Colonias Distribuidas de Hormigas en un Entorno Paralelo Asíncrono

D.Sc. Benjamín Barán[1]
bbaran@cnc.una.py

Lic. Marta Almirón
malmiron@cnc.una.py

Centro Nacional de Computación
Universidad Nacional de Asunción
Casilla de Correos 1439
Campus Universitario de San Lorenzo
Paraguay

Resumen

El comportamiento colectivo de una colonia de hormigas ha servido para inspirar una novedosa técnica denominada Sistema de Hormigas *(Ant System)*, una familia de algoritmos distribuidos para optimización combinatoria. El método consiste en simular la comunicación indirecta que utilizan las hormigas para establecer el camino más corto desde su nido hasta la fuente de alimento y regresar.

Basado en el asincronismo implícito de una colonia de hormigas, el presente trabajo propone la paralelización de un sistema de hormigas en un ambiente típicamente asíncrono, como el de una red de computadoras, permitiendo que cada procesador trabaje en forma independiente, a su propio ritmo y sin depender de datos ajenos, compartiendo sus mejores resultados con los demás procesadores de la red y aprovechando buenos resultados de otros procesadores para mejorar su capacidad de búsqueda. Con esto, se logra aprovechar los recursos computacionales generalmente disponibles, mejorando considerablemente los resultados experimentales.

Palabras Claves:

Sistemas distribuidos y paralelismo, inteligencia artificial, agente, optimización combinatoria.

1. Introducción

Ant System (AS) se basa en el comportamiento colectivo de las hormigas en la búsqueda de alimentos para su subsistencia. Resulta fascinante entender como animales casi ciegos, moviéndose aproximadamente al azar, pueden encontrar el camino más corto desde su nido hasta la fuente de alimentos y regresar. Para esto, cuando una hormiga se mueve, deja una señal odorífera, depositando una substancia denominada feromona, para que las demás puedan seguirla.

En principio, una hormiga aislada se mueve esencialmente al azar, pero las siguientes deciden con una buena probabilidad seguir el camino con mayor cantidad de feromonas. Considere la Figura 1 en donde se observa como las hormigas establecen el camino más corto. En el figura (a) las hormigas llegan al punto en que tienen que decidir por uno de los caminos que se les presenta; en (b) realizan la elección de manera aleatoria, algunas hormigas eligen el camino hacia arriba y otras hacia abajo; en (c)

[1] El presente trabajo fue parcialmente financiado por la Dirección de Investigaciones, Postgrado y Relaciones Internacionales – DIPRI de la Universidad Nacional de Asunción.

como las hormigas se mueven aproximadamente a una velocidad constante, las que eligieron el camino más corto alcanzarán el otro extremo más rápido que las otras que tomaron el camino más largo, depositando mayor cantidad de feromona por unidad de longitud; en (d) la cantidad de feromona depositada en el trayecto más corto hace que la mayoría de las hormigas elijan este camino, por realimentación positiva.

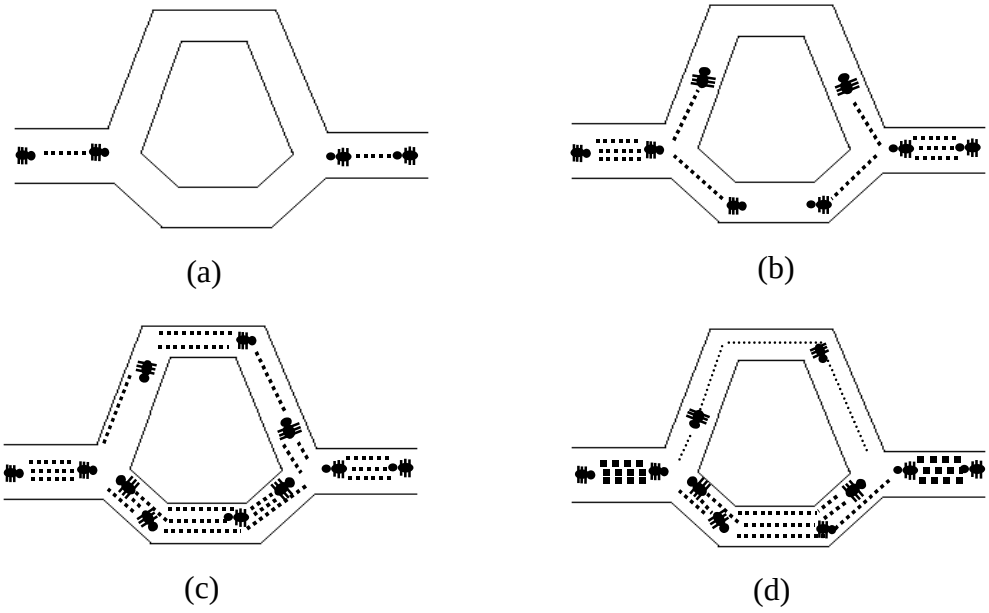

Figura 1: Comportamiento de las hormigas reales.

Esta innovadora técnica basada en agentes muy simples llamados hormigas, nació con la tesis doctoral de Marco Dorigo (1992) [Dor98] quien en 1996 publicó tres variantes del algoritmo para la resolución del problema del cajero viajante (*Traveling Salesman Problem - TSP* [DG97]) que se diferencian simplemente en el momento y la manera de actualizar una matriz de feromonas [DMC96]:

- *Ant-density*: con actualización constante de las feromonas por donde pasa una hormiga.
- *Ant-quantity*: con actualización de feromonas inversamente proporcional a la distancia entre 2 ciudades recorridas.
- *Ant-cycle*: con actualización de feromonas inversamente proporcional al trayecto completo, al terminar un recorrido. Este último presentó mejores resultados y las siguientes investigaciones se centraron en él.

Recientemente, Dorigo y Gambardella trabajaron en varias versiones extendidas del paradigma *AS*. Entre estas, *Ant-Q* es un híbrido entre *AS* y *Q-learning*, un conocido algoritmo de aprendizaje con realimentación positiva [GD95, DG96]. *Ant Colony System (ACS)* es una extensión de *Ant-Q* que presenta mejoras del algoritmo en tres aspectos principales [DG97]: i) una regla de transición de estados, con la que se ofrece un balance entre la exploración de nuevos caminos y explotación a priori del conocimiento acumulado acerca del problema; ii) una regla de actualización global que permite actualizar la matriz de feromonas solo con el mejor tour encontrado hasta el momento; iii) una regla de actualización local que permite a todas las hormigas actualizar la matriz de feromonas al terminar su tour. En dicha publicación se aplica además búsqueda local, utilizando el conocido método 3-opt

[JM97] para el problema del cajero viajante, que intenta reducir la longitud encontrada intercambiando los arcos, en particular, realizando tres cortes en un tour encontrado e intercambiando las ciudades destino, evitando de este modo invertir el sentido de las ciudades visitadas.

Otra variante de *AS*, conocida como *Max-Min Ant System (MMAS)*, fue presentada por Tomas Stützle y Holger Hoos [SH96], permitiendo actualizar la matriz de feromonas, solo a la hormiga con el mejor tour. Esto acelera la convergencia, pero puede llevar a estancamientos en soluciones sub-óptimas. A fin de evitar convergencias prematuras, se propuso poner un límite máximo y mínimo dentro del cual puede variar la cantidad de feromonas [SH96]. La aplicación con búsqueda local mejora notoriamente los resultados experimentales [SH97].

Como *Ant System* es una clase de algoritmo distribuido que consiste en un conjunto de agentes cooperativos que intercambian información de manera indirecta, el paralelismo es inherente al funcionamiento del algoritmo, es decir, el comportamiento de una hormiga es independiente de todas las demás durante la misma iteración. Se propusieron varias estrategias de paralelización. Así, en [BKS97] se publican detalles de una implementación paralela síncrona y otra parcialmente asíncrona que se resumen a continuación:

a) En la implementación paralela síncrona, un proceso inicial (*master*) levanta a un conjunto de procesos hijos, uno para cada hormiga. Después de distribuir la información inicial acerca del problema, cada proceso inicia la construcción del camino y calcula la longitud del tour encontrado. Después de terminar este procedimiento, los resultados son enviados al master, quien se encarga de actualizar el nivel de feromonas y calcular el mejor tour encontrado hasta ese momento. Se inicia una nueva iteración con el envío de la nueva matriz de feromonas.

b) En la implementación parcialmente asíncrona, se propone reducir la frecuencia de la comunicación, para esto, cada hormiga realiza un cierto número de iteraciones del algoritmo secuencial, independientemente de las otras hormigas. Solo después de estas iteraciones locales, se realiza una sincronización global entre las hormigas. Entonces el master actualiza el nivel de feromonas y se inicia el proceso de nuevo.

Como una extensión de este trabajo, Tomas Stützle presenta otra estrategia de paralelización [Stü98], proponiendo corridas independientes y paralelas de un mismo algoritmo, *(ACO o MMAS)* evitando de éste modo el *overhead* de comunicación. El método funciona solo si el fundamento del algoritmo es aleatorio, como en el caso de Ant System. La mejor solución de *k* corridas es elegida al final.

Los autores del presente trabajo presentaron otra estrategia de paralelización [ACB99], proponiendo que un procesador master levante a los demás procesos, cada procesador hijo realiza la computación del problema para un cierto número de hormigas, obteniendo resultados parciales que serán transmitidos a los otros procesadores, colaborando todos en la solución del problema global. Los resultados fueron obtenidos simulando procesos paralelos en una workstation DEC 3000 con procesador ALPHA, operando bajo el sistema operativo OSF/1 versión 2.0.

Muchos investigadores interesados por la originalidad y el rendimiento de los Sistemas de Hormigas, aplicaron la técnica con excelentes resultados a problemas tan diversos como:

- el paradigma del cajero viajante *(Traveling Salesman Problem)* [DMC96, DG97];
- el problema del ordenamiento secuencial *(Sequential Ordering Problem)* [GD97];
- el problema de ruteo de vehículos *(Vehicle Routing Problem)* [BHS99];
- el problema de asignación cuadrática *(Quadratic Assignment Problem)* [MC99];
- redes de telecomunicaciones *(Telecommunications Networks)* [CD98].

Esta técnica comienza a tener la madurez tecnológica adecuada para su utilización en problemas reales, como puede verse por la publicación de libros como [BDT99] y [CDG99].

El presente trabajo propone resolver el problema simétrico del cajero viajante (TSP) en un ambiente totalmente asíncrono, con una red de computadoras personales de uso no exclusivo, utilizando una nueva propuesta paralela de *Ant System*. Esto es, dadas *N* ciudades y las distancias entre estas (idénticas en ambos sentidos), se desea encontrar el camino más corto que permita recorrer las *N* ciudades, regresando al punto de partida, sin pasar por una misma ciudad más de una vez. Para esto, agentes computacionales muy simples llamados *hormigas* trabajan en cada procesador de una red de computadoras, explorando el espacio de soluciones, comunicando en forma asíncrona a los demás procesadores los resultados más alentadores, consolidando la información recogida en *matrices de feromonas* propias de cada procesador, que servirán para guiar la búsqueda de mejores soluciones en cada uno de los procesadores de la red, sin necesidad de sincronizar los procesos.

El trabajo presenta en la próxima sección el algoritmo *Ant System*, dejando la estrategia de paralelización para la sección 3. Las variantes experimentadas son introducidas en la sección 4 mientras los resultados experimentales son presentados en la sección 5. Finalmente, se presentan las conclusiones en la sección 6.

2. Ant System

Inspirados en el comportamiento de las hormigas, arriba descripto, Dorigo et al. [DMC96] proponen el algoritmo *Ant System (AS)*, presentado a continuación.

Para el presente trabajo, se considera un conjunto de *MAXC* ciudades que deben ser visitadas una sola vez con el objeto de encontrar la longitud mínima de recorrido y se define $b_i(t)$ *(i=1, ...,MAXC)* como el número de hormigas en la ciudad *i* al tiempo *t*. Por consiguiente, el número total de hormigas *MAXH* estará dado por:

$$MAXH = \sum_{i=1}^{MAXC} b_i(t) \qquad \textbf{(1)}$$

Para satisfacer la restricción de que una hormiga visite todas las ciudades una sola vez, se asocia a cada *hormiga k* una estructura de datos llamada lista tabú, $\mathbf{tabu}_k$, que guarda las ciudades ya visitadas por dicha hormiga. Una vez que todas las ciudades hayan sido recorridas, el trayecto o tour (ciclo) es completado, la lista tabú se vacía y nuevamente la hormiga está libre para iniciar un nuevo tour. Se define como $\mathbf{tabu}_k\mathbf{(s)}$ al elemento *s-esimo* de la lista tabú de la *hormiga k*.

El punto de partida para la solución del problema *TSP*, es la matriz de distancias $D = \{d_{ij}$, distancia entre las ciudades $i,j\}$, a partir de la cual se calcula la visibilidad $\eta_{ij} = 1/d_{ij}$. Por su parte, se denota como $\tau = \{\tau_{ij}\}$ a la matriz de feromonas a ser utilizada para consolidar la información que va siendo recogida por las hormigas; en otras palabras, la cantidad de feromona que se va almacenando entre cada par de ciudades (*i, j*).

$\tau_{ij}(t)$ especifica la intensidad de las feromonas del arco (*i, j*) en el tiempo *t*, y se actualiza según:

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

donde ρ es el coeficiente de persistencia de las feromonas, de forma tal que (1-ρ) representa la evaporación de la substancia entre *t* y *t*+1, mientras que la cantidad de feromona depositada en un arco (*i, j*), en dicho intervalo de tiempo, está dada por:

$$\Delta\tau_{ij}(t, t+1) = \sum_{k=1}^{MAXH} \Delta\tau_{ij}^{k}(t, t+1) \qquad \textbf{(3)}$$

con $\Delta\tau_{ij}^{k}(t, t+1)$ representando la cantidad de feromona depositada en el arco (*i, j*) por la hormiga *k-esima* entre *t* y *t+1*.

Durante la ejecución del algoritmo *Ant System*, cada *hormiga* elige en forma probabilística la próxima ciudad a visitar, realizando un cálculo de probabilidad que es función de la distancia y la cantidad de feromona depositada en el arco que une a las ciudades origen-destino, esto es:

$$p_{ij}(t) = \begin{cases} \dfrac{\left[\tau_{ij}(t)\right]^{\alpha} \times \left[\eta_{ij}\right]^{\beta}}{\sum_{j \notin Tabu_k} \left[\tau_{ij}(t)\right]^{\alpha} \times \left[\eta_{ij}\right]^{\beta}} & \text{si } j \notin \text{Tabu}_k \\ 0 & \text{de otra manera} \end{cases} \qquad \textbf{(4)}$$

donde α y β son constantes que expresan la importancia relativa del sendero de feromonas y la distancia entre las ciudades respectivamente. Así, un alto valor de α significa que el sendero de feromonas es muy importante y que las hormigas tienden a elegir caminos por los cuales otras hormigas ya pasaron. Si por el contrario, el valor de β es muy alto, una hormiga tiende a elegir la ciudad más cercana.

En el instante *t*, las *hormigas* se mueven de una ciudad a la siguiente (movimiento llamado: iteración), en donde se encontrarán en el instante *t*+1. Lógicamente, al cabo de (MAX*C* – 1) iteraciones, las hormigas han visitado la última ciudad y están en condiciones de regresar a su ciudad origen, posiblemente para actualizar la matriz de feromonas con la información recogida en el tour.

El proceso se repite iterativamente hasta que se cumpla algún criterio de parada. En este trabajo, el proceso termina si el contador de tour alcanza un número máximo de ciclos *NCMAX* (definido por el

usuario) o todas las hormigas realizan el mismo tour. En este último caso, es evidente que las hormigas han dejado de buscar nuevas soluciones, lo que constituye un criterio de convergencia del algoritmo (similar a la uniformización de la población de un algoritmo genético [BCC98]).

La cantidad de feromonas depositada en el trayecto es proporcional a la distancia del tour completo encontrado y por lo tanto, es de esperar una apreciable capacidad de búsqueda de soluciones globales. Para esto, se realiza el siguiente cálculo de $\Delta\tau_{i,j}^{k}$:

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{L_k} & \text{si la hormiga } k\text{-}esima \text{ camina por el arco } (i, j) \\ 0 & \text{de otra manera} \end{cases} \qquad \textbf{(5)}$$

donde Q es una constante y L_k es la longitud del tour completo realizado por la hormiga k.

En el Pseudocódigo 1 se presenta el algoritmo secuencial de *Ant System* en su versión más utilizada [CDM92].

3. Paralelismo Asíncrono del *Ant System*

La complejidad computacional del algoritmo secuencial impide su aplicación a problemas de gran envergadura. Por otra parte, *Ant System* tiene características que lo hacen especialmente apropiado para su paralelización y distribución entre los diversos procesadores de una red de computadoras en un contexto asíncrono. En efecto, el método utiliza la interacción de muchos agentes relativamente simples llamados *hormigas*, pero básicamente independientes entre sí en cada tour, que cooperan intercambiando información para el logro de un objetivo común.

Cada hormiga va construyendo su propio tour, con la única restricción de no viajar a una ciudad ya visitada con anterioridad. En consecuencia, el paralelismo está implícito en el mismo algoritmo. Consecuentemente, el presente trabajo propone que cada procesador realice la computación del problema para un cierto número de hormigas, obteniendo resultados parciales que podrán ser transmitidos a los otros procesadores en forma asíncrona, colaborando todos en la solución del problema global, pero sin necesidad de perder tiempo en sincronizar los procesadores, dado que la nueva información (útil para actualizar la matriz de feromonas) solo se usa si está disponible.

Con esta propuesta, el asincronismo elimina los tiempos muertos producidos por la espera en la sincronización de la comunicación, extremadamente perjudiciales cuando se trabaja con una red de computadoras cuyo tráfico no se puede controlar totalmente. En consecuencia, el presente trabajo propone que las hormigas migren de un procesador a otro en forma asíncrona, respetando políticas migratorias similares a las utilizadas con los Algoritmos Genéticos Paralelos [BCC98]; en otras palabras, políticas definidas por los siguientes parámetros:

- *El intervalo de migración*: que establece cada cuantos ciclos cierta cantidad de hormigas migrarán de un procesador a otro.

- *La tasa de migración*: que indica cuantas hormigas han de comunicarse al otro procesador cuando se cumpla el intervalo de migración.

- *El criterio de selección*: que determina la política a seguir para la selección de las hormigas que han de migrar. Por ejemplo que sean elegidas al azar. Sin embargo, con el fin de ayudar a los demás procesos, se elegirán las hormigas que hayan obtenido las mejores soluciones (tour con menores distancias).

1. Fase de inicialización
   Inicializar contador de ciclos ***NC***
   Para cada arco ***(i.j)***:
   valor inicial de $\tau_{ij}(t) = c$ /**c*= constante positiva pequeña*/
   $\Delta\tau(ij) = 0$
   Colocar ***MAXH*** hormigas en ***N*** ciudades

2. Colocar la ciudad origen en lista $\boldsymbol{tabu}_k$ de cada hormiga

3. Repetir hasta llenar $\boldsymbol{tabu}_k$
   Para cada hormiga:
   Elegir próxima ciudad a ser visitada según ecuación (4)
   Mover la hormiga a la próxima ciudad
   Insertar la ciudad en $\boldsymbol{tabu}_k$

4. Repetir para cada hormiga ***k***
   Regresar a la ciudad origen
   Calcular la longitud $\boldsymbol{L_k}$ del ciclo
   Guardar el camino más corto hasta ciclo **NC**: $L_0^{NC} = min\{L_0^{NC-1}; min_k\{L_k\}\}$
   Para cada arco ***(i,j)***
   Calcular $\Delta\tau_{ij}$ según ecuación (3)

5. Para cada arco ***(i,j)***
   Actualizar τ_{ij} según ecuación (2)
   $\Delta\tau_{ij} = 0$

6. Aumentar contador de ciclos ***NC***
   Si ***NC*** < $\boldsymbol{Nc_{max}}$
   Vaciar $\boldsymbol{tabu}_k$
   Ir a la fase 2
   Sino
   Imprimir camino más corto L_0^{NC}
   Fin

Pseudocódigo 1: Ant System secuencial de Dorigo et al.[DMC96].

Cuando cada proceso recibe a las hormigas migrantes, éstas se ubican en sus ciudades de origen, donde se aplica un criterio de selección similar al utilizado con los Algoritmos Genéticos [Gol89] a fin de mantener constante el número original de hormigas por cada ciudad. En otras palabras, las hormigas que hayan encontrado mejores soluciones, tienen mayor probabilidad de sobrevivir, simulando la selección natural en el que el más fuerte sobrevive [BCC98].

La manera más fácil de entender al operador de selección consiste en imaginar una ruleta, en la que el número máximo de partes en que se divide la ruleta es igual al número de hormigas en la ciudad, y el tamaño de cada parte es inversamente proporcional a la distancia encontrada por cada una de ellas. Esto se debe a que buscamos minimizar la distancia. Una vez seleccionadas las hormigas que sobrevivirán al siguiente tour, utilizando la ruleta, se actualiza τ_{ij} del procesador en cuestión. Con esto, un buen resultado influye no solo en el procesador que encontró dicho resultado, sino que tiene mayor probabilidad de actualizar inclusive la matriz de feromonas del procesador que la recibió.

A continuación se presenta la versión paralela implementada que utiliza un proceso *Master* que se encarga de administrar la implementación paralela (Pseudocódigo 2) incluyendo el lanzamiento de los procesos *Esclavos* (Pseudocódigo 3), para que éstos realicen los cálculos propiamente dichos.

1. Fase de inicialización
   Levantar procesos
   Enviar parámetros a cada esclavo
   Fin= Falso
2. Repetir mientras no sea ***Fin***
   Recibir solución de los esclavos
   Guardar mejor solución
   Si todos los procesos terminaron
   Fin= Verdadero
3. Eliminar procesos esclavos

Pseudocódigo 2: Proceso *Master*.

1. Fase de inicialización
   Recibir parámetros
   Inicializar variables
2. Repetir mientras (***NC*** < NC_{max} y todas las hormigas no realicen el mismo tour)
   Mover hormigas hasta completar tour
   Calcular distancia recorrida para cada hormiga
   Escoger hormigas migrantes
   Enviar hormigas migrantes a otros procesos
   Recibir hormigas migrantes de otros procesos
   Ubicar migrantes en su ciudad origen
   Seleccionar las hormigas que sobrevivirán
   Actualizar matriz de feromonas
   Guardar camino más corto
3. Enviar mejor solución al master.

Pseudocódigo 3: Proceso *Esclavo.*

4. Variantes implementadas

La Tabla 1 presenta 4 variantes de las estrategias de paralelización que han sido implementadas, entre otras alternativas. Básicamente, se realizaron experiencias modificando la manera en que se eligen las hormigas que viajarán a los demás procesadores (solo las mejores o distintas para cada procesador) y que hormigas serán las que actualicen la matriz de feromonas (todas, o solo las mejores).

Migrantes / Actualizan	Diferenciados por procesador	Idénticos para todos los procesadores
Todas	AS1	AS2
Las mejores	AS3	AS4

Tabla 1: Versiones implementadas.

En las versiones con migrantes diferenciados por procesador, se eligen las hormigas que encontraron los mejores resultados para transmitirlas a los demás procesadores, en un número preestablecido, escogiendo al azar que hormigas migrarán. Claramente, cada procesador recibe hormigas diferentes con la esperanza de realizar búsquedas en espacios no necesariamente similares. La segunda alternativa es enviar a las mejores hormigas a todos los demás procesadores, paralelizando la tradicional búsqueda secuencial.

La actualización de la matriz de feromonas de un procesador puede ser realizada por la totalidad de las hormigas disponibles en un procesador al momento de la actualización, o sólo con las mejores *k* hormigas. Resultados experimentales fueron obtenidos con *k* igual al 10, 20, 30 y 60% del total de hormigas, que en las experiencias realizadas, coincide con el total de ciudades.

La plataforma computacional utilizada en los experimentos está formada por una red local *Ethernet* de computadoras personales con procesadores AMD-K6 de 350 MHz, 128 MB de memoria RAM, corriendo bajo sistema operativo Linux Red Hat 6.0. Las implementaciones fueron realizadas en lenguaje C utilizando librerías PVM (*Parallel Virtual Machine*).

Nro. de Procesadores	2	4	6
Versión AS1			
Mejor solución	425.649	426.544	425.649
Solución promedio	430.142	429.088	427.217
Desviación estándar	1.632	2.112	1.901
Versión AS2			
Mejor solución	425.820	425.820	425.820
Solución promedio	428.059	427.960	427.543
Desviación estándar	2.216	1.922	1.795

Tabla 2: Comparación de versiones AS1 y AS2 para el problema Oliver30.

Para la comparación de las diversas versiones se realizaron experiencias, resolviendo el problema Oliver30 [WSF89] para 30 ciudades, con diversos números de computadoras, como lo ilustra la Tabla 2 que compara las versiones AS1 y AS2, donde se presentan los mejores valores obtenidos, valores promedios y desviación estándar para 10 corridas independientes por experimento. En todos los casos se corrieron un mismo número de ciclos (5000), por lo que las versiones paralelas con *p* procesadores corrieron un número proporcionalmente menor de ciclos (5000/*p*) por procesador, de forma a mantener constante el número total de ciclos. Como puede apreciarse, la solución promedio va mejorando en calidad en la medida que crece el número de procesadores.

La Tabla 3 compara los resultados obtenidos con las versiones AS3 y AS4, utilizando 5 procesadores, para resolver el problema Oliver30. Esta tabla permite observar que no justifica actualizar la matriz de feromonas con más de 10 a 20 % del número total de hormigas, dado que solo se incrementa el esfuerzo computacional sin ninguna mejora apreciable (de hecho empeoran los resultados con el número de hormigas). En consecuencia, se sugiere utilizar *k*= 10% del total de las hormigas, dado que valores menores no conducen a una actualización suficiente de la matriz de feromonas.

Nro de Hormigas que actualizan feromonas	1	3	6	9	18
Versión AS3					
Mejor solución	424.692	**423.741**	**423.741**	423.912	424.635
Solución promedio	433.236	427.104	426.314	425.954	427.188
Desviación estándar	7.731	4.338	4.428	2.286	3.193
Versión AS4					
Mejor solución	**423.741**	424.692	423.912	423.912	425.649
Solución promedio	435.916	427.771	425.892	425.776	426.930
Desviación estándar	9.201	4.740	2.080	1.898	1.095

Tabla 3: Comparación de versiones AS3 y AS4 para el problema Oliver30.

Puede notarse en la Tabla 3 que la versión AS3 encuentra 2 veces la solución óptima, por lo que se la puede considerar ligeramente superior a la versión AS4. Dada la superioridad de la versión AS3, esta fue modificada permitiendo evaporación sólo en los caminos por donde las hormigas han pasado en la iteración actual. Esta nueva versión, denotada como AS3.1, presenta un mejoramiento adicional para el problema Oliver30. En consecuencia, se realizaron experimentos con el problema Eilon50 [WSF89], en cuyo caso se encuentra la solución óptima sólo cuando el 10% de las hormigas actualizan la matriz de feromonas, corroborando lo arriba recomendado.

En conclusión, de todas las versiones presentadas, la AS3.1 es la que encontró los mejores resultados experimentales.

Versión AS3.1 – Problema de 30 ciudades					
Nro de Hormigas que actualizan feromonas	1	3	6	9	18
Mejor solución	456.525	**423.741**	**423.741**	**423.741**	**423.741**
Solución promedio	480.557	423.847	423.809	423.758	423.758
Desviación estándar	21.222	0.216	0.084	0.051	0.051
Versión AS3.1 – Problema de 50 ciudades					
Nro de Hormigas que actualizan feromonas	1	5	10	15	30
Mejor solución	723.491	**427.855**	428.721	429.778	436.956
Solución promedio	745.941	429.817	431.812	434.680	444.917
Desviación estándar	12.422	2.188	2.340	2.579	5.268

Tabla 4: Comparación de la versión AS3.1 para los problemas Oliver30 y Eilon50.

5. Aceleración y Eficiencia de las Implementaciones

Como ejemplo de los resultados experimentales obtenidos utilizando la red de computadoras personales no dedicadas arriba descripta, la Figura 2 muestra la aceleración promedio en 10 corridas utilizando 2, 4 y 6 procesadores, en este ejemplo, para la versión AS2. Estos resultados se mantienen muy similares para todas las versiones. Claramente, existe una buena escalabilidad con el número de procesadores.

En lo relativo a la eficiencia, la Figura 3 demuestra que la paralelización del algoritmo es sumamente benéfica y sencilla, logrando eficiencias superiores al 100% en los problemas más grandes. En consecuencia, es de esperar que la presente propuesta pueda ser utilizada con un número creciente de procesadores sin mayores complicaciones.

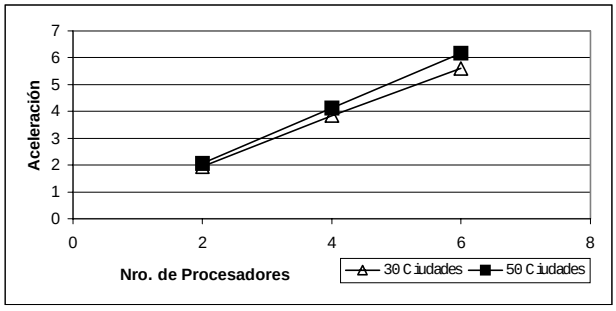

Figura 2: Aceleración experimental con versión AS2

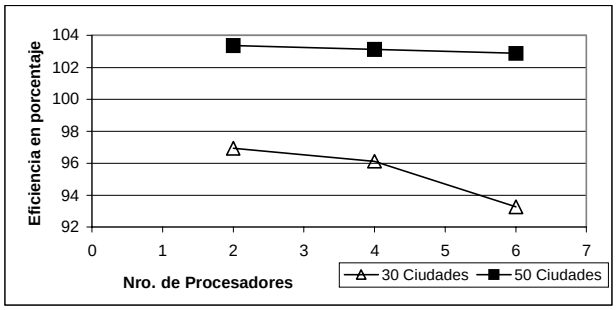

Figura 3: Eficiencia experimental con versión AS2

6. Conclusiones

Esta novedosa técnica basada en el comportamiento de las colonias de hormigas ha sido aplicada a diversos problemas con excelentes resultados, demostrando ser una buena opción para resolver problemas de optimización combinatoria, por su facilidad de paralelización y sus excelentes resultados computacionales.

El presente trabajo presentó diversas alternativas de paralelización asíncrona del algoritmo *Ant System*, analizando diversas políticas de migración de hormigas y actualización de la matriz de feromonas, en el contexto asíncrono de una red de computadoras. Resultados experimentales demuestran una buena eficiencia, que eventualmente superó el 100% para los problemas mayores, consiguiéndose muy buena escalabilidad, lo que permite postular su utilización eficiente con un número creciente de procesadores y en problemas de mayor tamaño y complejidad, en lo que se está trabajando a la fecha.

Bibliografía

[ACB99] Almirón M., Chaparro E. y Barán B. "Sistema distribuido de hormigas para el problema del cajero viajante" *XXV Conf. Latinoam. de Informática* – CLEI'99, Paraguay, 1999.

[BCC98] Barán B., Chaparro E. y Cáceres N., "A-Teams en la Optimización del Caudal Turbinado de una Represa Hidroeléctrica", *Conf. Iberoam. Intelig. Arti.l IBERAMIA'98*, Portugal, 1998.

[BDT99] Bonabeau E., Dorigo M. y Theraulaz T., *From Natural to Artificial Swarm Intelligence*. New York: Oxford University Press, 1999.

[BHS99] Bullnheimer B., Hartl R. y Strauss C. "An improved ant system algorithm for the vehicle routing problem". *Annals of Operations Research* (Dawind, Feichtinger and Hartl (eds.): Nonlinear Economic Dynamics and Control, 1999.

[BKS97] Bullnheimer B., Kotsis G. y Strauss C. "Parallelization Strategies for the Ant System", Reporte Técnico POM 9/97, Universidad de Viena, Viena – Austria, 1997.

[CB97] Chaparro E. y Barán B., "Algoritmos Asíncronos Combinados en un Ambiente Heterogéneo de Red*", XXIII Conf. Latinoam. de Informática (CLEI)*, Chile, 1997.

[CD98] Di Caro G. y Dorigo M., "Mobile Agents for Adaptive Routing", *Proceedings of the 31st Hawaii International Conference on System*, Hawaii, 1998.

[CDG99] Corne D., Dorigo M. y Glover F., *New Ideas in Optimization*. McGraw-Hill. 1999.

[DG96] Dorigo M. y Gambardella L., "A study of some properties of Ant-Q". *IV Int. Conf. on Parallel Problem from Nature*, Berlin – Alemania: Springer-Verlag, pp. 656-665, 1996.

[DG97] Dorigo M. y Gambardella L., "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem". *IEEE Trans. on Evolut. Comp.*, Vol 1, No. 1, pp. 53-66, 1997.

[DMC96] Dorigo M., Maniezzo V. y Colorni A., "The Ant System: Optimization by a colony of cooperating agents", *IEEE Trans. on Systems, Man, & Cybernetics*, Vol 26, No. 1, pp. 1-13, 1996.

[Dor98] Dorigo M., *Ant Colony System*, URL: http://iridia.ulb.ac.be/dorigo/ACO/ACO.html.

[GD95] Gambardella L. y Dorigo M., "Ant-Q: A reinforcement learning approach to the traveling salesman problem" *12th Int. Conf. on Machine Learning*. San Francisco, pp. 252-260, 1995.

[GD97] Gambardella L. y Dorigo M., "HAS-SOP: An Hybrid Ant System for the Sequential Ordering Problem", Reporte Técnico *IDSIA11-97*, Lugano-Suiza, 1997.

[Gol89] Golberg, D.E., *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison – Wesley, Reading, MA, 1989.

[JM97] Johnson D.S. y McGeoch L.A., "The traveling salesman problem: a case study in local optimization", *Local Search in Comb. Optimization*, Eds. New York: Wiley: New York, 1997.

[MC99] Maniezzo V. y Colorni A. "The Ant System applied to the Quadratic Assignment Problem". A ser publicado en la IEEE Transactions on Knowdledge and Data Engineering, 1999.

[SH96] Stützle T. y Hoos H., "Improvements on the Ant System: Introducing Max-Min Ant System". *Int. Conf. on Artif. Neural Networks and Genetic Alg.*, Viena – Austria, 1997.

[SH97] Stützle T. y Hoos H., "Max-Min Ant System and Local Search for Combinatorial Optimization Problems". *2nd Int. Conf. on Metaheuristics* – MIC97, Francia, 1997.

[Stü98] Stützle T., "Parallelization Strategies for Ant Colony Optimization", *Proc. of Parallel Problem Solving from Nature* – PPSN-V, Springer Verlag, Vol. 1498, pp. 722-731, 1998.

[WSF89] Whitley D., Starkweather T. y Fuquay D., "Scheduling Problems and Travelling Salesman: the Genetic Edge Recombination Operator", *Proc. of the 3rd Int. Conf. on Genetic Alg.*.