
17. **fin** Procedimiento

En este método se crea un arreglo denominado TempArray que va a guardar la "deseabilidad" de cada nodo, para esto se determina para cada uno los que cumplen las restricciones del problema, su valor b correspondiente (paso 6 del pseudocódigo), que es cero para aquellos nodos que no cumplen al menos una de las restricciones. Luego desde el paso 11 hasta el 13 se lleva a cabo el proceso de normalización que consiste en dividir cada elemento del arreglo entre la suma total de todos ellos que, como se puede apreciar en el algoritmo, se va almacenando en la variable SumArray y, producto de esto, se obtiene el vector normalizado. La principal característica de los elementos del vector en ese momento es que suman exactamente 1.

Con el vector en estas condiciones es factible aplicar el método conocido por la ruleta, o sea Seleccionar_nodo en el pseudocódigo. La idea es dar a cada nodo una probabilidad de ser seleccionado acorde a su "deseabilidad" calculada. Se genera un número aleatorio entre 0 y 1, y se escoge la columna que tenga asociada una probabilidad en cuyo rango se encuentre el número generado.

A continuación el paso 15 se actualizan las restricciones, cabe resaltar que esta actualización sea realiza de acuerdo con el nodo seleccionado y no es más que modificar el tiempo disponible del trabajador seleccionado y del trabajo que se le asignó. Por último se adiciona el nodo generado a la solución que se va generando.

En la Figura 8 correspondiente a la sección Anexos se puede consultar el diagrama de clases de este algoritmo.

2.3 Implementación del algoritmo Max-Min AntSystem en II Etapas para el WPP.

En esta sección se hará referencia solamente a los aspectos que diferenciaron la implementación del Max-Min AS en II Etapas de la del Max-Min AS tradicional. En todos los puntos técnicos y teóricos asociados al problema WPP y al algoritmo AS, la concepción del modelo ACO en II etapas no difiere de la del ACO tradicional, y solamente el método executeWPP tratado en la sección anterior tiene variaciones en cuanto al AS original. La diferencia se debe al hecho de que este método conduce el hilo general del algoritmo y, para aplicar las nuevas concepciones del modelo ACO en II Etapas, solamente era necesario modificar este procedimiento. En el AS en II Etapas este método se encarga de hacer una división del espacio de búsqueda en dos partes, en la primera etapa se hallan soluciones parciales que se completan en la segunda. A continuación mostramos el pseudocódigo de este método:

1. Procedimiento **executeIIstepWPP()**
2. **Mejor_solución_global = +∞**
3. **ch1 = ch * r** **//la cantidad de hormigas de la primera etapa**
4. **ch2 = ch – ch1** **//la cantidad de hormigas de la segunda etapa**
5. **longSol = Determinar_longitud_subsoluciones()**
6. *mientras* **no se cumple condición_de_parada**

   //inicio de la primera etapa
7. *para* **ch1 hormigas hacer**
8. **Crear_nueva_hormiga(nodo_inicial)**

9. **fin** *para*
10. *mientras* **la longitud de las sub-soluciones de las hormigas < longSol**
11. *para* **todas las ch1 hormigas**
12. **Calcular_nuevo_nodo()**
13. **fin** *para*
14. **fin** *mientras*
15. *para* **ch1 hormigas hacer**
16. **Calcular_mejor_solucion_local()**
17. **Guardar_mejores_subsoluciones()**
18. **fin** *para*
19. **Actulizar_feromona()**

//fin de la primera etapa

//inicio de la segunda etapa

20. *para* **ch2 hormigas hacer**
21. **Crear_nueva_hormiga(subsolución)**
22. **fin** *para*
23. *mientras* **queden hormigas que no han terminado**
24. *para* **cada hormiga que no ha completado su solución**
25. **Calcular_nuevo_nodo()**
26. **fin** *para*
27. **fin** *mientras*
28. **Calcular_mejor_solucion_local()**

//fin de la segunda etapa

29. **Calcular_mejor_solución_del_ciclo**
30. *si* **Mejor_solución_del_ciclo < Mejor_solución_global**
31. **Mejor_solución_global = Mejor_solución_del_ciclo**
32. **fin** *si*
33. **Actulizar_feromona()**

34. **fin** *mientras*
35. **fin** Procedimiento

Uno de los elementos a resaltar es el hecho de que la condición de parada de las hormigas en la primera etapa viene dada por que hayan resuelto una parte del problema en dependencia del valor del factor r, para este problema en especifico se la condición de parada de la primera etapa es que se completen la una cantidad de trabajos, que no es más que el total de trabajos que se tiene multiplicado por el factor r. Otro elemento a tener en cuenta es que al finalizar la primera etapa se eligen, dentro de todas las sub-soluciones encontradas por las ch1 hormigas, solamente algunas. Luego en el comienzo de la segunda etapa, al crear las hormigas que tomarán parte en dicho proceso, al método Crear_nueva_hormiga no se le pasa como parámetro un nodo inicial, sino una sub-solución escogida aleatoriamente entre las que fueron guardadas de la primera etapa. Este método le asigna a cada nueva hormiga de la segunda etapa una sub-solución de la primera que pasa a ser su propia solución, incompleta por el momento, y su nodo inicial vendría siendo el último nodo de la sub-solución.

Otro elemento muy importante a tener en cuenta es el hecho de que Max-Min AS en II Etapas trabaja con la misma heurística, el mismo tipo de actualización del vector de feromonas y los mismos parámetros que el Max-Min AS tradicional. Tampoco le hace modificaciones a la representación del problema WPP. Luego, si existe diferencia entre ellos con respecto a la calidad de las soluciones se debe enteramente a las modificaciones hechas en el método executeWPP para dividir en dos el espacio de búsqueda y nada más.

En la Figura 9 correspondiente a la sección Anexos se puede consultar el diagrama de clases de este algoritmo.

2.4 Conclusiones.

En el estudio presentado en este capítulo se pudo encontrar una modelación para aplicarle ACO al Problema de Planificación de las Fuerzas de Trabajo, a pesar de ser un problema con muchas restricciones. Se logró adecuar la nueva estrategia de búsqueda en II etapas a este problema, así como realizar sendas implementaciones de los algoritmos MMAS y TS-MMAS.

Capítulo 3. Presentación de los resultados

En el presente capítulo serán expuestos los resultados experimentales obtenidos al aplicarle Max-Min AS y Max-Min AS en II Etapas al WPP, así como la comparación entre estos resultados y los que existen de este problema que se obtuvieron mediante la utilización de Algoritmos Genéticos para llegar a conclusiones acerca del comportamiento de la metaheurística ACO para este problema.

3.1 Herramientas y tecnología para la implementación.

El lenguaje de programación utilizado en este trabajo fue el Java. Desarrollado por Sun Microsystems, es actualmente libre, siendo esto muy conveniente para su uso en el desarrollo de aplicaciones en los países del tercer mundo. Su mayor ventaja es la independencia de plataforma, un programa hecho en Java corre indistintamente en cualquiera de los ordenadores disponibles en el mercado, así funcionen sobre Windows, Linux, Mac o cualquiera de los sistemas operativos más comunes. Otras cualidades a resaltar son su seguridad, uno mismo puede implementar las directivas de seguridad de la aplicación; su estabilidad, es virtualmente imposible que una aplicación Java dé problemas de memoria o de asignación de recursos; es orientado a objetos y fácil de aprender, todo esto sin mencionar las enormes ventajas que tiene en el entorno web, que no vienen al caso en este trabajo.

En fin, todas estas cualidades mencionadas hicieron de Java el lenguaje de programación más adecuado para este trabajo, sobre todo por el inmenso volumen de datos a manejar, lo cual para Java no es problema.

Entre los IDE disponibles para Java fue seleccionado el NetBeans. Es un ambiente de programación muy cómodo, que compila en tiempo real y muy fácil de usar a la hora de depurar un programa.

3.2 Fuentes de datos.

Las fuentes de datos utilizadas para probar estos algoritmos son las mismas bases de datos que se utilizaron en la implementación y prueba de este problema mediante

Algoritmos Genéticos. Un ejemplo de cómo están estructuradas estas bases de casos es el siguiente:

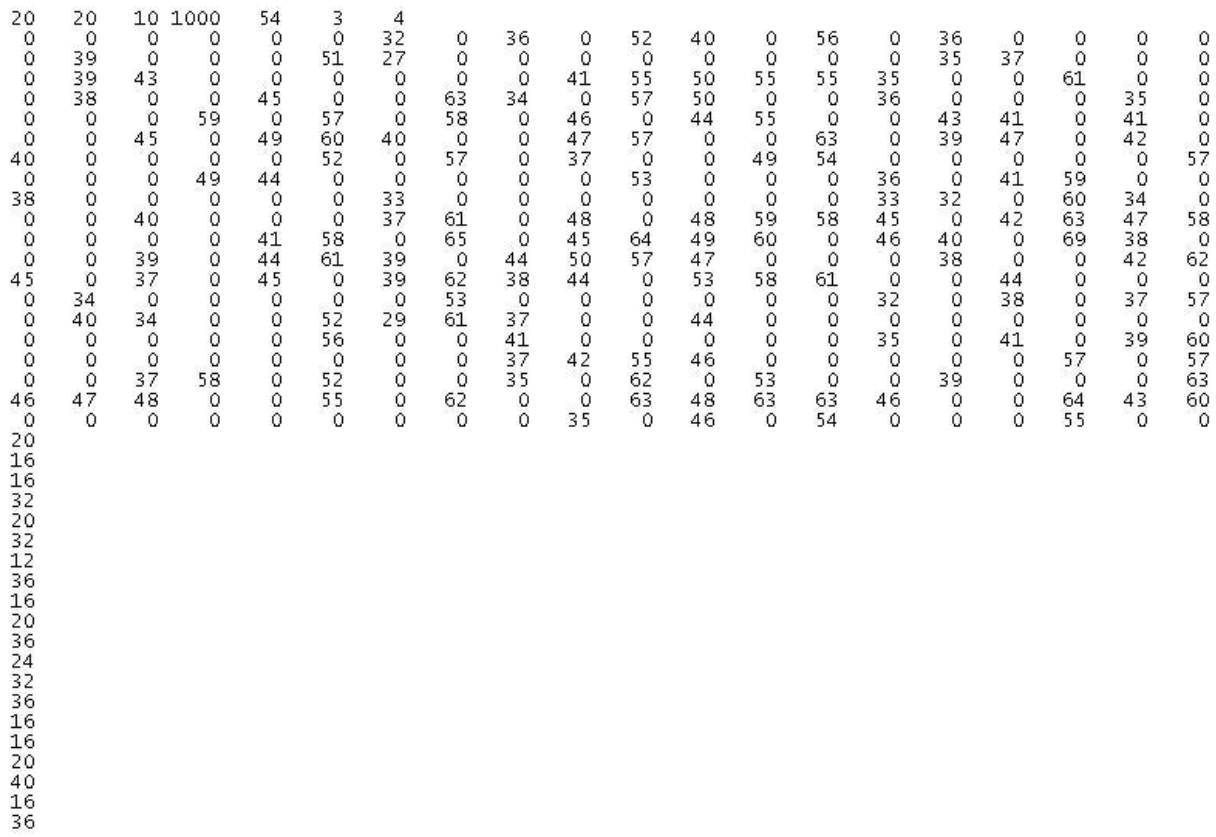


Figura 5. Estructura de las bases de casos.

Para que quede claro que significa cada elemento de la bases de casos:

La primera fila de la base de caso contiene por su respectivo orden la siguiente estructura:

- Número de trabajadores, número de trabajos, número máximo de trabajos que puede contener una solución, un dato que no se usa, tiempo máximo que puede ser asignado un trabajador, máximo número de tareas que puede realizar un trabajador, tiempo mínimo que puede ser asignado un trabajador.

A continuación le sigue la matriz de costo. Hay que resaltar que esta matriz es del orden de número de trabajadores *x* número de trabajos y que en las posiciones donde contiene un cero significa que el trabajador no está calificado para realizar dicho trabajo, como se había explicado anteriormente.

Por último, la duración de los trabajos.

Para la validación de estos algoritmos se cuenta con veinte bases de casos para este problema, y para cada una de ella se analizaran los resultados que arrojan estos algoritmos.

3.3 Resultados experimentales.

A continuación se muestra la tabla con los resultados obtenidos al aplicarles los algoritmos a las bases de datos mencionadas. Los parámetros usados en los algoritmos fueron obtenidos luego de muchos y largos procesos de pruebas y observaciones y finalmente resultaron: constante de evaporación = 0.7, cantidad de hormigas = 20, para el caso del algoritmo AS en II Etapas el factor de proporcionalidad r se utilizaron varios valores 0.2, 0,25, 0,3 y para r = 0,2 fue el que brindó mejores resultados, y el otro parámetro se especifica en la tabla. En el caso de alfa (α) y beta (β) se usaron distintos valores siendo alfa (α)= 2, beta (β)= 3; alfa (α)= 3, beta (β)=2, siendo (α)= 2, (β)= 3 los que brindaron mejores resultados. Como ya se dijo, estos resultados fueron obtenidos usando como lenguaje de programación Java, IDE NetBeans versión 6.0.1 y bajo el sistema operativo Windows 2003 Advanced Server en una computadora que cuenta con 3GHz de velocidad del procesador y 1 Gb de memoria RAM. Los resultados mostrados en la tabla fueron tomados como el promedio de 10 ejecuciones aleatorias de los algoritmos, además se muestran las mejores soluciones para cada algoritmo.

Bases de Casos	MMAS-WPP		TS-MMAS-WPP	
	Mejor Solución	Costo promedio	Mejor Solución	Costo promedio
s20_01	858	***892.9***	879	923.8
s20_02	933	***956***	932	965.5
s20_03	946	***959***	957	983.8
s20_04	981	***994.3***	1002	1022.1
s20_05	990	***1013***	1012	1038.5
s20_06	936	***950.3***	929	962.6
s20_07	942	***963.7***	947	971.1
s20_08	1054	***1073.3***	1020	1083.5
s20_09	912	***938.8***	932	980.4
s20_10	873	***891.3***	888	915.8
u20_01	1076	***1280.9***	1048	1307.1
u20_02	1000	***1061.1***	1006	1167.4
u20_03	992	***1065.8***	1063	1120.4
u20_04	1093	***1207.9***	1115	1357.1
u20_05	935	***1021.3***	1012	1195.5
u20_06	961	1088.7	984	***1068.9***
u20_07	930	***993.5***	977	1096.8
u20_08	1133	1308.7	992	***1257.9***
u20_09	807	***821.3***	815	864.4
u20_10	927	***964.6***	927	997.1

Tabla 1. Resultados experimentales del Max-Min AS y Max-Min AS II Etapas.

Como se puede apreciar en la tabla anterior los mejores resultados para este problema se obtienen con el Max-Min AS y no con el Max-Min AS en II Etapas, a pesar de que se gana en tiempo de ejecución. Este problema está dado porque en la primera etapa del Max-Min AS en II Etapas la condición de parada está basada en que se complete un número de tareas, lo cual es el objetivo, pero estas tareas no se resuelven con un número mínimo de trabajadores y el costo de estas subsoluciones por lo general son muy similares, no existiendo una gran diferencia entre ellos, lo que hace muy difícil la selección de las mejores subsoluciones.

3.3.1 Modificación del Max-Min AS en II Etapas

A raíz de esta problemática se plantea una mejora para el Max-Min AS en II Etapas. Dicha mejora no difiere mucho de este, por lo general son muy similares, solo difieren en el hecho de que en la nueva variante no se almacenan las subsoluciones encontradas en la primera etapa, sino que se actualiza solamente la matriz de feromona. Todo este proceso se realiza de forma similar, utilizando solo un porciento de hormigas y un porciento de ciclos. Luego en la segunda etapa se comienza a partir de la selección de un nodo inicial, como en el Max-Min AS tradicional, pero con las hormigas restantes y los ciclos pendientes.

De forma general esta variante hace un proceso de marcado de pedazos de sub-soluciones que aportan menos costo a la solución final, este proceso tiene lugar en la primera etapa a través de la actualización de los rastros de feromona. De esta forma las mejores sub-soluciones no tienen que estar en el inicio de la solución global, pueden formar parte en cualquier lugar de ésta.

En la Figura 10 de la sección Anexos se puede observar el diagrama de clases correspondiente a esta modificación. Esta modificación al igual que el Max-Min en II Etapas obtiene ganancias en cuanto al tiempo y logra a su vez buenas soluciones, estas se pueden apreciar a continuación:

Bases de Casos	MMAS		TS-MMAS		MTS-MMAS	
	Mejor Solución	Costo promedio	Mejor Solución	Costo promedio	Mejor Solución	Costo promedio
s20_01	858	***892.9***	879	923.8	889	897.7
s20_02	933	956	932	965.5	927	***944***
s20_03	946	***959***	957	983.8	948	972.2
s20_04	981	994.3	1002	1022.1	942	***951.5***
s20_05	990	1013	1012	1038.5	984	***993.9***
s20_06	936	***950.3***	929	962.6	953	978.9
s20_07	942	963.7	947	971.1	918	***934.2***
s20_08	1054	1073.3	1020	1083.5	985	***994***
s20_09	912	938.8	932	980.4	891	***909.9***
s20_10	873	891.3	888	915.8	886	***890.5***
u20_01	1076	1280.9	1048	1307.1	924	***951.9***
u20_02	1000	1061.1	1006	1167.4	952	***960.1***
u20_03	992	1065.8	1063	1120.4	986	***1025.1***
u20_04	1093	1207.9	1115	1357.1	1000	***1023***
u20_05	935	1021.3	1012	1195.5	948	***969.2***
u20_06	961	1088.7	984	1068.9	1002	***1016.4***
u20_07	930	993.5	977	1096.8	939	***970.5***
u20_08	1133	1308.7	992	1257.9	922	***940.7***
u20_09	807	***821.3***	815	864.4	805	832.4
u20_10	927	964.6	927	997.1	914	***945.2***

Tabla 2. Resultados experimentales de los tres algoritmos ACO aplicados.

Como se puede apreciar en la tabla la modificación al Max-Min AS en II Etapas mejora en gran medida los resultados obtenidos. Para poder observar mejor lo antes mencionado, obsérvese la siguiente grafica:

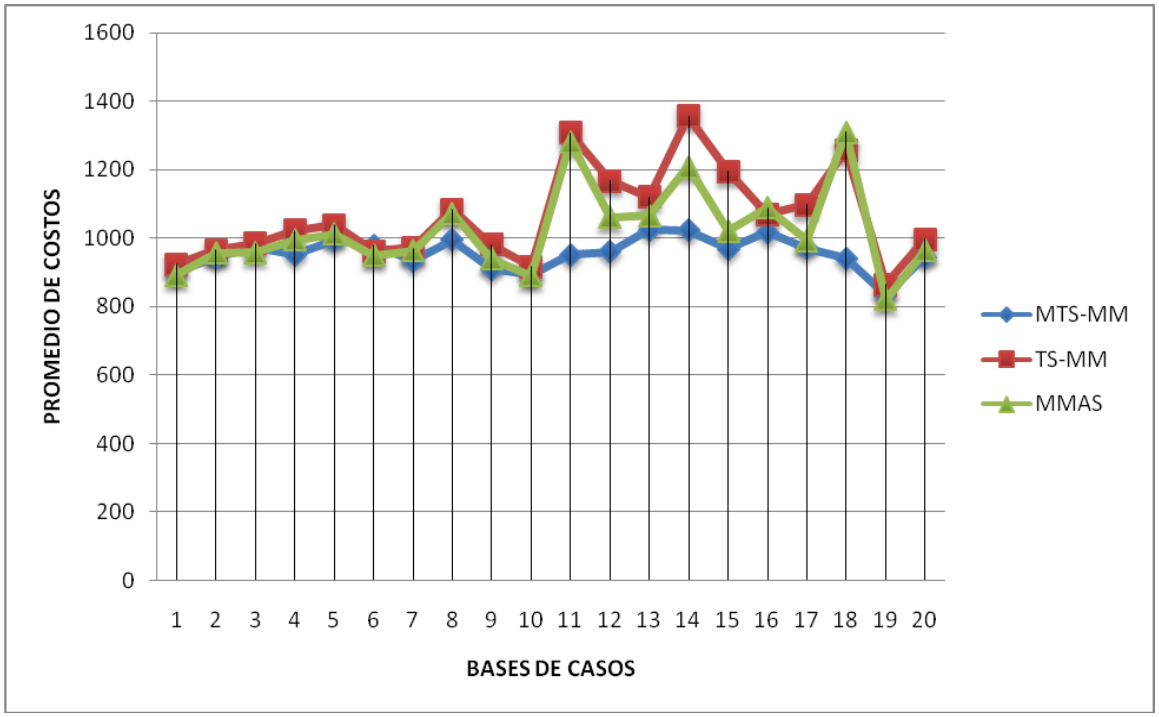


Figura 6. Gráfica que representa los resultados de los tres algoritmos aplicados.

3.3.2 Comparación de los resultados obtenidos con los obtenidos utilizando Algoritmos Genéticos.

Los resultados asociados a AG se pueden encontrar en(Enrique Alba 2007b), los que se muestran a continuación son los referentes a la implementación de AG de forma secuencial.

Bases de Casos	AG-WPP	MMAS-WPP	TS-MMAS-WPP	MTS-MMAS-WPP
s20_01	963	***892.9***	923.8	897.7
s20_02	994	956	965.5	***944***
s20_03	1156	***959***	983.8	972.2
s20_04	1201	994.3	1022.1	***951.5***
s20_05	1098	1013	1038.5	***993.9***
s20_06	1193	***950.3***	962.6	978.9
s20_07	1086	963.7	971.1	***934.2***
s20_08	1287	1073.3	1083.5	***994***
s20_09	1107	938.8	980.4	***909.9***
s20_10	1086	891.3	915.8	***890.5***
u20_01	1631	1280.9	1307.1	***951.9***
u20_02	1264	1061.1	1167.4	***960.1***
u20_03	1539	1065.8	1120.4	***1025.1***
u20_04	1603	1207.9	1357.1	***1023***
u20_05	1356	1021.3	1195.5	***969.2***
u20_06	1205	1088.7	1068.9	***1016.4***
u20_07	1301	993.5	1096.8	***970.5***
u20_08	1106	1308.7	1257.9	***940.7***
u20_09	1173	***821.3***	864.4	832.4
U20_10	1214	964.6	997.1	***945.2***

Tabla 3. Resultados obtenidos utilizando AG y ACO.

Como se podrá observar continua la *mejora* al Max-Min en II Etapas ofreciendo los mejores resultados, a continuación se muestra una grafica donde se podrá visualizar mejor esta conclución:

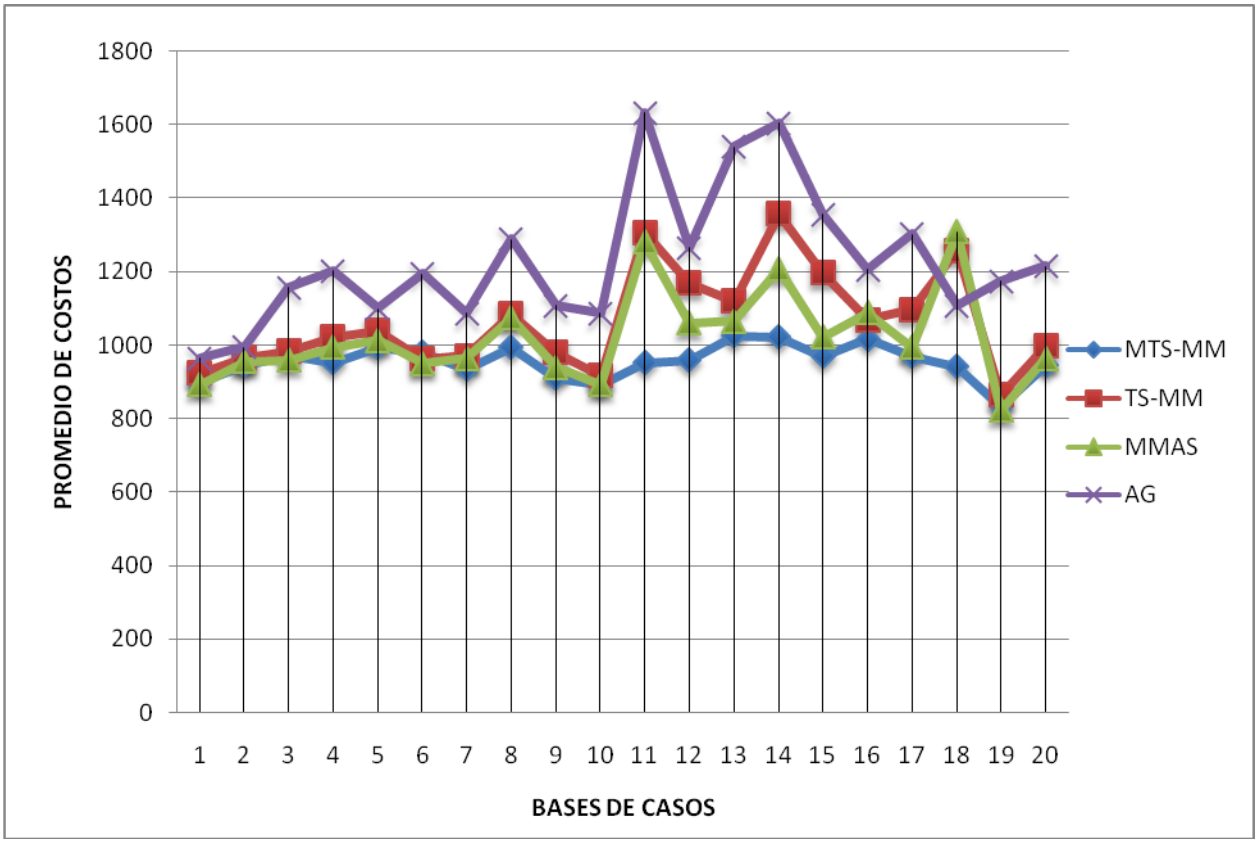


Figura 7. Gráfica que representa los resultados de los tres algoritmos ACO aplicados y los de AG.

3.4 Análisis estadístico de los resultados.

Para validar los resultados obtenidos se utilizaron técnicas estadísticas que justifican la veracidad de los resultados y no dejarlo solamente a la apreciación. Para el análisis estadístico se escogió el valor promedio de las 10 corridas de cada base de datos y las pruebas utilizadas fueron el test de Friedman que sirve para comparar *J* promedios poblacionales cuando se trabaja con muestras relacionadas y el test Wilcoxon par a par, para en caso de diferencias significativas en los grupos, detectar en cuales muestras están las diferencias. Las tablas resumen las dos pruebas, donde la primera columna representa los algoritmos utilizados y la columna "Rangos Medios" muestra el ranking del test de Friedman, la letra de superíndice las diferencias significativas par a par encontradas por el test de Wilcoxon según el orden en el abecedario.

	Rangos Medios
MTS-MMAS-WPP(r=20)	1.90^{a}
MTS-MMAS-WPP(r=25)	2.24^{ab}
MTS-MMAS-WPP(r=30)	3.00^{b}
MMAS-WPP	3.76^{c}
TS-MMAS-WPP(r=20)	4.67^{d}
TS-MMAS-WPP(r=25)	6.33^{e}
TS-MMAS-WPP(r=30)	6.38^{e}
AG-WPP	7.71^{f}

Aunque no se trazó como objetivo la disminución del tiempo de ejecución conseguido por la incorporación de la estrategia de búsqueda en II etapas a la metaheurística

ACO, aquí mostramos el análisis estadístico para el tiempo de ejecución de los algoritmos presentados.

	Rangos Medios
MTS-MMAS-WPP(r=30)	1.05^{a}
MTS-MMAS-WPP(r=25)	2.24^{b}
TS-MMAS-WPP(r=30)	2.94^{c}
MTS-MMAS-WPP(r=20)	3.90^{d}
TS-MMAS-WPP(r=25)	4.86^{e}
TS-MMAS-WPP(r=20)	6.05^{f}
MMAS-WPP	6.95^{g}

3.5 Conclusiones.

La metaheurística ACO es una buena técnica para resolver el problema de Planificación de la Fuerzas de Trabajo, ya que como se pudo observar en los resultados, es capaz de encontrar mejores soluciones que las presentadas por los algoritmos genéticos.

La nueva estrategia de exploración en II Etapas trae consigo un decremento significativo del tiempo computacional y aunque en su versión original no mejora las soluciones encontradas con ACO, por las caraterística del problema WPP, la modificación presentada en este capítulo si supera en cuanto a calidad de soluciones al ACO tradicional.

Se presenta un estudio de caraterística del espacio de búsqueda para los cuales es más eficiente el uso de la modificación de ACO en II etapas.

Estos elementos quedan validados con el uso de análisis estadísticos mediante el test de Friedman para un grupo de muestras relacionadas y el test de Wilcoxon para el análisis par a par.

Conclusiones Generales

Como resultado de este trabajo se logró realizar una modelación del Problema de Planeamiento de las Fuerzas de Trabajo que permitió la aplicación de la Metaheurística ACO al mismo.

Se implementaron satisfactoriamente 2 variantes de la metaheurística ACO a este problema, Max-Min AS y Max-Min AS en II Etapas, así como una modificación al Max-Min AS en II Etapas.

Se comprobó mediante el análisis de los resultados que la modificación planteada para el Max-Min AS en II Etapas obtenía los resultados más significativos para este problema en específico, ya fuese con respecto a los demás algoritmos ACO aplicados como a los resultados que se tenían de dicho problema utilizando Algoritmos Genéticos.

Recomendaciones

Como recomendaciones de este trabajo numeramos los siguientes aspectos:

1. Realizar una nueva modelación del problema WPP para aplicarle la metaheurística ACO, con motivo de comparar los resultados de este trabajo.
2. Aplicar otros algoritmos ACO, tales como Ant System y Ant Colony System que han sido probados en la solución de otros problemas, obteniendo buenos resultados.
3. Hacer un estudio de métodos de búsqueda local que puedan ser aplicados al problema WPP, en aras de mejorar las soluciones encontradas por las hormigas.
4. Realizar la implementación de una versión paralela de los algoritmos aquí expuestos para comparar los resultados con las versiones paralelas de otros algoritmos presentados en la bibliografía para la solución de este problema.

Referencias Bibliográficas

A, M. V. y. C. (1999) The Ant System applied to the Quadratic Assignment Problem. In Transactions on Knowdledge and Data Engineering. lEEE

Estados Unidos.

A. Díaz, F. G., H.M. Ghaziri, J.L. Gonzalez, M. Laguna, P. Moscato and Tseng, and F.T. (1996) Optimización Heurística y Redes Neuronales, Paraninfo, Madrid.

B. Bullnheimer, R. F. H., y C. Strauss (1999) A new rank-based version of the Ant System: A computational study. Central European Journal for Operations Research and Economics, 7(1): 25-38.

Barán, B. and Almirón, M. (2000) Colonias de Hormigas en un entorno paralelo asíncrono. In XXVI Conferencia Latinoamericana de Informática - CLEI'00. México.

Bello, R., Puris, A., Nowé, A., Martínez, Y. and García:, M. M. (2006) Two Step Ant Colony System to Solve the Feature Selection Problem. In CIARP 2006.

Birattari, M., Stützle, T., Paquete, L. and Varrentrapp, K. (2002) In GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference(Ed, W.B. Langdon y otros, e.) Morgan Kaufmann Publishers, San Francisco, CA, USA, pp. 11-18.

Bonabeau, E., Dorigo, M. and Theraulaz, G. (1999) Swarm Intelligence: From Natural to Artificial Systems. In Oxford University Press. New York, NY.

Bullnheimer B., H. R. y. S. C. (1999) An improved ant system algorithm for the vehicle routing problem. In Annals of Operations Research. (Ed, Dawind, F. a. H.) Nonlinear Economic Dynamics and Control.

Cáceres, A. P., Bello, R., Nowe, A. and Jiménez., Y. M. (2006) Two-Stage Ant Colony System for solving the Traveling Salesman Problem Mexico.

Caro, M. D. y. G. D. (1999) In New Ideas in Optimization(Ed, Hill, M.) London UK, pp. 11-32.

Dorigo, G. D. C. a. M. (1998) "Mobile Agents for Adaptive Routing. In Proceedings of the 31st Hawaii International Conference on System. Hawaii.

Dorigo, M. and Caro, G. D. (1999) In New Ideas in Optimization(Ed, D. Corne, M. D. y. F. G., editores) McGraw Hill, London, UK, pp. 11-32.

Dorigo, M., Caro, G. D. and Gambardella, L. M. (1999) In Artificial Life, pp. 137-172.

Dorigo, M. and Gambardella, L. M. (1997) In lEEE Transactions on Systems, Man, and Cybernetics - Part B, Vol. 26 : 1, pp. 29-41.

Dorigo, M., Maniezzo, V. and Colorni, A. (1996) In lEEE Transactions on Systems, Man, and Cybernetics - Part B, Vol. 26:1, pp. 29-41.

Dorigo, M. and Stützle, T. (2003) In Handbook of Metaheuristics(Ed, F. Glover and G. Kochenberger, e.) Kluwer Academic Publishers, pp. 251-285.

Dorigo, M. and Stützle, T. (2004) Ant Colony Optimization. MIT Press: 303.

Dorigo M., M. V. y. C. A. (1996) The Ant System: Optimization by a colony of cooperating agents. EEE Transaction on Systems, Man, and Cybernetics-Part B, USA, 26(1): 1-13.

E. Bonabeau, M. D. a. T. T. (1999) From Natural to Artificial Swarm Intelligence, New York: Oxford University Press, New York.

Enrique Alba, G. L. (2007a) Parallel GA for Large Instances of the Workforce Planning Problem. Seventh International Conference on Intelligent System Design and Application.

Enrique Alba, G. L., Francisco Luna (2007b) Parallel Metaheuristics for Worforce Planning.

Gambardella, D. M. a. L. (1997a) Ant Colonies for the Traveling Salesman Problem. BioSystems, 43: 73-81.

Gambardella, É. D. T. y. L. M. (1997b) Adaptative memories for the quadratic assigment problem. In DSIA-97. Lugano, Switzerland,.

Gambardella, M. D. y. L. M. (1997c) Ant Colony System: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, 1(1): 53-66.

Goss, S., Aron, S., Deneubourg, J. L. and Pasteels, J. M. (1989) Self-organized shortcuts in the Argentine ant. In Naturwissenschaften. Vol. 76, pp. 579-581.

Hoos, T. S. y. H. H. (2000) MAX-MIN Ant System. Future Generation Computer Systems, 16(8): 889-914.

Johnson, M. R. G. a. D. S. (1979) Computers and Intractability: A Guide to the Theory of NP-Completeness, H. Freeman and Company, San Francisco.

Kelly, I. H. O. a. J. P. (1996) Meta-Heuristics: Theory and Applications, Ed. Kluwer Academic, Boston.

L. M. Gambardella, È. D. T., y M. Dorigo (1999) Ant colonies for the quadratic assigment problem. Journal of the Operational Research Society, 50(2): 167-176.

M, G. L. y. D. (1997) HAS-SOP: An Hybrid Ant System for the Sequential Ordering Problem. In Technical Report IDSIA11-97. Lugano - Suiza.

M. Dorigo, G. D. C., and L. M. Gambardella (1999) Ant algorithms for discrete optimization. Artificial Life, 5(2): 137-172.

M. Dorigo, V. M., and A. Colorni (1996) The Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems Man, and Cybernetics, 26: 29-41.

Michel, R. and Middendorf, M. (1998) In Proceedings of PPSN-V, Fifth International Conference on Parallel Problem Solving from Nature, Vol. 1498 (Ed, A. E. Eiben, T. B., M. Schoenauer, y H.-P. Schwefel, editores) Springer Verlag, Berlín, Alemania, pp. 692-701.

O. Cordón, I. F. d. V., F. Herrera, y L. Moreno (2000) new ACO model integrating evolutionary computation concepts: The Best-Worst Ant System. In ANTS 2000. (Ed, M. Dorigo, M. M., y T. Stützle) Université Libre de Bruxelles, Belgium,, pp. 22-29.

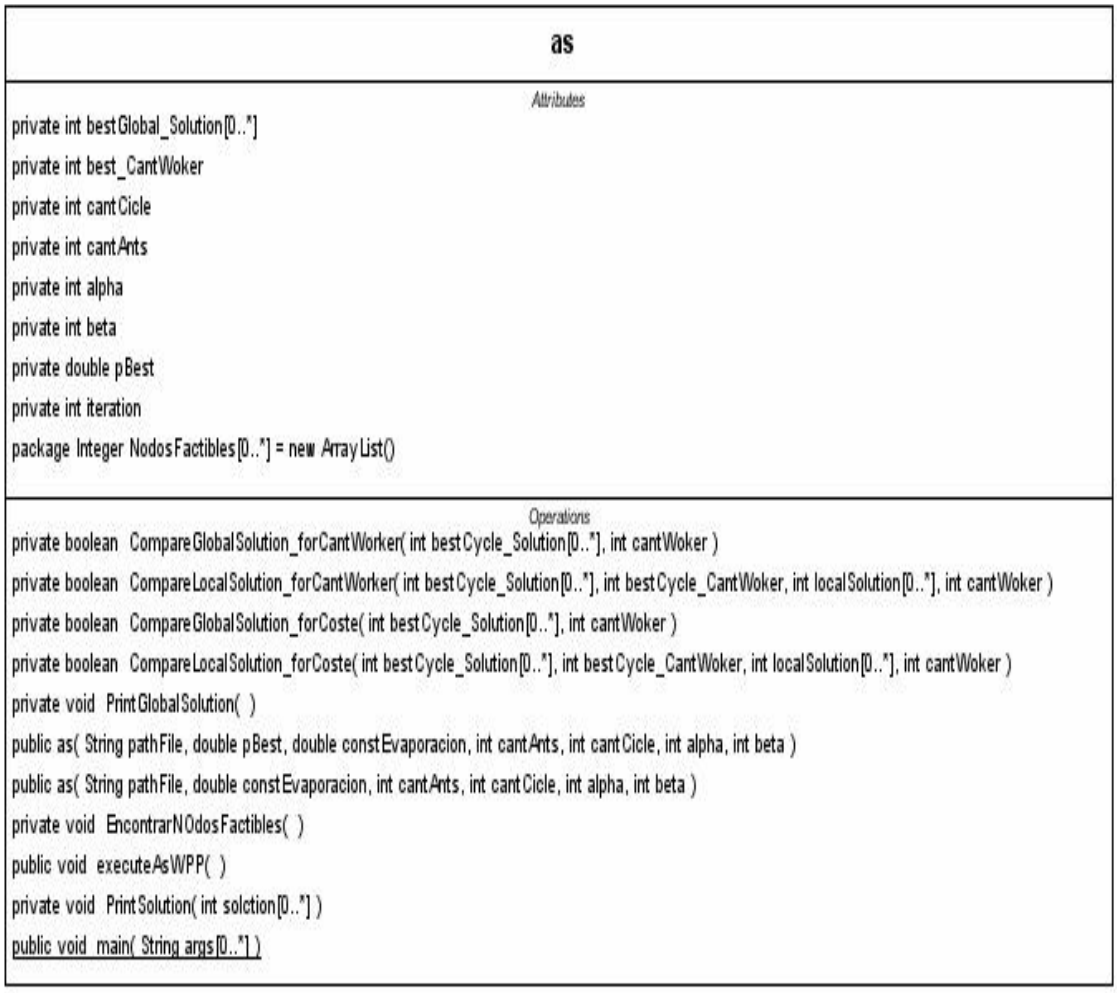
Pasteels, J. M., Deneubourg, J. L. and Goss, S. (1987) Self-organization mechanisms in ant societes (I): Trail recruitment to newly discovered food sources. In Experientia Supplementum. Vol. 54, pp. 155-175.

Sergio Alonso, O. C., Iñaki Fernández de Viana, Francisco Herrera (2003) La Metaheurística de Optimización Basada en Colonias de Hormigas: Modelos y Nuevos Enfoques. Vol. 1.

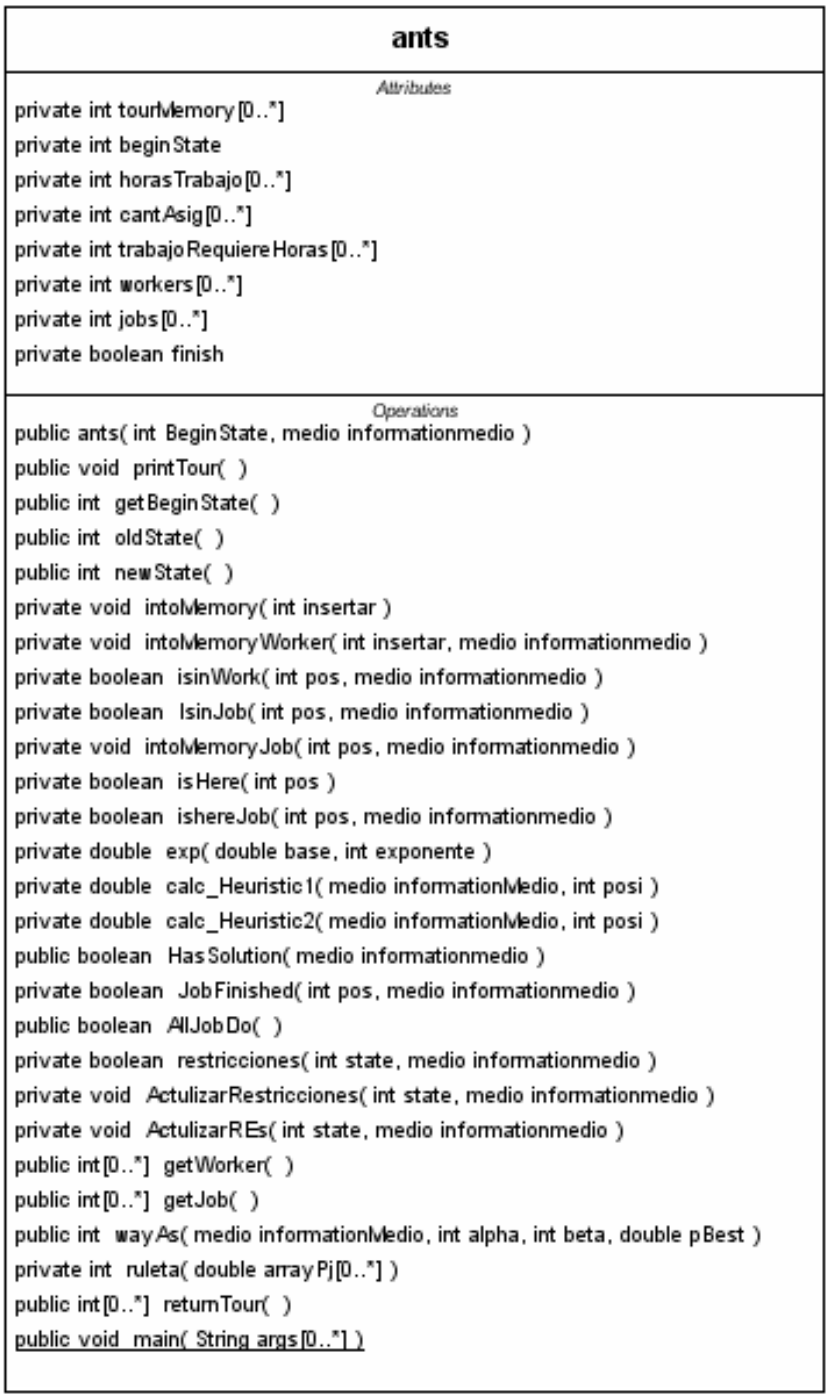
Stützle, T. and Hoos, H. H. (1997) In Proceedings of the 1997 IEEE International Conference on Evolutionary Computation (ICEC'97)(Ed, T. Bäck, Z. M., y X. Yao, editores) Piscataway, NJ, USA, pp. 309-314.

Stützle, T. and Hoos, H. H. (2000) In Future Generation Computer Systems, Vol. 16:8, pp. 889-914.

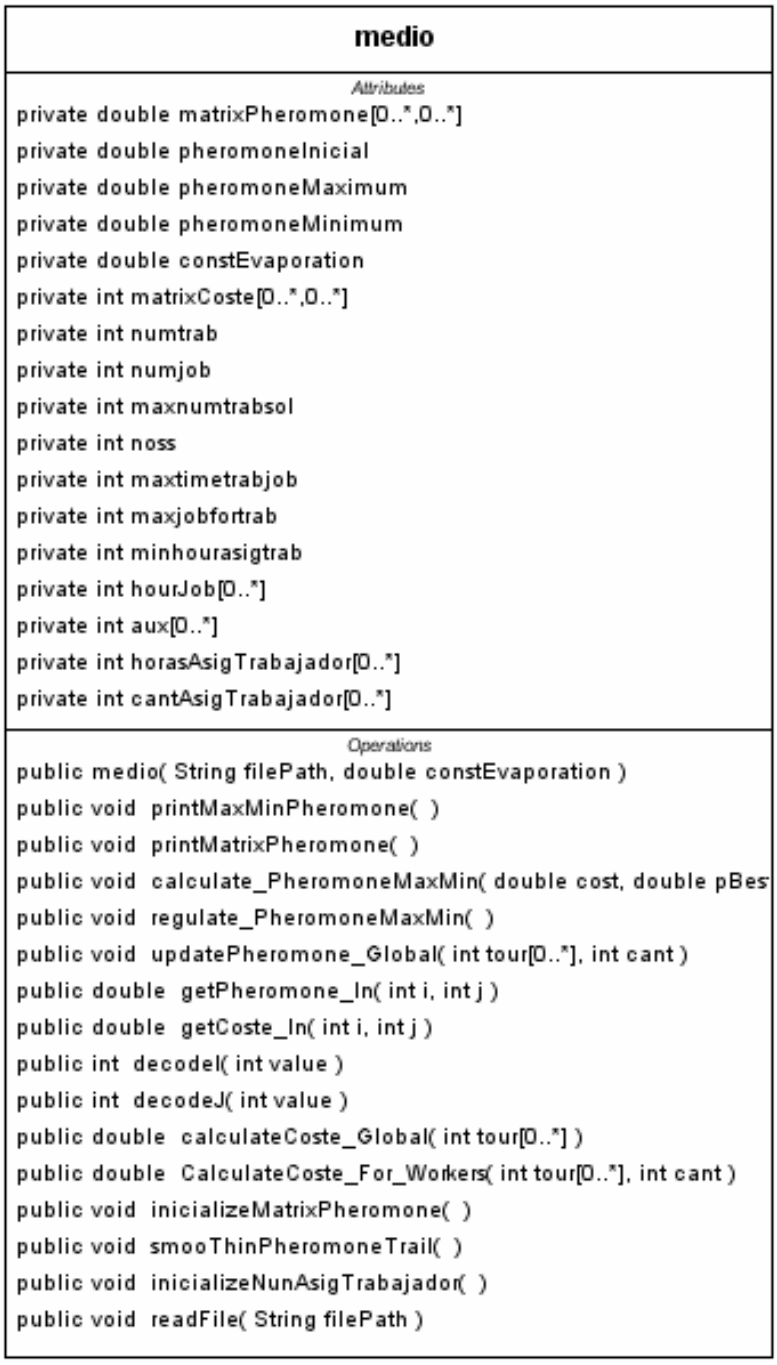
A continuación se muestran las distintas clases para cada algoritmo aplicado al WPP, así como el diagrama de clase de cada uno de ellos.



Clase 1. Clase as del Max-Min AS.



Clase 2. Clase ants del Max-Min AS.



Clase 3. Clase medio del Max-Min AS.

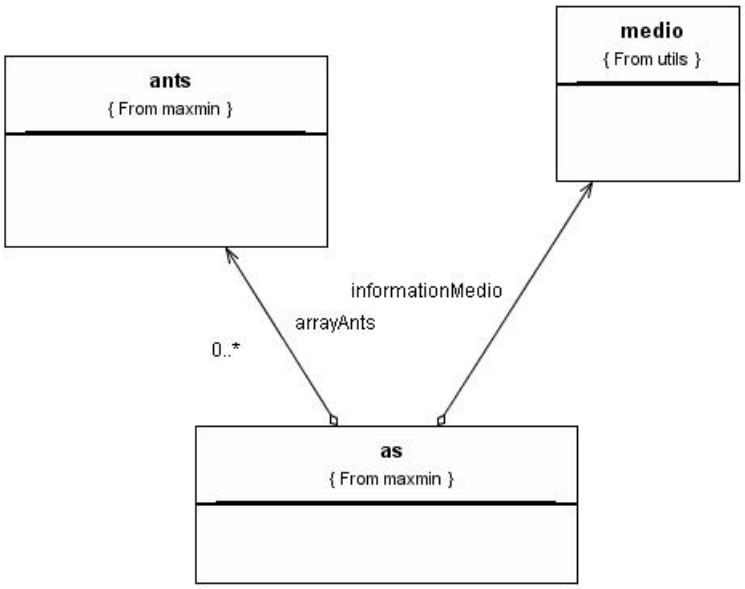
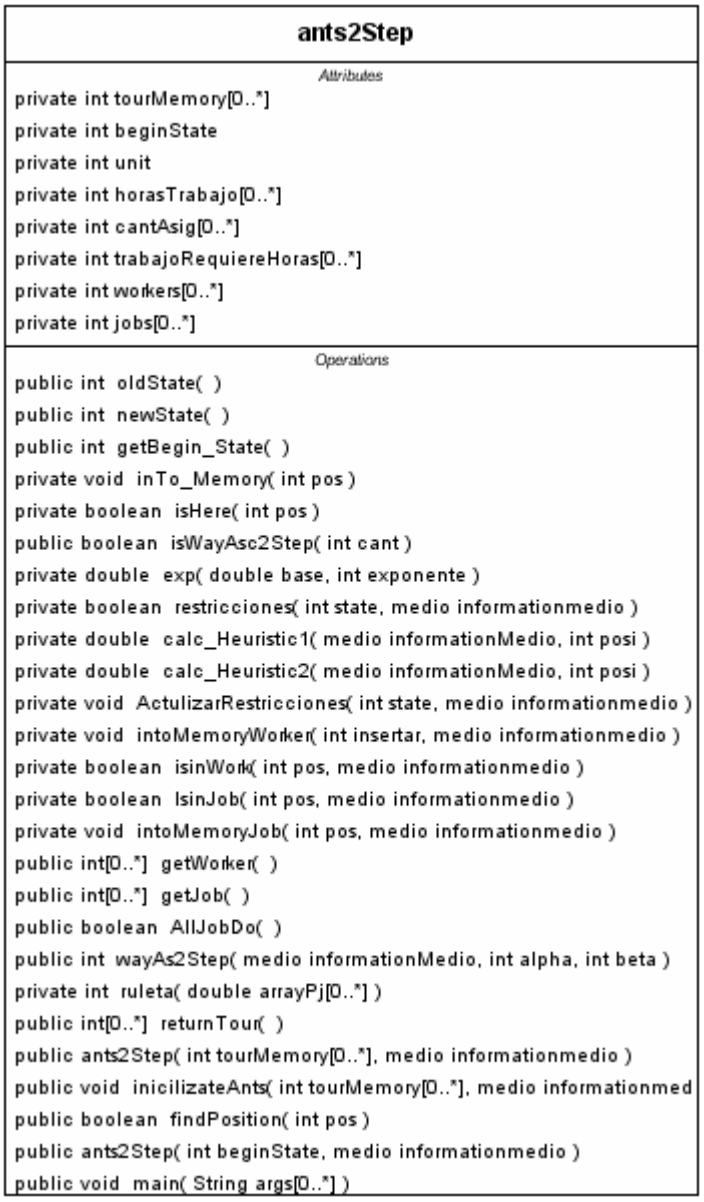
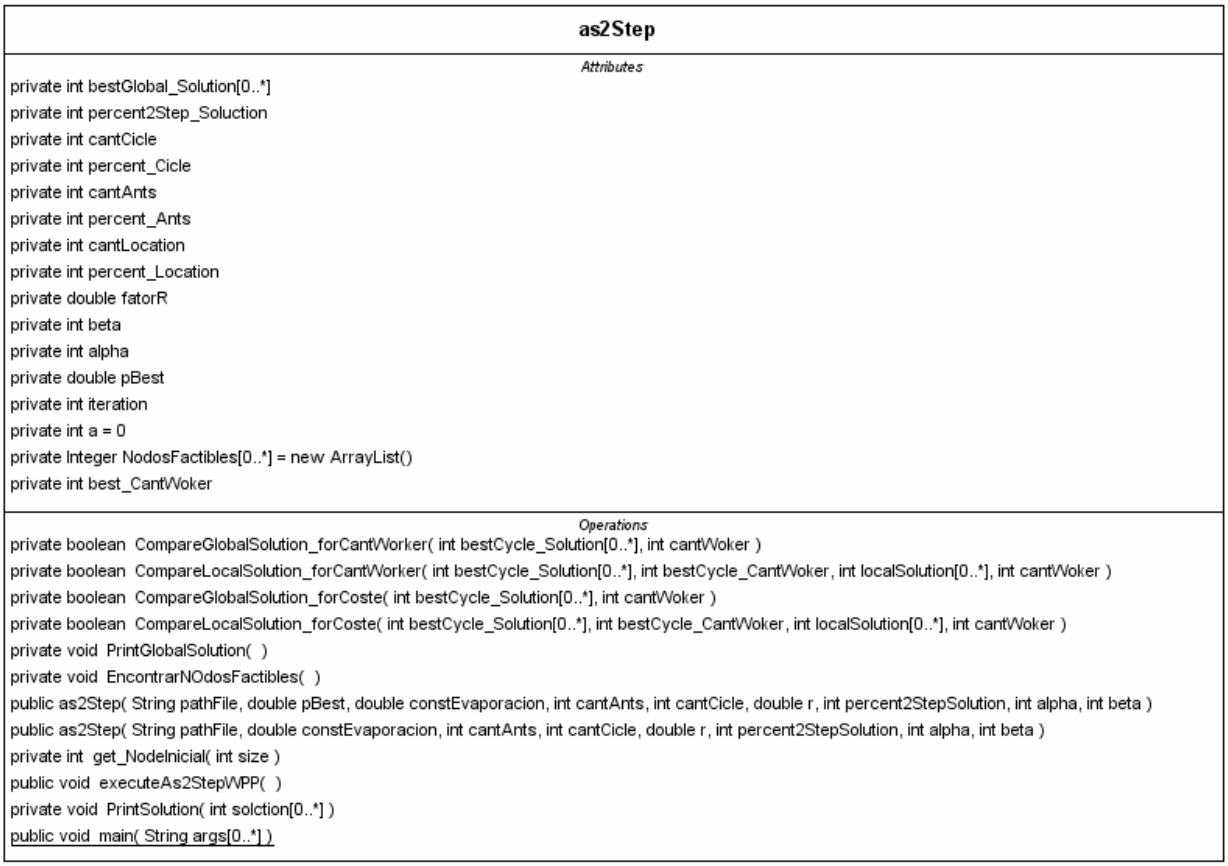


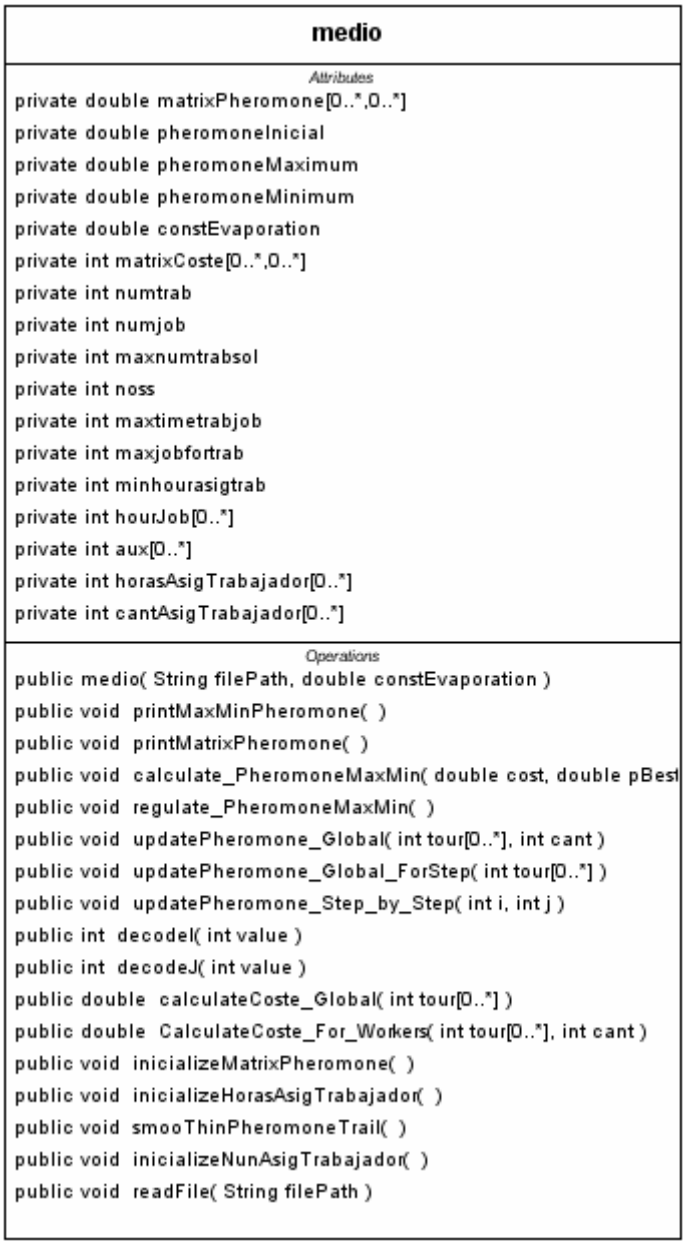
Figura 8. Diagrama de clases del Max-Min AS.



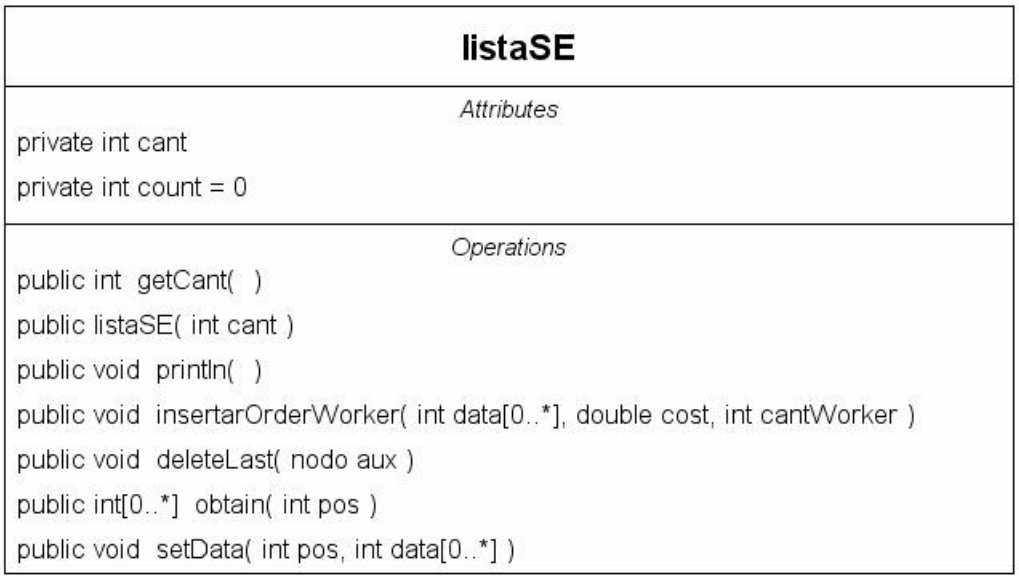
Clase 4. Clase ants para el Max-Min AS en II Etapas.



Clase 5. Clase as para el Max-Min AS en II Etapas.



Clase 6. Clase medio para el Max-Min AS en II Etapas.



Clase 7. Clase listaSE para el Max-Min AS en II Etapas.

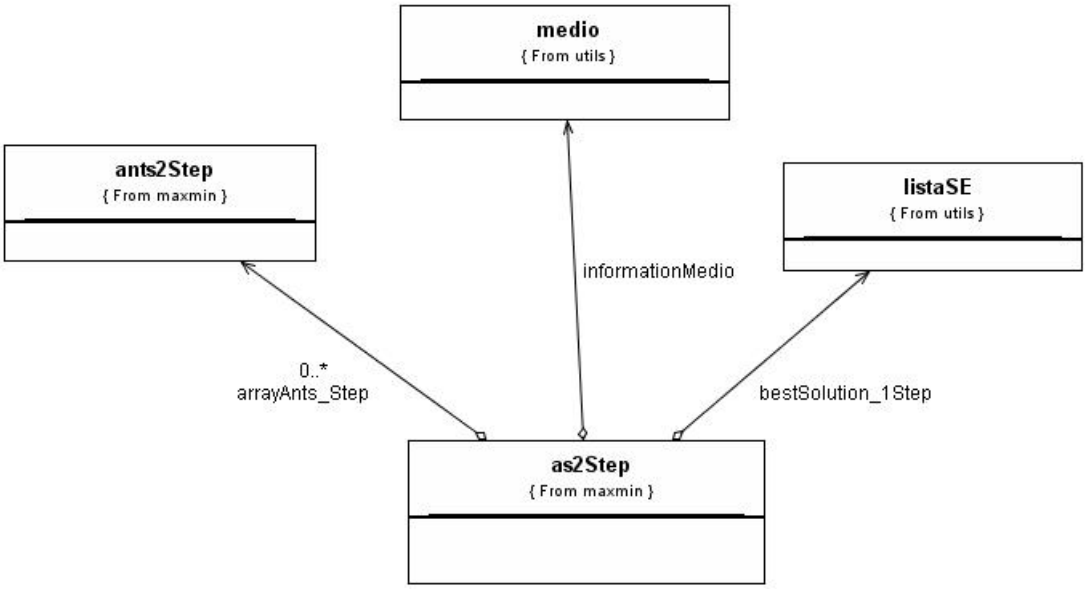
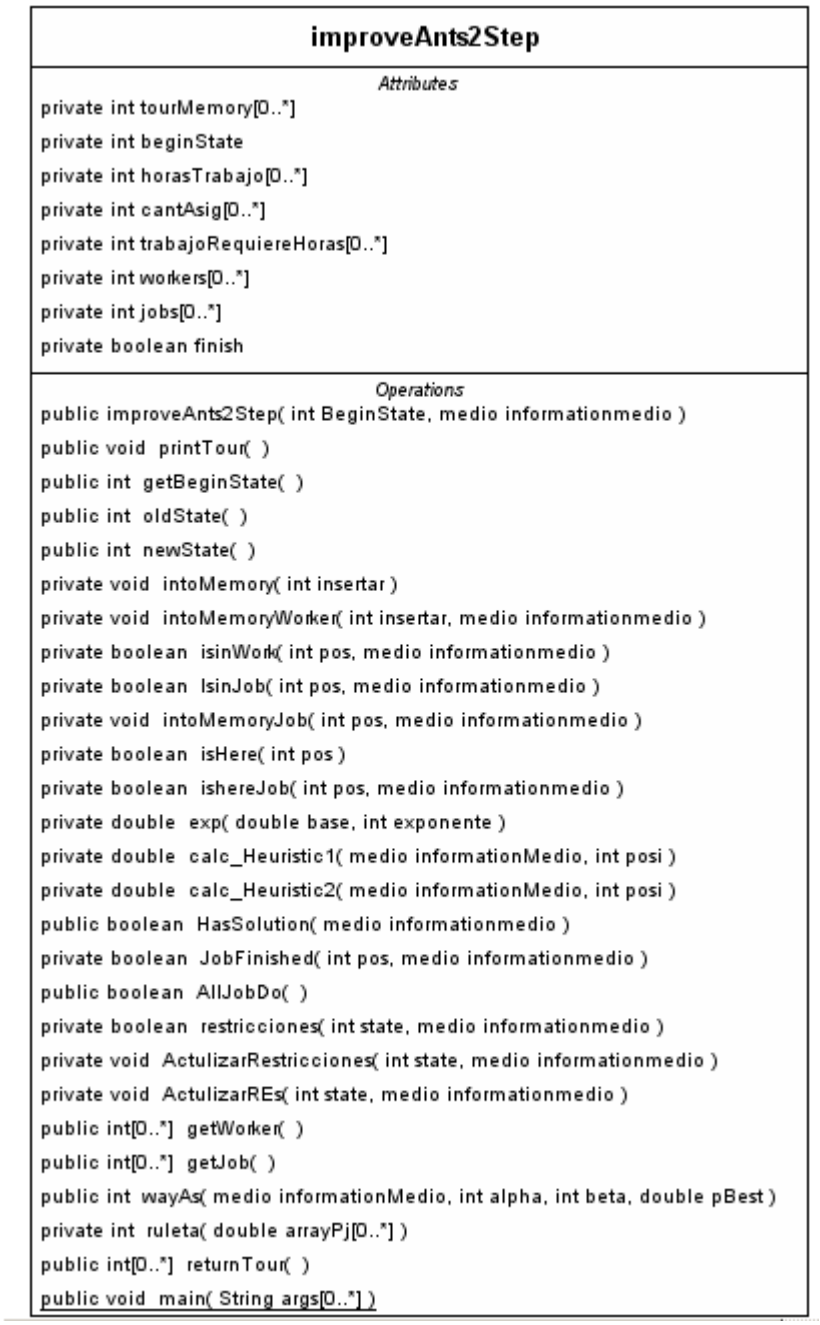
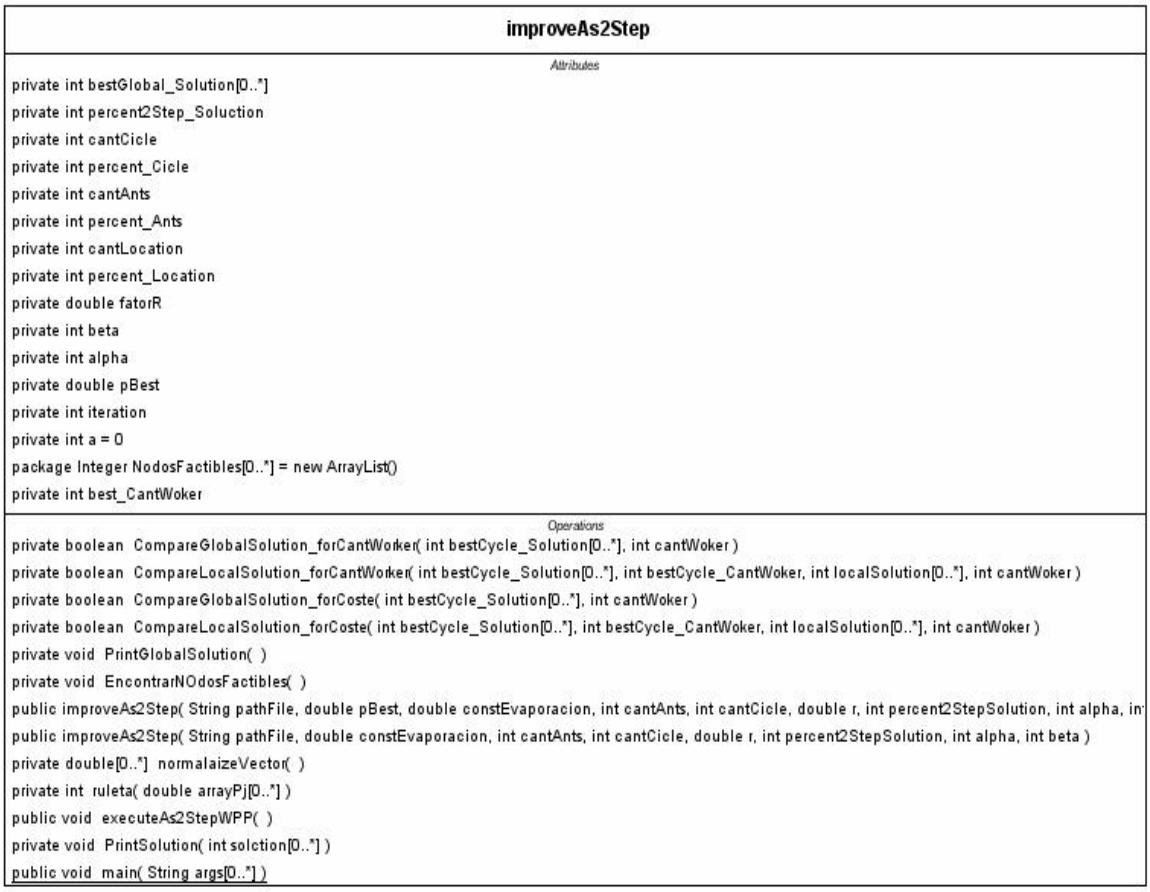


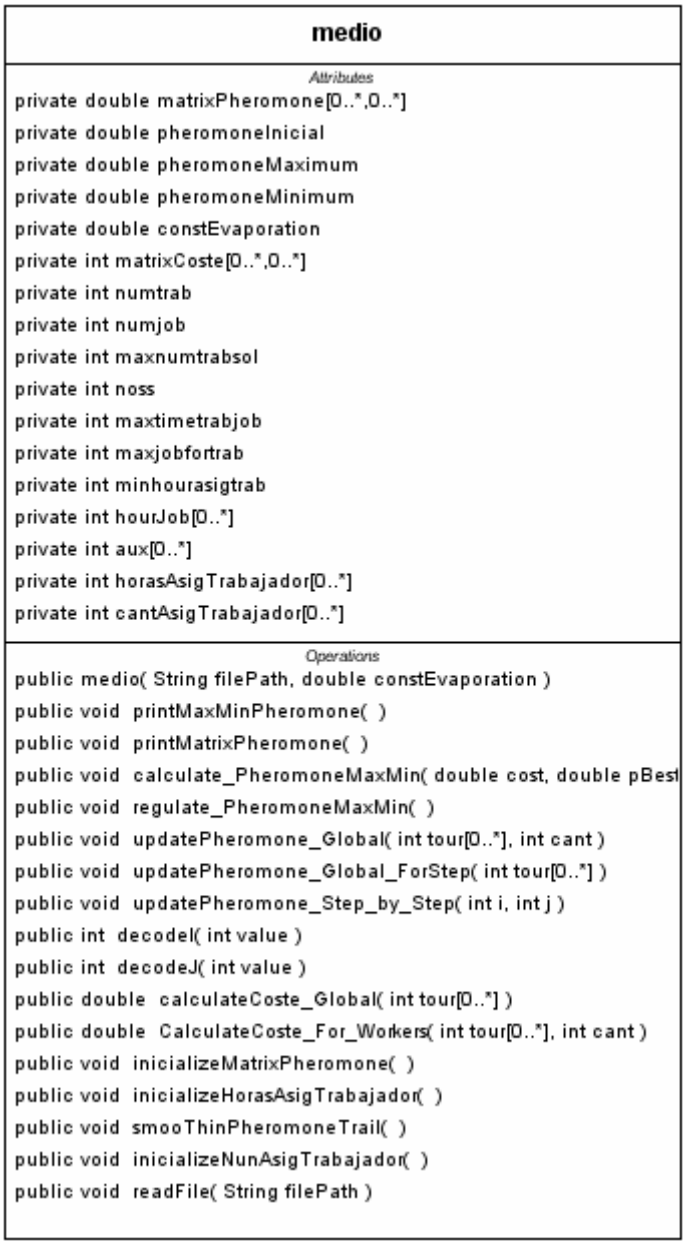
Figura 9. Diagrama de clases para el Max-Min AS en II Etapas.



Clase 8. Clase ants para la Modificación del Max-Min AS en II Etapas.



Clase 9. Clase as para la Modificación del Max-Min AS en II Etapas.



Clase 10. Clase medio para la Modificación del Max-Min AS en II Etapas.

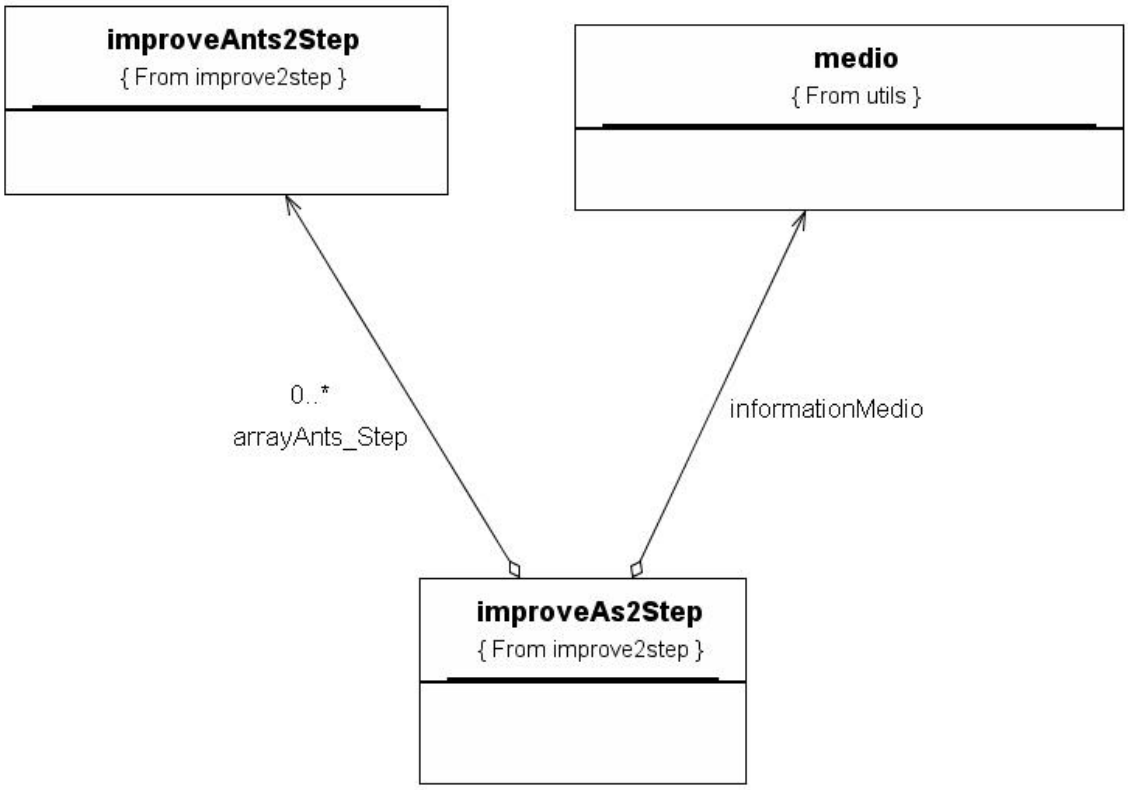


Figura 10. Diagrama de clases para la Modificación del Max-Min AS en II Etapas.