Algoritmo ACO aplicado al TSP: Resumen de una experiencia práctica

Benjamín Arenas F., benarenas@hotmail.com
Alejandro Pavez S., apavez@mi.terra.cl
Rodrigo Vidal K., rovika@yahoo.com
Universidad Técnica Federico Santa María
Departamento de Informática

Abstract: La metaheurística Ant Colony Optimization (ACO) [4] es uno de los nuevos paradigmas de resolución de problemas combinatorios del tipo NP. En este artículo se presentan los resultados obtenidos al aplicar un algoritmo variante de la metaheurística ACO en torno al mundialmente conocido Traveling Salesman Problem (TSP) [10], se discuten las oportunidades de desarrollo futuro y se presentan nuevos tuors solución con mejores resultados que los mundialmente encontrados hasta hoy.

Palabras claves: TSP, ACO, Simulated Annealing, Metaheurística

1. Introducción

La importancia del TSP dentro del mundo de problemas NP radica en que es utilizado como conjunto de pruebas para los nuevos algoritmos asociados a la resolución de esta clase de problemas. Su naturaleza NP-Completo hace posible que se puedan resolver una gran familia de problemas equivalentes mediante transformaciones particulares para cada caso [8]. Además, como el TSP presenta una gran cantidad de aplicaciones en el mundo real, se ha convertido en uno de los problemas más estudiados por la comunidad científica mundial. Las características de la nueva metaheurística ACO, aplicadas al TSP [2], [3], [7], nos puede ayudar a encontrar soluciones satisfactorias para este problema.

A continuación, en el punto 1.1 se presentará el problema del TSP, en el punto 1.2 se presentará la analogía de la Colonia de Hormigas, y en el punto 1.3 se presentarán las Hormigas Artificiales. En el punto 2 se presenta la modificación propuesta, en el punto 3.1 se presentan las características del Hardware y el Software usado, en el punto 3.2 se presenta la Descripción de las pruebas realizadas, y en el punto 3.3 se presentan los resultados obtenidos. En el punto 4 se presentan las conclusiones y oportunidades de desarrollo futuro.

1.1. El problema del vendedor viajero[1]

El TSP [10] consiste en que una persona debe visitar un conjunto de *n* ciudades, comenzando en una ciudad específica (elegida arbitrariamente como origen), debiendo regresar a ella, luego de haber visitado todas las demás y sin haber visitado dos veces una misma ciudad. El objetivo es minimizar la longitud de la secuencia de visita de las ciudades, bajo el criterio de minimización (maximización) de la función objetivo. Un ejemplo de un tour es el que se muestra a continuación en el figura 1.

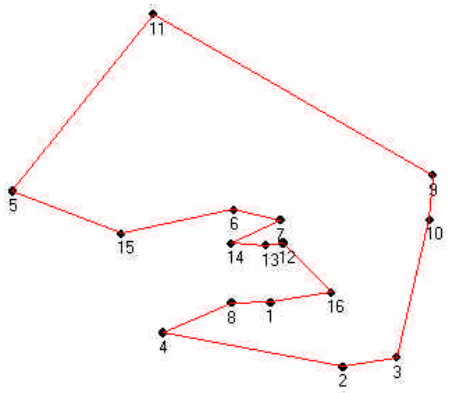

Figura 1 – Un ejemplo de tour TSP Euclidiano (ulisses16)

El problema puede ser representado a través de una matriz $n \times n$ (simétrica ó asimétrica) de distancias *d* donde d_{ij} es la distancia entre la ciudad *i* y la ciudad *j*. El valor de d_{ii} es siempre 0. Si tenemos *n* ciudades, existen a lo más *(n-1)!* circuitos diferentes (y *(n-1)! / 2* en el caso del TSP simétrico).

El problema es del tipo NP-Completo, es decir, es un problema representativo de la clase NP cuyo tiempo de resolución crece en forma exponencial ante incrementos lineales del número de elementos que intervienen en el problema. En otras palabras, no se ha encontrado un algoritmo que lo resuelva en tiempo polinomial.

Una representación del TSP asimétrico desde el punto de vista de la 'Programación Lineal' es la siguiente:

[1] En ingles TSP, acrónimo de Traveling Salesman Problem

$$\min \sum_i \sum_{i \neq j} d_{i,j} x_{i,j}$$

$$\text{tal que } \sum_i x_{i,j} = 1 \; \forall j$$

$$\sum_j x_{i,j} = 1 \; \forall i$$

$$\sum_{i \in S} \sum_{j \notin S} x_{i,j} \geq 1 \; \forall \text{ subconjunto apropiado } S, |S| \geq 2$$

$$\text{donde } x_{i,j} = \begin{cases} 1 & \text{si el tour pasa desde } i \text{ a } j \\ 0 & \text{si no} \end{cases}$$

y $d_{i,j}$ es la distancia desde i a j

En este trabajo se hará uso de problemas TSP simétricos, es decir: $d_{i,j} = d_{j,i}$.

1.2 La analogía de la colonia de hormigas

La Metaheurística [4] se basa en el comportamiento de una colonia de Hormigas [1]. Las hormigas reales son capaces de encontrar el camino más corto entre una fuente de comida y su nido - por ejemplo - sin usar mecanismos visuales, sino sólo explotando el rastro de Feromona[2].

Una forma en que las hormigas explotan la feromona para encontrar el camino más corto entre dos puntos se muestra en la figura 2.

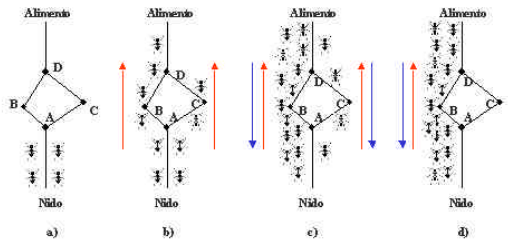

Figura 2 – Las hormigas y el camino más corto

En a) las hormigas llegan a un punto en el cual deben decidirse por uno de los dos caminos a seguir; en b) se realiza una elección de camino, la que puede ser aleatoria al haber bajos niveles de feromona o guiada al haber una diferencia notable entre la cantidad de feromona de cada camino; en c), dado que la velocidad de una hormiga se considera aproximadamente constante, es claro decir que las hormigas que eligieron el camino más corto se demoren menos que las otras en llegar hasta el otro extremo, lo cual conduce a una mayor acumulación de feromona en el camino. En d), la feromona acumulada en mayor cantidad en el camino más corto y más circulado guía a las hormigas a la fuente de alimento de la forma más rápida, donde el camino menos transitado pierde la feromona depositada en él a causa de la evaporación.

Informalmente, esta metaheurística trabaja de la siguiente manera:

"Cada hormiga genera un tour completo escogiendo ciudades de acuerdo a una regla de transición de estados probabilística. Las hormigas prefieren moverse a ciudades que están conectadas a caminos de bajo costo con una alta cantidad de feromona. Una vez que todas las hormigas han completado su tour, se aplica una regla de actualización de la feromona global. A continuación se evapora una fracción de la feromona, mientras que por otra parte algunas hormigas depositan una cantidad proporcional de ella sobre los caminos que formaron parte de su tour (de acuerdo al costo total de cada uno), lo cual provoca un incremento a través del tiempo. Luego, el proceso comienza a iterar. A través de este mecanismo, la colonia genera una convergencia común, dirigida hacia el camino más corto."

La actualización de feromona es la coordinación entre los distintos tours realizados en forma concurrente por cada una de las hormigas.

1.3 Las hormigas artificiales

De acuerdo a este comportamiento es posible definir el concepto de Hormiga Artificial. Un ejemplo de pseudocódigo de la implementación de la metaheurística ACO se presenta a continuación:

```
procedimiento ACO()
   establecer_feromona_inicial();
   mientras (criterio no satisfecho)
      crear_hormigas();
       para (cada hormiga)
          {mover_hormiga();}
          hasta (completar tour);
       fin para
       actualizar_feromona();
       destruir_hormigas();
    fin mientras
fin procedimiento
```

[2] Feromona (Pheromone), sustancia olorosa producida por un animal que afecta a la conducta de otros animales. Las hormigas acostumbran emplear una feromona para indicar el rastro que lleva hasta la comida, otra para provocar ataques contra los enemigos que han descubierto, y una tercera que señala la necesidad de huir.

```
procedimiento mover_hormiga()
   para (todo el vecindario factible)
      probabilidades_movimento();
   fin para
   ciudad_selec:=selec_movimiento();
   llevar_hormiga(ciudad_selecc);
fin procedimiento
```

Para este estudio se utilizó una implementación del algoritmo Ant Colony System (ACS) [7]. En él cada una de las *n* hormigas construye una solución (tour) del TSP. Inicialmente, cada una de las hormigas es colocada en alguna ciudad (en forma aleatoria o ciclomática). Una vez hecho esto, a cada hormiga se le aplica una regla de elección probabilística de los caminos a seguir. En particular, la probabilidad con que la hormiga *k*-ésima (actualmente en la ciudad *i*) decide ir a la ciudad *j* en la *t*-ésima iteración es:

$$p_{ij}^{k}(t) = \frac{\left[t_{ij}(t)\right]^{a} \cdot \left[h_{ij}\right]^{b}}{\sum_{l \in N_i^k} \left[t_{il}(t)\right]^{a} \cdot \left[h_{il}\right]^{b}} \qquad \text{si } j \in N_i^k$$

Ecuación 1 – Probabilidad de que la hormiga k-ésima decida ir a la ciudad j desde la ciudad i

donde $h_{ij} = 1/d_{ij}$ es un valor disponible *a priori* definido por la distancia d_{ij}, α y β son dos parámetros los cuales determinan la influencia relativa del camino de feromona y la distancia respectiva, y N_i^k es el vecindario factible para la hormiga *k*, esto es, el conjunto de ciudades a las cuales la hormiga *k* aún no ha visitado. El rol de los parámetros α y β es el siguiente. Si α=0, las ciudades cercanas tienen más probabilidad de ser seleccionadas: esto corresponde a un algoritmo clásico *voraz estocástico*[3] [8] (con múltiples puntos de inicio desde que las hormigas están distribuidas en las ciudades). Si β=0, sólo trabajará la amplificación de feromona: este método podría dirigirnos a una rápida situación de *estancamiento* con la correspondiente generación de tours los cuales, en general, son fuertemente subóptimos. La búsqueda de estancamiento se define como la situación donde todas las hormigas siguen la misma ruta y construyen un óptimo local. Por lo tanto, debe existir una mezcla entre la influencia de la información heurística y los caminos de feromona.

Después que todas las hormigas han construido su tour, los caminos de feromona son actualizados. Esto se hace primero disminuyendo el nivel de feromona en todos los arcos por un factor constante, para luego permitir a cada hormiga agregar feromona a los arcos que fueron visitados:

$$t_{ij}(t+1) = (1-r) \cdot t_{ij}(t) + r \cdot \Delta t_{ij}^{gb}(t)$$

Ecuación 2 – Ecuación de actualización de feromona para ACS.

donde $0 < r \leq 1$ es el factor de evaporación de la feromona. El parámetro r es utilizado para evitar acumulación ilimitada de feromona en los caminos y le permite al algoritmo "olvidar" las malas decisiones previamente tomadas. Si un arco no es elegido por las hormigas, su feromona asociada decrece exponencialmente. $\Delta t_{ij}^{gb}(t)$ es la cantidad de feromona que la mejor hormiga global *gb* coloca en el arco visitado. Esto se define como sigue:

$$\Delta t_{ij}^{gb}(t) = \begin{cases} 1/L^{gb}(\mathrm{t}) & \text{si arco}(i,j) \text{ es usado por la hormiga gb} \\ 0 & \text{en otro caso} \end{cases}$$

Ecuación 3 – Cantidad de feromona que la k-ésima hormiga coloca en el arco visitado

donde $L^{gb}(\mathrm{t})$ es el largo del tour la mejor hormiga global gb. Dado por la ecuación anterior, el mejor tour de una hormiga es el que más feromona recibió por arcos a través del tour. En general, los arcos que son usados por muchas hormigas y los cuales están contenidos en tours cortos podrían recibir más feromona y por consiguiente son más probables a ser seleccionados en futuras iteraciones del algoritmo.

2. La modificación propuesta

El trabajo se concentró en la implementación del algoritmo ACS agregando una modificación sustancial en el criterio de decisión de las hormigas. Esta modificación radica en aplicar el concepto de temperatura - definido en los algoritmos de tipo simulated annealing [8] - dentro de la cadena de decisión del próximo movimiento. Una vez calculadas las probabilidades tradicionales de movimiento, cada Hormiga tiene la opción aleatoria de moverse bajo el criterio de exploración temprana y explotación tardía. Esto se define como sigue:

$$Rand \geq p_{ij}^{k}(t) \Rightarrow j \equiv Rand(N_i^k)$$

Ecuación 4 – Modificación basada en temperatura.

donde se establece que si la probabilidad asociada a la ciudad *j* (Ver ecuación 1), seleccionada como el

[3] En ingles: *stochastic greedy*

próximo movimiento, es inferior a un número aleatorio, la nueva ciudad destino será asignada en forma aleatoria desde el vecindario factible N_i^k.

Claramente puede observarse que a medida que el algoritmo genere probabilidades mayores de movimiento debido al mejor tour encontrado, la probabilidad de asignación de un nuevo movimiento aleatorio es menor (este es análogo al concepto de temperatura).

Las modificaciones necesarias al pseudocódigo mostrado anteriormente sólo radican en el procedimiento mover_hormiga, lo cual se ve reflejado a continuación en el nuevo procedimiento:

```
procedimiento nuevo_mover_hormiga()
   para (todo el vecindario factible)
      probabilidades_de_movimento();
   fin para
   ciudad_selec:=selec_movimiento();
   si (RAND ³ probab(ciudad_selec))
    cuidad_selec:=RAND(vecind_fact);
   fin si
   llevar_hormiga (ciudad_selec);
fin procedimiento
```

3. La experimentación en la práctica

A continuación, en el punto 3.1 se presentará las características del hardware y software usado, en el punto 3.2 se presenta la descripción de pruebas realizadas, y en el punto 3.3 se presentan los resultados obtenidos.

3.1 Características del hardware y software usado

La implementación y ejecución de este algoritmo fueron realizadas en un equipo de tipo Intel Pentium 166MMX con sistema operativo Windows NT versión 4.0 y 64MB RAM. El orden de magnitud del tiempo asociado a las soluciones presentadas no superan los 10^2 segundos.

El software desarrollado permite, bajo una interfaz gráfica, establecer los parámetros asociados a los factores de evaporación ρ, α, β y el archivo del problema a resolver.

3.2 Descripción de las pruebas

Como esquema de trabajo se definió realizar las pruebas del algoritmo implementado utilizando tours probados y generales. Los tours se obtuvieron de la biblioteca TSPLIB[4], de los que se seleccionaron, los de tipo Euclidiano, es decir, el costo asociado a realizar el viaje desde la ciudad *i* hacia la ciudad *j* se define como la distancia asociada al ciclo euclidiano entre las ciudades del tour.

El parámetro α tomó los valores [0.5, 1, 1.5, 2, 2.5, 3, 3.5] mientras que el parámetro β se mantuvo constante en el valor 2, esto debido a que la ecuación de probabilidad de movimiento (Ver ecuación 1) normaliza los valores calculados a través de su término inferior. Además, las hormigas se distribuyeron tanto en forma aleatoria como ciclomática. Los valores del factor de evaporación ρ fueron [0.3, 0.5, 0.7]. El número de hormigas, en general, se tomó menor o igual al número de ciudades asociadas al problema. El número de iteraciones para cada corrida no superó las 100 unidades. Los tours seleccionados fueron ulisses16 y ulisses22.

3.3 Los resultados obtenidos

Los resultados obtenidos utilizando los tours seleccionados se presentan en la tabla 1 donde **TSP** indica el nombre del tour, **Valor actual** indica el mejor valor registrado la fecha en la biblioteca TSPLIB, **Valor encontrado** indica el o los mejores valores encontrados a través de la implementación realizada para este estudio, **Tour encontrado** detalla el tour asociado al Valor encontrado.

[4] http://www.iwr.uni-heildeberg.de/iwr/comopt/soft/TSPLIB95/TSPLIB.html

TSP	Valor actual	Valor encontrado	Tour encontrado
Ulisses16	74.108735958153	74.0964506569666	1-8-4-2-3-16-13-12-6-7-10-9-11-5-15-14
		74.0013360991673	1-8-4-2-3-16-12-13-14-6-7-10-9-11-5-15
		73.9998263127488	1-8-4-2-3-16-13-12-7-6-10-9-11-5-15-14
		73.9876180451750	1-3-2-4-8-15-5-11-9-10-7-6-14-13-12-16
Ulisses22	75.6651494713561	75.5431916856301	1-14-13-12-7-6-15-5-11-9-10-19-21-20-16-3-2-17-18-4-22-8
		75.4544852957371	1-14-13-12-7-6-15-5-11-9-10-19-21-20-16-3-2-17-4-18-22-8

Tabla 1 – Detalle de los resultados obtenidos

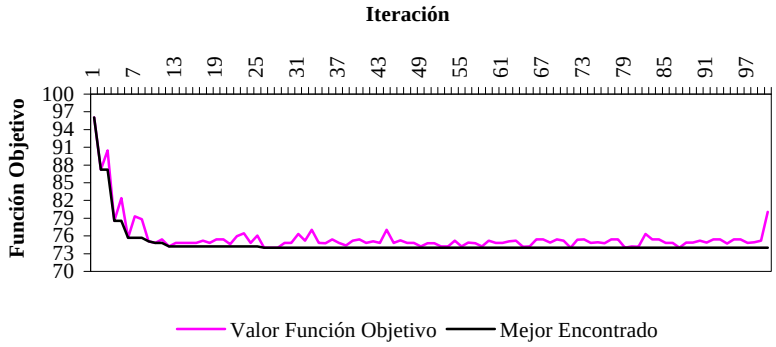

Gráfico 1 – Curva de comportamiento

El gráfico 1 muestra el comportamiento general del algoritmo híbrido en la búsqueda de soluciones. Se aprecia una rápida convergencia hacia el valor óptimo, además de una característica de búsqueda exploratoria, lo que permite escaparse de un óptimo local.

4. Conclusiones

Una de las proyecciones futuras es poder traspasar la actual implementación a pruebas, problemas y máquinas mucho más grandes, lo cual podría generar resultados aún más satisfactorios.

Otro punto importante es la aplicación de técnicas de paralelización, donde las distintas instancias de los grupos de agentes trabajan comunicando la mejor solución obtenida localmente cada cierto número de intervalos. Es posible pensar que al tener permanentemente la mejor solución global encontrada, esto podría ayudar a usar la técnica de evaporación de feromona con respecto a esta solución.

Además, es necesario realizar un estudio asociado al control de parámetros [9], lo cual podría ayudar a determinar patrones o guías con respecto a ellos, incluso la autoparametrización, para la obtención de resultados más cercanos a los óptimos buscados.

Una posibilidad de ayudar a la metaheurística, en el caso de ciclos euclidianos, radica en la aplicación de un operador de eliminación de cruces entre arcos (2-opt).

La metaheurística ACO es una técnica abierta, que permite complementarse con otras heurísticas existentes. En nuestro caso particular, simulated annealing se adaptó perfectamente dentro del flujo de decisión del próximo movimiento de las hormigas.

5. Agradecimientos

A María Cristina Riff por su constante insistencia, apoyo y ayuda, lo que llevó la investigación más allá de lo inicialmente propuesto.

Al Departamento de Informática USM por permitirnos presentar nuestro trabajo a la comunidad universitaria a través del coloquio de informática.

6. Bibliografía

[1] Dorigo M., V. Maniezzo & A. Colorni (1996). **The Ant System: Optimization by a Colony of Cooperating Agents.** *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26(1):29-41.

[2] Dorigo M. & L.M. Gambardella (1997). **Ant Colonies for the Traveling Salesman Problem.** *BioSystems*, 43:73-81.

[3] Dorigo M. & L.M. Gambardella (1997). **Ant Colony System: A Cooperative Learning**

Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53-66.

[4] Dorigo M. and G. Di Caro (1999). **The Ant Colony Optimization Meta-Heuristic.** In D. Corne, M. Dorigo and F. Glover, editors, *New Ideas in Optimization*, McGraw-Hill, in press.

[5] Dorigo M., G. Di Caro & L. M. Gambardella (1999). **Ant Algorithms for Discrete Optimization.** *Artificial Life*, 5(2), in press. Also available as Tech.Rep.IRIDIA/98-10, Université Libre de Bruxelles, Belgium.

[6] Colorni A., M.Dorigo, F.Maffioli, V. Maniezzo, G. Righini, M. Trubian (1996). **Heuristics from Nature for Hard Combinatorial Problems.** *International Transactions in Operational Research*, 3(1):1-21.

[7] Stützle T. and M. Dorigo (1999). **ACO Algorithms for the Traveling Salesman Problem.** In K. Miettinen, M. Makela, P. Neittaanmaki, J. Periaux, editors, *Evolutionary Algorithms in Engineering and Computer Science*, Wiley, 1999

[8] Adenso Días, Fred Glober, Hassan M. Ghaziri, J. L. González, Manuel Laguna, Pablo Moscato, Fan T. Tseng, **Optimización Heurística y Redes Neuronales**. Editorial Paraninfo, Primera Edición, 1996

[9] Michalewicz, Eiben, Hinterding, **Parameter Control on Evolutionary Algorithms**, *IEEE Transactions on Evlutionary Computation*, 1999

[10] Gerhard Reinelt, **The Traveling Salesman. Computational Solutions for TSP Applications**, Springer-Verlag, Computer Science Editorial I, 1994