SISTEMA DISTRIBUIDO DE HORMIGAS PARA EL PROBLEMA DEL CAJERO VIAJANTE[1]

Marta Almirón
malmiron@cnc.una.py

Enrique Chaparro Viveros
eviveros@cnc.una.py

Benjamín Barán
bbaran@cnc.una.py

Centro Nacional de Computación
Universidad Nacional de Asunción
Casilla de Correos 1439
Campus Universitario de San Lorenzo - Paraguay
Tel./Fax: (595)(21) 585 619 y 585 550
http://www.cnc.una.py

Resumen

La metáfora natural de una colonia de hormigas ha sido utilizada para definir un Sistema de Hormigas (*Ant System)*, una familia de algoritmos distribuidos para optimización combinatoria. Simples agentes computacionales, llamados *hormigas,* intercambian información para encontrar buenas soluciones, en este trabajo, para el conocido paradigma del Cajero Viajante (TSP – *Traveler Salesman Problem*).

Al moverse, una hormiga real deposita una substancia denominada feromona, marcando el camino que fue recorrido. Inspirado en esta metáfora, *Ant System* utiliza una matriz de feromonas para establecer comunicación indirecta entre las diversas *hormigas,* facilitando la búsqueda de buenas soluciones.

El presente trabajo propone ofrecer a las hormigas un conocimiento previo de las características del problema, escalando la matriz de feromonas a partir de la matriz de distancias para el algoritmo conocido como *Ant Quantity* y disminuir considerablemente los tiempos de ejecución al paralelizar el algoritmo *Ant Cycle,* en un ambiente típicamente asíncrono.

Palabras Claves

Sistemas distribuidos y paralelismo, inteligencia artificial, sistemas evolutivos, *Ant System,* Hormiga, Feromonas, TSP.

1. INTRODUCCIÓN

Ant System, propuesto originalmente por Dorigo et al. [1-2], 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 casi 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 un rastro odorífero, depositando una substancia denominada feromona.

Las feromonas son un sistema de comunicación química entre animales de la misma especie [3], que transmiten informaciones esenciales para la supervivencia de la especie como ser el estado fisiológico, reproductivo y social, así como la edad, sexo y parentesco del animal emisor. Dichas señales son recibidas por el sistema olfatorio del animal receptor, quien las interpreta para decidir sobre su comportamiento posterior.

[1] Este trabajo ha sido parcialmente desarrollado con los auspicios de la Dirección de Investigaciones, Postgrado y Relaciones Internacionales (DIPRI) de la Universidad Nacional de Asunción (UNA).

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. Como ejemplo, considere la Figura 1.a donde las hormigas han establecido un camino desde su nido hasta su fuente de alimento, y asumamos que repentinamente una piedra corta el camino (Figura 1.b), la mitad de las hormigas se dirigirán hacia el extremo A y la otra mitad hacia el extremo B. Al ser menor la distancia para rodear el extremo B, mayor cantidad de hormigas pasaran por este camino y por lo tanto mayor cantidad de feromonas será depositada en dicho trayecto (Figura 1.c); en consecuencia, las hormigas influenciadas por el sendero de feromonas elegirán rodear el obstáculo por el extremo B (Figura 1.d), lo que a su vez servirá para depositar más feromonas en dicho trayecto, que por realimentación positiva, será seleccionada como el camino más corto.

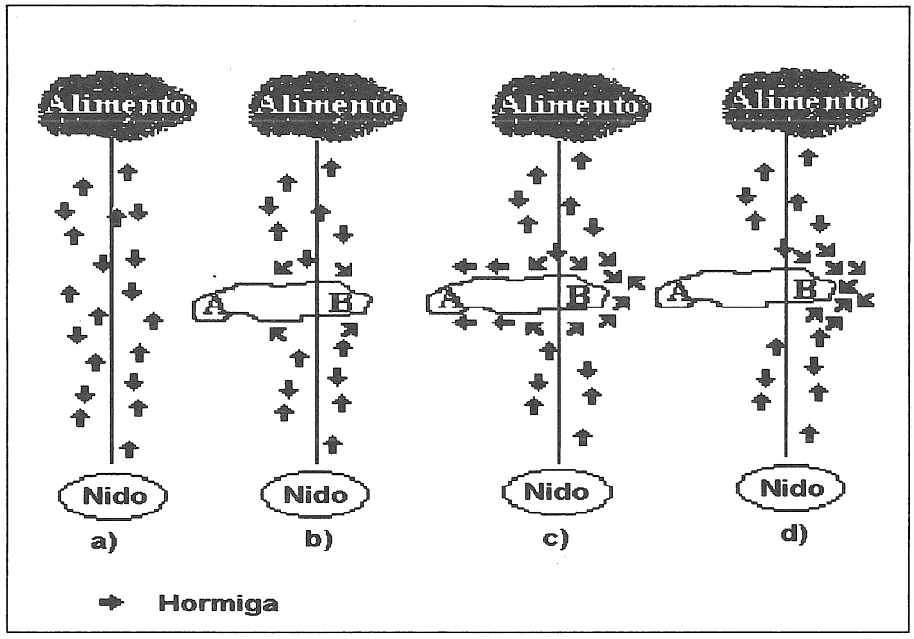

Figura 1: Comportamiento de las hormigas reales en la búsqueda de alimentos.

El presente trabajo propone resolver el problema simétrico del cajero viajante (TSP - *Traveler Salesman Problem*) utilizando una familia de algoritmos, conocida como *Ant System*, inspirado en la metáfora natural arriba explicada. 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 llamados *hormigas* irán explorando el espacio de soluciones, consolidando la información recogida en una *matriz de feromonas*, que servirá para guiar la búsqueda de mejores soluciones. El trabajo presenta en la próxima sección al *Ant System*, proponiendo mejoras al algoritmo *Ant Quantity* en la sección 3 y la paralelización del algoritmo *Ant Cycle* en la sección 4. Resultados experimentales de una implementación paralela son presentados en la sección 5, mientras las conclusiones se dejan para la sección 6.

2. ANT SYSTEM

Inspirados en el comportamiento de las hormigas, arriba descripto, Dorigo et al. [1, 2, 4, 5] proponen tres variantes del algoritmo Ant System (AS), que básicamente difieren en el momento y la modalidad en que actualizan la matriz de feromonas que sirve para guiar el movimiento de los agentes computacionales llamados hormigas; estos algoritmos son:

- *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.

Para el presente trabajo, se considera un conjunto de *MAXC* ciudades y se define $b_i(t)$ como el número de hormigas en la ciudad *i* en el tiempo *t*, por consiguiente, el número total de hormigas *MAXH* estará dado por:

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

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 [6], $\mathbf{tabu_k}$, 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 $\mathbf{tabu_k(s)}$ al elemento *s-esimo* de la lista tabú de la *hormiga k*.

El punto de partida para la solución del problema en cuestión, es la matriz de distancias D = $\{d_{ij}$ - distancia entre la ciudad *i* y la ciudad *j*$\}$, a partir de la cual se calcula la visibilidad $V_{ij} = 1/d_{ij}$. Por su parte, se denota como T = $\{T_{ij}\}$ 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*.

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 feromonas depositada en el arco que une a las ciudades origen-destino, esto es:

$$p_{ij}(t) = \begin{cases} \dfrac{\left[T_{ij}(t)\right]^{\alpha} \times \left[V_{ij}\right]^{\beta}}{\sum_{j \notin Tabu_k} \left[T_{ij}(t)\right]^{\alpha} \times \left[V_{ij}\right]^{\beta}} & \text{si } j \notin \text{Tabu}_k \\ 0 & \text{de otra manera} \end{cases} \qquad (1)$$

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. Se resalta aquí que cada *hormiga* debe realizar un tour legal o sea, no puede viajar a una ciudad ya visitada con anterioridad hasta que complete su tour.

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 completo.

La matriz $T_{ij}(t)$ que especifica la intensidad de las feromonas del arco (i, j) en *t*, se actualiza según la fórmula:

$$T_{ij}(t+1) = \rho \times T_{ij}(t) + AT_{ij}(t, t+1) \qquad (2)$$

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), está dada por:

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

con AT_{ij}^{k} (*t, t+1*) representando la cantidad de feromonas por unidad de longitud, depositada en el arco (i, j) por la hormiga *k-esima* entre *t* y *t+1*. Cabe destacar aquí que la principal diferencia entre las diversas variantes del algoritmo *Ant System*, está dada justamente en la forma de calcular AT_{ij}^{k} (*t, t+1*), conforme será explicado en las próximas subsecciones.

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).

1.1 Ant Quantity

En esta variante del algoritmo (ver Pseudocódigo 1), la hormiga k que viaja desde la ciudad i a la ciudad j, deposita en el trayecto una cantidad de feromonas inversamente proporcional a la distancia d_{ij}, esto es:

$$AT_{ij}^{k}(t,t+1)=\begin{cases}\dfrac{Q_1}{d_{ij}} & \text{si la hormiga } k\text{ - esima camina por el arco } (i,j) \text{ entre } t \text{ y } t+1\\ 0 & \text{de otra manera}\end{cases} \qquad (4)$$

donde Q_1 es una constante.

```
t=0;                                          /*t es contador de tiempo*/
nc=0;                                         /*nc es el contador de ciclos*/
s=1;                                          /*índice de la lista tabú*/
Para cada arco (i,j) inicializar T_ij(t) = c;  /*c es una constante positiva pequeña*/
Para cada arco (i,j) AT_ij(t) = 0;
Colocar las MAXH hormigas en las MAXC ciudades;
Colocar la ciudad origen de la hormiga k-esima en tabu_k(s)

DO WHILE (nc<=NCMAX y que todas las hormigas no realicen el mismo tour)
        FOR t=1 hasta MAXC-1                  /*se repite hasta que la lista tabú este llena*/
                s=s+1
                FOR i=1 hasta i<=MAXC
                        FOR k=1 hasta k<=b(t)
                                Elegir la ciudad j a mover, con probabilidad p_ij(t) dado por la ecuación (1);
                                Mover la hormiga k-esima a la ciudad;
                                Insertar la ciudad j en tabu_k(s);
                                Calcular AT_ij(t,t+1) = AT_ij(t,t+1) + Q_1 / d_ij;
                        END FOR
                END FOR
                Para cada arco (i,j) calcular T_ij(t,t+1) según la ecuación (2);
        END FOR
        Guardar el camino más corto;
        nc=nc+1;
        IF (nc<=NCMAX)
                Vaciar todas las listas tabú;
                Para cada arco (i,j) AT_ij(t+1) = 0;
                s=1;
                Colocar la ciudad origen de la hormiga k-esima en tabu_k(s);
        END IF
END WHILE
Imprimir el mejor camino;
```

Pseudocódigo 1: *Ant-quantity* secuencial.

2.2 Ant Cycle

A diferencia del *Ant Quantity*, en esta variante (ver Pseudocódigo 2), la cantidad de feromonas depositada en el trayecto es proporcional a la distancia del tour completo encontrado y por lo tanto, es de esperar un rendimiento superior en la capacidad de búsqueda de soluciones globales. Esta variante realiza el siguiente cálculo:

$$AT_{ij}^{k} = \begin{cases} \dfrac{Q_2}{L_k} & \text{si la hormiga } k\text{ - esima camina por el arco } (i,j) \\ 0 & \text{de otra manera} \end{cases} \qquad (5)$$

donde Q_2 es una constante y L_k es la longitud del tour completo realizado por la hormiga k.

```
t=0;                                                    /*t es contador de tiempo*/
nc=0;                                                   /*nc es el contador de ciclos*/
s=1;                                                    /*índice de la lista tabú*/
Para cada arco (i,j) inicializar T_ij(t) = c;           /*c es una constante positiva pequeña*/
Para cada arco (i,j) AT_ij(t) = 0;
Colocar las MAXH hormigas en las MAXC ciudades;
Colocar la ciudad origen de la hormiga k-esima en tabu_k(s)

DO WHILE (nc<=NCMAX y que todas las hormigas no realicen el mismo tour)
        FOR t=1 hasta MAXC-1                            /*se repite hasta que la lista tabú este llena*/
                s=s+1
                FOR i=1 hasta i<=MAXC
                        FOR k=1 hasta k<=b(t)
                                Elegir la ciudad j a mover, con probabilidad p_ij(t) dado por la ecuación (1);
                                Mover la hormiga k-esima a la ciudad;
                                Insertar la ciudad j en tabu_k(s);
                        END FOR
                END FOR
        END FOR
        Calcular L^k;                                   /*distancia recorrida por cada hormiga*/
        Calcular AT_ij(t,t+1) = AT_ij(t,t+1) + Q_2 / L^k
        Para cada arco (i,j) calcular T_ij(t,t+1) según la ecuación (2);
        Guardar el camino más corto;
        nc=nc+1;
        IF (nc<=NCMAX)
                Vaciar todas las listas tabú;
                Para cada arco (i,j) AT_ij(t) = 0;
                s=1;
                Colocar la ciudad origen de la hormiga k-esima en tabu_k(s);
        END IF
END WHILE
Imprimir el mejor camino;
```

Pseudocódigo 2: *Ant-cycle* (M. Dorigo et al.)

3. PROPUESTA DE MEJORAS PARA EL ALGORITMO ANT QUANTITY

Dado que la matriz de feromonas se actualiza con una cantidad de feromonas proporcional a la distancia de cada arco, la presente sección propone inicializar la matriz de feromonas conforme:

$$T_{ij}(0) = f_{min} + \left[\frac{f_{max} - f_{min}}{d_{max} - d_{min}}\right] * \left(d_{max} - d_{ij}\right) \qquad (6)$$

donde:

f_{max} es una constante definida como valor máximo de feromonas correspondiente a la menor distancia.
f_{min} es una constante definida como valor mínimo de feromonas correspondiente a la mayor distancia.
d_{max} mayor distancia de la matriz de datos d_{ij}.
d_{min} menor distancia de la matriz de datos d_{ij}.

Con esto, se privilegian los caminos más cortos en la probabilidad de que una hormiga elija cierta ciudad, y como se verá a continuación, esta propuesta mejora notoriamente los resultados experimentales, dando a las hormigas un conocimiento previo del problema, al riesgo de poder perder alguna solución globalmente óptima, que utiliza caminos largos entre algunos pares de ciudades. Como se puede apreciar en la Figura 2, a una mayor distancia le corresponde una cantidad menor de feromonas en $t=0$.

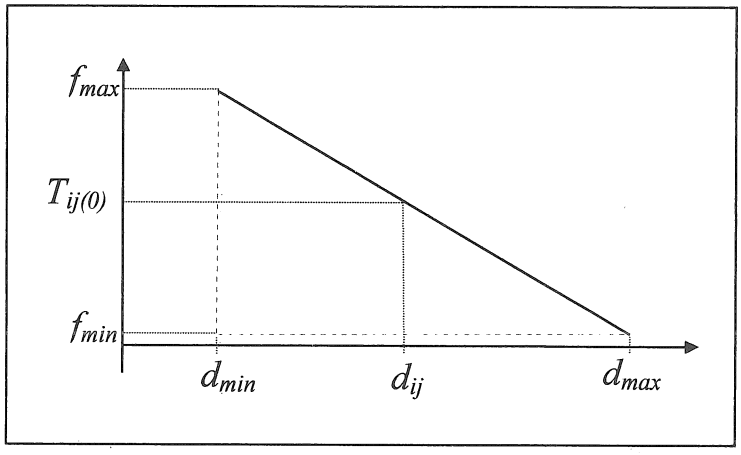

Figura 2: Escalamiento lineal de feromonas con respecto a la distancia para inicializar la matriz de feromonas.

Resultados experimentales

Para la implementación de los dos algoritmos Ant Quantity (con y sin escalamiento inicial de la matriz de feromonas), se utilizó como plataforma computacional una workstation DEC 3000 modelo 300 con procesador ALPHA de 150 MHz con 32 MB de memoria RAM, operando bajo sistema operativo OSF/1 versión 2.0.

Para comparar el desempeño de las dos versiones de Ant Quantity, se han realizado 10 corridas con cada posible combinación de los parámetros α, β y ρ mencionados a continuación:

$\alpha \in \{0.5; 1.0; 2.0\}$
$\beta \in \{2.0; 5.0; 10.0\}$
$\rho \in \{0.5; 0.9; 0.99\}$

La Figura 3 muestra los mejores promedios en 10 corridas (distancia del tour en función del tiempo de cómputo para encontrar esta solución). En dicha figura, se puede apreciar cuanto sigue:

a) La solución más rápida (0.1866 segundos) encuentra un tour de 2255.7 km y corresponde a la variante con inicialización, propuesto en este trabajo.
b) La solución óptima de 2195 km es encontrada con la variante aquí propuesta en 0.2383 segundos con α=2, β=10, ρ=0.9 y en 0.3299 segundos con α=1, β=5, ρ=0.9.
c) Por su parte, la variante original de Dorigo et al. (sin inicialización), solo encuentra la solución óptima en 0.8149 segundos.

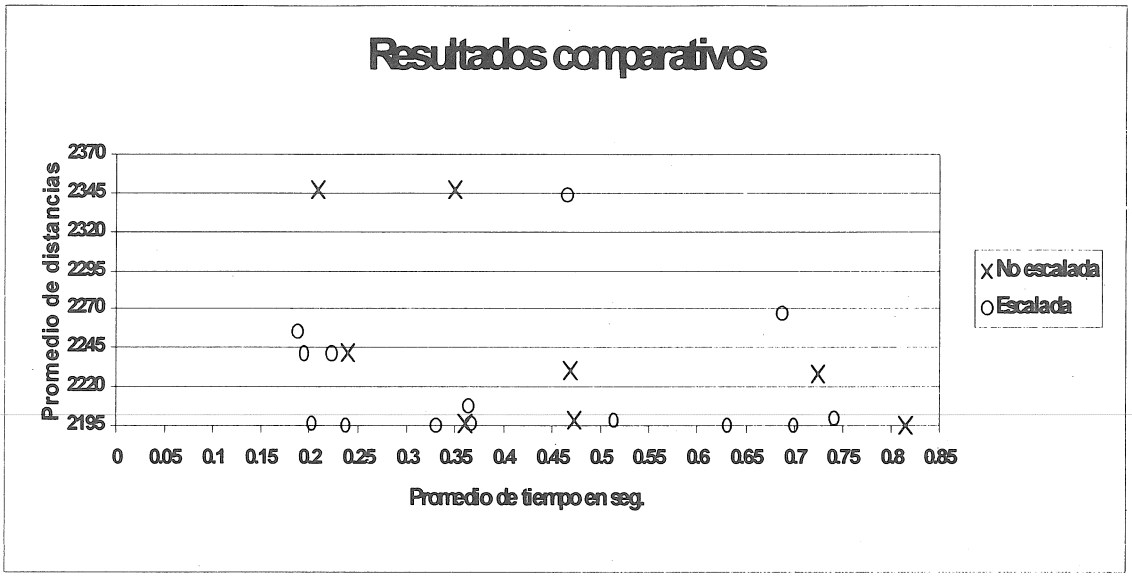

Figura 3: Gráfico con los mejores resultados promediados para 10 corridas sucesivas.

Los mejores promedios en tiempo y distancia están graficados en la Figura 4 formando una curva *paretto* (frontera óptima) con los mejores puntos (en el sentido de menor distancia y/o tiempo de cómputo). Se observa que el mejor resultado (0.2399 s - 2242.1 km) en tiempo y distancia obtenido por el algoritmo de Dorigo et al. no forma parte de la frontera óptima, que está totalmente definida por los resultados encontrados con la variante propuesta.

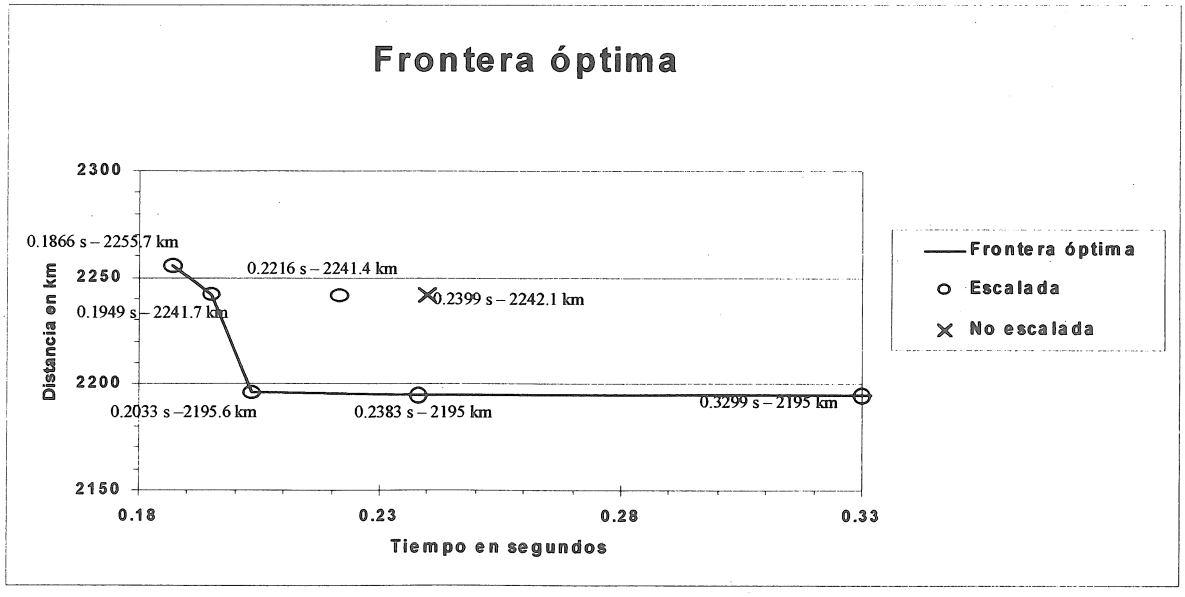

Figura 4: Gráfico con los mejores promedios de tiempo y distancia

4. PARALELISMO ASÍNCRONO DEL ALGORITMO ANT CYCLE

La gran proliferación de redes de computadoras con procesadores heterogéneos, típicamente asíncronos, y la búsqueda de mejores soluciones obtenidas en menor tiempo de procesamiento, justifican la paralelización de nuevos algoritmos [7]. En este sentido, los algoritmos *Ant System* tienen características que lo hacen especialmente apropiados para su paralelización y distribución entre los diversos procesadores de una red de computadoras o una computadora paralela. 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 serán transmitidos a los otros procesadores, colaborando todos en la solución del problema global.

Por otra parte, 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; en otras palabras, políticas definidas los siguientes parámetros:

- *El intervalo de migración*: que establece cada cuantos ciclos cierta cantidad de hormigas migraran 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 hormigas que hayan obtenido mejores soluciones (tour con menores distancias).

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 [8] a fin de mantener el número original de hormigas por cada ciudad. En otras palabras, las hormigas que hayan encontrado mejores soluciones, tienen mayor probabilidad de sobrevivir a la selección.

A continuación se presenta el pseudocódigo de la versión paralela, que utiliza un proceso *Master* que se encarga de administrar la implementación paralela (Pseudocódigo 3) incluyendo el lanzamiento de los procesos *Esclavos* (Pseudocódigo 4), para que éstos realicen los cálculos propiamente dichos.

```
Leer_Datos;
Levantar_Procesos_Esclavos;
Enviar_Parámetros_a_cada Esclavo;
Fin = FALSE;
DO WHILE NOT (Fin)
    Recibir MensajeTerminación_de Esclavos;
    IF (Todas las máquinas terminaron)
        Fin = TRUE;
    END IF
END DO
Enviar_Mensaje_de Fin_a_Esclavos;
Eliminar_Procesos_Esclavos;
```

Pseudocódigo 3: Proceso *Master* del *Ant-cycle* paralelo

```
Recibir_Parámetros;
Iniciar_Variables;
Leer_Datos;
DO WHILE (nc<=NCMAX y que todas las hormigas no realicen el mismo tour)
        Mover_hormigas                  /*hasta que cada una complete su tour*/
        Calcular_distancia_recorrida;
        Escoger_hormigas_migrantes;
        Enviar_hormigas_migrantes;      /*a otros procesos*/
        Recibir_hormigas_migrantes      /*de otros procesos*/
        Seleccionar_hormigas;
        Actualizar_matriz_de_feromonas;
        Guardar_camino_más_corto;
END DO
Enviar_Mensajes_de fin_al_Master
```

Pseudocódigo 4: Proceso *Esclavo* del *Ant-cycle paralelo*

5. RESULTADOS EXPERIMENTALES DE LA PARALELIZACIÓN

Para comparar el desempeño de los dos algoritmos *Ant Cycle* se hicieron pruebas con el problema Oliver30, problema del cajero viajante para 30 ciudades [9]. Se simularon procesos paralelos en una workstation DEC 3000 modelo 300 con procesador ALPHA de 150 MHz con 32 MB de memoria RAM, operando bajo el sistema operativo OSF/1 versión 2.0. Se hicieron 10 corridas de 5000 ciclos para cada una de las siguientes implementaciones:

- secuencial,
- paralelo con 2 procesos,
- paralelo con 4 procesos,
- paralelo, con 8 procesos.

A cada proceso hijo se le asigna un número de ciclos igual al número total de ciclos divido la cantidad de procesos hijos existentes, de forma a mantener constante el número de ciclos totales que son computados. Los tiempos obtenidos fueron obtenidos midiendo los tiempos utilizados por cada uno de los procesos, simulándose así un procesamiento paralelo.

En todas las experiencias computacionales realizadas con el problema de Oliver30, el algoritmo *Ant-Cycle* converge sin inconvenientes a la misma solución en 5000 ciclos. La Tabla 1 muestra los tiempos promedios empleados en cada caso, la aceleración y la eficiencia obtenida en cada caso, mientras la Figura 5 muestra la rápida reducción del tiempo de procesamiento en función del número de procesos hijos, indicando las claras ventajas de paralelizar el referido algoritmo.

Implementación	*Tiempo [s]*	*Acelaración*	*Eficiencia*
Secuencial	464,888	Referencia (1)	Referencia (100 %)
2 procesos hijos	301,346	1,5	77,1 %
4 procesos hijos	151,704	3,1	76,6 %
8 procesos hijos	78,297	5,9	74,2 %

Tabla 1: Resultados experimentales de la paralelización del algoritmo *Ant Cycle.*

Como se puede ver en la Tabla 1, la paralelización del algoritmo *Ant Cycle* es bastante eficiente, además de ser muy simple y natural en su implementación, por lo que es de esperar que se pueda utilizar un gran número de procesadores sin mayores inconvenientes.

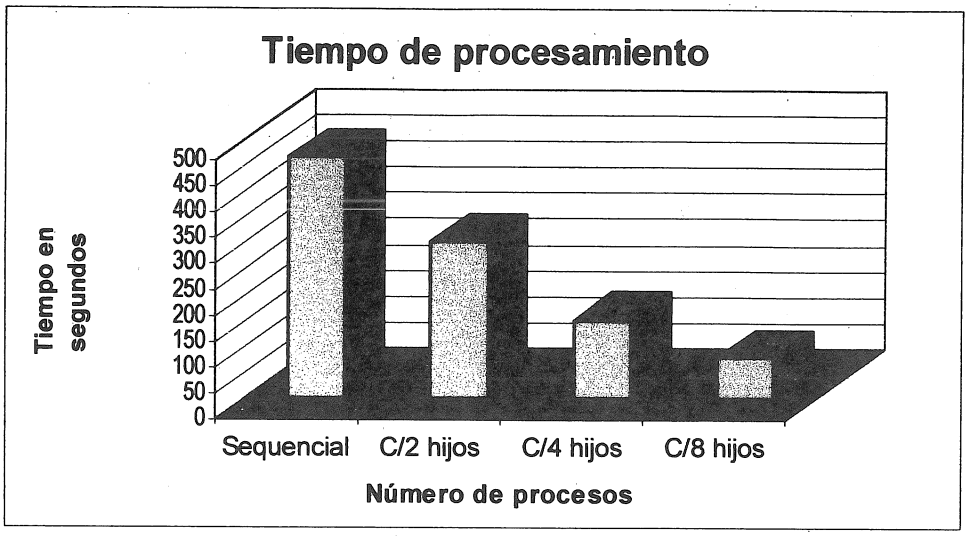

Figura 5: Promedios de tiempo experimentales obtenidos.

6. CONCLUSIONES

Se ha implementado una novedosa técnica de Inteligencia Artificial, conocida como *Ant System*, que basa su principio de funcionamiento en la observación de como las hormigas encuentran el camino más corto hasta su fuente de alimentación. A partir de esta técnica evolutiva propuesta inicialmente por Dorigo et al. [1] se presentan diversas mejoras a los algoritmos secuenciales originales, como ser:

- La inicialización de la matriz de feromonas con valores inversamente proporcionales a las distancias entre ciudades (sección 3), con lo que se logra acelerar la convergencia del algoritmo *Ant Quantity*, además de tener una influencia positiva en la búsqueda de buenas soluciones. Así, la sección 3 muestra resultados experimentales que demuestran la superioridad de esta propuesta, al encontrar los puntos de la frontera de soluciones óptimas en la gráfica tiempo-distancia (Figura 4), mientras el algoritmo sin escalamiento no consigue una sola solución óptima. En efecto, de 27 corridas realizadas (cada una repetida 10 veces) en 16 se encontró la solución óptima utilizando el algoritmo con la mejora propuesta mientras el algoritmo de Dorigo et al. sólo encuentra esta solución en 10 corridas de las 27. Si en cambio, comparamos los promedios de todas las corridas, el algoritmo sin inicialización tarda 0,208 s para encontrar un tour promedio de 2346.8 Km, mientras el método con inicialización de feromonas propuesto, requiere solo de 0.187 s para encontrar un mejor tour promedio de 2255.7 Km.

- La paralelización del algoritmo *Ant System*, para su implementación asíncrona en sistemas distribuidos como redes de computadoras. Para esto, se propuso paralelizar el cálculo de los diversos ciclos, permitiendo que las hormigas lleven información de un proceso a otro, siguiendo una política migratoria definida y manteniendo constante el número total de hormigas por ciudad (sección 4). Resultados experimentales surgidos de la paralelización del algoritmo *Ant Cycle* demuestran una buena eficiencia (del orden del 75 %) y escalabilidad, permitiendo la utilización de un creciente número de procesos, según sea la dimensión del problema en cuestión (sección 5).

En conclusión, el presente trabajo propone mejoras a una prometedora y novedosa familia de algoritmos evolutivos conocidos como *Ant System*, basada en agentes muy simples llamados *hormigas*, que prometen interesantes aplicaciones en áreas de optimización combinatoria, para lo cual los autores se encuentran trabajando en diversas aplicaciones del mismo utilizando implementaciones PVM (*Parallel Virtual Machine*) en redes de computadoras heterogéneas.

Referencias

[1] Dorigo M y Maniezzo V y Colorni A., "Ant System: An Autocatalytic Optimizing Process". *Technical Report 91-016*, Dipartimento di Electtronica e Informazione - Politecnico di Milano, Italia, pp. 1-26, 1991.

[2] Dorigo M., Maniezzo M. y Colorni A., "The Ant System: Optimization by a colony of cooperating agents", *IEEE Transactions on Systems, Man, and Cybernetics*-Part B, USA, Vol. 26, No. 1, pp. 1-13, 1996.

[3] Melo Ismael A., A través de feromonas los animales mantienen comunicación, URL: `http//www.uam.mx/ organo-uam/documentos/V-II/ii31-12.html.`

[4] Colorni A., Dorigo M., Maniezzo V., "Distributed Optimization by Ant Colonies". *European Conference on Artificial Life* (ECAL91), Paris-France, pp. 134-142, 1991.

[5] Dorigo M., Maniezzo V y Colorni A., "An investigation of some properties of an Ant algorithm". *Proceedings of the Parallel Problem solving from Nature Conference* (PPSN 92), Bruselas - Bélgica, 1992.

[6] Gutierrez M., De los Cobos S., Perez B., *Búsqueda Tabú: Un Procedimiento Heurístico para Solucionar Problemas de Optimización Combinatorial.* URL: `http://wwwazc.uam.mx/enlinea2/art_btab/tabuline.htm.`

[7] Chaparro E. y Barán B., "Algoritmos Asíncronos Combinados en un Ambiente Heterogéneo de Red", *XXIII Conferencia Latinoamericana de Informática (CLEI)*, Valparaiso – Chile, 1997.

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

[9] Whitley D., Starkweather, Fuquay D., "Sheduling Problems ant Traveling Salesmen: The Genetic Edge Recombination Operator", *Proc. Of the Third Int. Conf. On Genetic Algorithms*, Morgan Kaufmann, 1989.