Análisis del Ómicron ACO con optimización local

Pedro Esteban Gardel Sotomayor
Universidad Nacional de Asunción
Centro Nacional de Computación
San Lorenzo, Paraguay
pgardel@cnc.una.py

Osvaldo Gómez
Universidad Nacional de Asunción
Centro Nacional de Computación
San Lorenzo, Paraguay
ogomez@cnc.una.py

Benjamín Barán
Universidad Nacional de Asunción
Centro Nacional de Computación
San Lorenzo, Paraguay
bbaran@cnc.una.py

Abstract

La optimización por Colonia de Hormigas (*Ant Colony Optimization* o *ACO*) es una metaheurística inspirada por el comportamiento de búsqueda de alimentos de las hormigas. Esta metaheurística ha sido exitosamente empleada en la resolución de difíciles problemas de optimización combinatoria como el problema del cajero viajante (*Traveling Salesman Problem* o *TSP*). El presente artículo analiza el desempeño del *Ómicron ACO* (*OA*), una nueva alternativa de algoritmo *ACO*, comparándolo con el *MAX-MIN Ant System* (*MMAS*), uno de los *ACO* más reconocidos, en la resolución de dos instancias del *TSP* de 100 y 442 ciudades respectivamente. Con el objeto de realizar una comparación completa, se incluye un optimizador local (*Local Search*) como acelerador de convergencia, verificándose experimentalmente ciertas ventajas del *OA* sobre el más tradicional *MMAS.*

Palabras Claves: Inteligencia artificial, Optimización por Colonia de Hormigas, *Ómicron ACO*, *MAX-MIN Ant System*, Problema del Cajero Viajante, Optimización Local.

1- Introducción.

La Optimización por Colonia de Hormigas (*Ant Colony Optimization* o *ACO*) es una metaheurística propuesta por Dorigo et al. [1] que ha sido inspirada por el comportamiento de búsqueda de alimentos de las colonias de hormigas. En los últimos años *ACO* ha demostrado su efectividad en la resolución de diferentes problemas de optimización combinatoria considerados difíciles.
El *Ómicron OA* es un nuevo algoritmo *ACO* propuesto por Gómez y Barán [2] que ha demostrado excelentes resultados en la resolución de instancias pequeñas del Problema del Cajero Viajante (*Traveling Salesman Problem* o *TSP*) [2]. El *MAX-MIN Ant System* (*MMAS*) creado por Stützle y Hoos es considerado como uno de los mejores algoritmos *ACO* desarrollados hasta el momento [3]. En trabajos anteriores se comparó al *OA* con el *MMAS* sin optimizador local en pequeñas instancias del *TSP* [2], comprobándose un mejor desempeño del *OA*.
Con el objetivo de realizar una comparación más completa entre el *OA* y el *MMAS*, el presente estudio analiza el desempeño de ambos algoritmos al resolver instancias del *TSP* de mayor tamaño, de 100 y 442 ciudades respectivamente. Además, a fin de acelerar la convergencia, se incorporan un optimizador local y las modificaciones sugeridas por Stützle y Hoos [3] para la resolución de grandes instancias *TSP* con algoritmos *ACO*.
Debido a que los parámetros de *OA* aún no han sido configurados para hacerlos variables en el tiempo el mecanismo "*Pheromone Trail Smoothing (PTS)"* [3] no fue implementado en el *MMAS,* ni se estudió el comportamiento de los algoritmos en corridas muy largas.
El presente artículo está organizado de la siguiente manera. En la sección 2 se describe el problema de prueba y se dan ciertas definiciones necesarias. El enfoque típico de *ACO*, una descripción del *MMAS*, las modificaciones introducidas en el *OA* original y su pseudocódigo se presentan en la sección 3. En la sección 4 se presentan y explican los resultados experimentales de la comparación de desempeño del *OA* y el *MMAS*. Finalmente las conclusiones y los futuros trabajos se presentan en la sección 5.

2- Problema de Prueba.

Para el presente artículo se utilizó el Problema del Cajero Viajante Simétrico (*TSPs*) para comparar los algoritmos analizados. El *TSP* es un problema combinatorio difícil de resolver pero fácil de entender, que ha sido considerablemente estudiado por la comunidad científica.
ACO ha sido exitosamente aplicado por investigadores para resolver este problema [1,3]. Para realizar las comparaciones de desempeño se han usado instancias del *TSP* extraídas de la TSPLIB [4], fuente de instancias *TSP* ampliamente utilizada por investigadores del área. El problema del Cajero Viajante puede ser representado por un gráfico completo $G = (N, E)$ siendo N el conjunto de nodos, también llamados ciudades, y E el conjunto de arcos que conectan completamente todos los nodos. Cada arco $(i,j) \in E$ tiene asignado un valor d_{ij} que representa la distancia entre las ciudades i y j. El *TSP* es el problema de encontrar el recorrido más corto o recorrido óptimo, denotado como r^* , que visite cada uno de los $n= |N|$ nodos de G una vez exactamente. Para *TSP* simétricos la distancia entre las ciudades es independiente de la dirección en que se atraviesen los arcos, esto es $d_{ij} = d_{ij}$ para cada par de nodos. En este trabajo $l(r_x)$ denotará la longitud del recorrido r^*. Una población, *P,* es un conjunto de recorridos cualesquiera $P = \{P_i\}$. Supongamos que r_x y r_y son dos recorridos *TSP* o soluciones del mismo conjunto de ciudades, entonces se utilizaron las siguientes definiciones:

a) La distancia entre r_x y r_y, $\delta\ (r_x, r_y)$, es definida como:

n menos el número de arcos contenidos en ambos recorridos.

b) La distancia media de un recorrido a una población *P*, $\delta\ (P, r)$, se define como:

$$\delta\ (P,r) = \frac{1}{m} \sum_{i=1}^{m} \delta\ (P_i, r)$$

donde *m* es el tamaño de la población *P* (*m* = | *P* |).

c) La distancia media de una población, $\delta\ (P)$, se define como:

$$\delta\ (P) = \frac{2}{m(m-1)} \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \delta\ (P_i, P_j)$$

d) El espacio discreto de búsqueda total de una instancia *TSP*, *S,* se define como:

$$S = \{\ r_x\ \}$$

para todo r_x posible. Ω denota un subespacio de *S* , por lo tanto $\Omega \subseteq S$.

e) La zona central del subespacio definido por *P,* Ω_P, se define como:

$$\Omega_P = \{\ r_x \mid \delta\ (P, r_x) < \delta\ (P)\ \}$$

es el conjunto de recorridos con una distancia media a la población *P* menor que la distancia media de la población.

f) *CAOP* (Cantidad de arcos del óptimo no hallados en una población *P*) nos indica la cantidad de arcos de *r** que no se hallan en ningún un individuo de una población *P*.

3- Descripción de los algoritmos analizados.

Diversas modificaciones se han implementado en el algoritmo *ACO* original a fin de adecuarlo a la resolución de instancias *TSP* de gran tamaño. Estas variantes fueron introducidas principalmente para reducir el tiempo de computo sin disminuir la calidad de la solución final. En particular, Stützle y Hoos [3] presentaron una variedad de modificaciones especializadas para el *MMAS*. En consecuencia este trabajo utiliza estas mismas modificaciones también para el *OA*.

3.1 *Ant Colony Optimization ACO*.

Aún cuando existen diversas variantes de *ACO*, a continuación se presenta la que puede ser considerada como el enfoque típico [7].

ACO usa una matriz de feromonas $\boldsymbol{\tau} = \{\tau_{ij}\}$ para la construcción de soluciones potencialmente buenas. Los valores iniciales de $\boldsymbol{\tau}$ son fijados a un valor constante: $\tau_{ij} = \tau_0\ \forall\ (i,j)$ siendo $\tau_0 > 0$. También aprovecha la información heurística utilizando un parámetro $\eta_{ij} = 1/d_{ij}$. Situados en la

ciudad i, N_i representa el conjunto de ciudades aún no visitadas. La posibilidad de escoger la ciudad j estando en la ciudad i está definida por la siguiente ecuación:

$$P_{ij} = \begin{cases} \dfrac{\tau_{ij}^{\alpha} \times \eta_{ij}^{\beta}}{\sum_{\forall g \in N_i} \tau_{ig}^{\alpha} \times \eta_{ig}^{\beta}} & si \quad j \in N_i \\ 0 & si \quad j \notin N_i \end{cases} \tag{1}$$

donde los parámetros α y β definen la influencia relativa entre la información heurística y los niveles de feromonas.
En cada una de las iteraciones del algoritmo, cada hormiga comienza en una ciudad aleatoriamente escogida, luego va eligiendo al azar las ciudades a visitar del conjunto N_i mediante las probabilidades dadas por la ecuación (1), hasta completar un recorrido. La evaporación de las feromonas se aplica a todos los arcos (i,j) de acuerdo a:

$$\tau_{ij} = (\ 1 - \rho \ ) \times \tau_{ij} \tag{2}$$

donde el parámetro $\rho \in (0,1]$ determina la tasa de evaporación. Considerando una estrategia elitista, la mejor solución encontrada hasta el momento (r_{best}) actualiza τ de acuerdo a:

$$\tau_{ij} = \tau_{ij} + \Delta\tau \tag{3}$$

donde:

$$\Delta\tau = \frac{1}{l(r_{best})} \quad si \ (i,j) \in r_{best}$$
$$\Delta\tau = 0 \qquad si \ (i,j) \notin r_{best} \tag{4}$$

3.2 *MAX-MIN Ant System MMAS.*

Las principales diferencias entre el *MMAS* propuesto por Stützle y Hoos en [3] y el enfoque típico del *ACO* son:
1-) Valores máximo y mínimo son impuestos a la matriz de feromonas, esto es:

$$\tau_{min} \leq \tau_{ij} \leq \tau_{max} \qquad \forall \ (i,j) \in E \tag{5}$$

2-) Al construir los recorridos las hormigas utilizan una lista de "Ciudades candidatas", $C_i = \{c_1, c_2, c_3 \ldots c_p\}$ que contiene las p ciudades más cercanas a la ciudad i, en vez de N_i en la ecuación (1) de forma a elegir la siguiente ciudad a visitar. Únicamente si todas las ciudades de C_i ya han sido visitadas se le permite a la hormiga escoger una de las ciudades restantes de N_i. Las listas de

"Ciudades Candidatas" son guardadas en una matriz C donde cada fila C_i representa a la lista asociada a la ciudad *i.* Esta modificación proporciona un considerable ahorro de tiempo de cómputo sin perjudicar sensiblemente la calidad de la solución final [3].

3.3 *Ómicron ACO.*

En *OA*, una matriz de feromonas de valores constantes

$$\tau^o = \left\{ \tau_{ij}^o = 1 \quad \forall \ (i,j) \ \in \ E \right\} \tag{6}$$

es definida inicialmente [2]. Con esta matriz $\boldsymbol{\tau^0}$ se escoge una población utilizando $\boldsymbol{\tau^0}$ y la ecuación (1). Siguiendo el ejemplo del *MMAS*, se decidió usar una lista de "Ciudades candidatas" en vez de N_i en la ecuación (1). Resulta importante enfatizar que *OA* mantiene una población $P = \{P_x\}$ de m individuos o soluciones diferentes, las mejores soluciones encontradas hasta el momento. El mejor individuo de P en cualquier momento es denotado por P^*, mientras que el peor por P_{worst}. En cada iteración un nuevo individuo P_{new} es generado, éste reemplaza a P_{worst} en P si P_{new} es mejor que P_{worst} y diferente a todos los individuos de P.

Después de K iteraciones $\boldsymbol{\tau}$ es recalculada. Al principio $\boldsymbol{\tau} = \boldsymbol{\tau^0}$, luego a cada elemento τ_{ij} se le suma una cantidad igual a O/m cada vez que un arco (i,j) aparece en alguno de los m individuos de P. Este proceso se repite cada K iteraciones hasta que una condición de parada es alcanzada, (ver detalles en pseudocódigo). Nótese que $1 \leq \tau_{ij} \leq (1+O)$ donde $\tau_{ij} = 1$ si el arco (i,j) no está presente en ningún P_x, mientras que $\tau_{ij} = (1+O)$ si el arco (i, j) está presente en cada P_x de P. El parámetro O (*Ómicron*) inspira el nombre de este algoritmo [2].

3.4 Optimizador local.

El algoritmo de optimización local elegido como acelerador es el "*Fast 2-opt"* [5]. Este algoritmo opera de la siguiente manera: dado un recorrido legal r_0 (figura 1), se rompen dos arcos (N_1, N_2) y (N_k, N_{k+1}) obteniéndose dos semirrecorridos separados (figura 2). Seguidamente se reconectan los semirrecorridos generándose un nuevo recorrido r_2 que tiene una distancia de dos arcos con el original r_0 (figura 3). Se comparan las longitudes de ambos recorridos. Si $l(r_0) > l(r_2)$ se cambia r_0 por r_2. Nótese que para probar que $l(r_0) > l(r_2)$ basta con probar que $l(N_1, N_2) + l(N_k, N_{k+1}) > l(N_1, N_k) + l(N_2, N_{k+1})$. Este proceso se repite hasta que ya no sea posible encontrar un cambio que mejore el recorrido. Este recorrido final se considera como un óptimo local.

El algoritmo considera un tour dirigido, esto es, elegida una ciudad N_1 al azar, ya queda definido el arco (N_1, N_2) a romper.

Como el tiempo necesario para analizar todos los posibles movimientos es extremadamente alto, se realizaron las siguientes modificaciones al algoritmo original.

1) Una vez elegida la primera ciudad de corte N_1, el primer arco a romper (N_1, N_2) queda definido y se limita la búsqueda de la segunda ciudad de corte N_k al conjunto

de las l ciudades más cercanas a la ciudad N_1. Así se analizan los movimientos en los que se utilizan a los elementos de esta lista como segunda ciudad de corte N_k. Si después de haber agotados los elementos de la lista no fue posible encontrar un mejor recorrido, se elige otra ciudad diferente a N_1 como primera ciudad de corte.

2) Se creó una lista de "*Don't look bits*" (*DLB*) que asocia un bit a cada ciudad [5]. Inicialmente todos los bits están apagados (0). El bit asociado a la ciudad i (DLB_i) se enciende (1) cada vez que fracasa un intento por mejorar el recorrido habiéndose elegido $N_1 = i$. El bit asociado a la ciudad j se apaga cada vez que j es elegida como segunda ciudad de corte, esto es $DLB_j = 0$ cada vez que $N_k = j$. Al escoger a las ciudades candidatas para N_1 se consideran solo a aquellas con bits apagados.

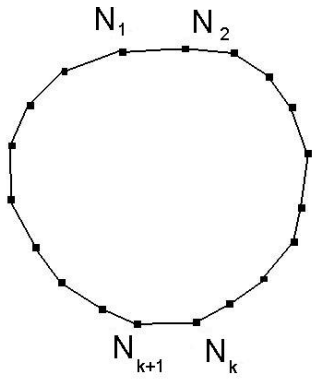


Figura :1

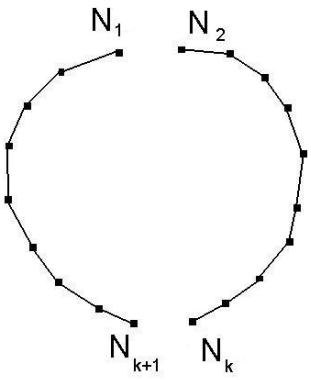


Figura: 2

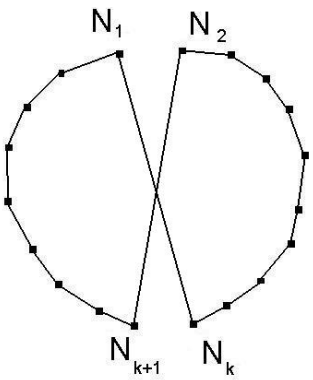


Figura: 3

Estas dos variaciones introducidas al algoritmo original disminuyen en muy poco la calidad de la solución final con un gran ahorro del tiempo de procesamiento [5].

A continuación el pseudocódigo del *OA* considerando un *TSP* con n ciudades:

Pseudocódigo de: *Ómicron ACO.*

Parámetros de entrada: n, matriz $D = \{d_{ij}\}$, O, K, α, β, C.
Parámetros de salida: P (m mejores soluciones).
τ = *Inicializar matriz de feromonas ().*
P = *Inicializar población* (τ^0).
Repetir hasta alcanzar condición de parada.
    Repetir K veces
        P_{new} = *Nueva solución* (τ).
        Fast 2-opt (P_{new}).
        Si $l(P_{new}) < l(P[0])$ y $P_{new} \neq$ a todos los elementos de P.
            P = *Actualizar población* (P_{new}, P).
        Fin si.
    Fin repetir.
    τ = *Actualizar matriz de feromonas* (P).
Fin repetir.

Pseudocódigo de: *Inicializar matriz de feromonas* ().
Repetir para cada arco (*i,j*).

$\tau_{ij} = 1$.

Fin repetir.

Pseudocódigo de: *Inicializar población* (τ).
$x = 1$.
Mientras x<m

P_{new} = *Nueva solución* (τ).
Si *Pnew* ≠ cada individuo de *P*.

$P[x] = P_{new.}$

Fin si.
$x = x+1$.

Fin mientras.
P = Ordenar *P* de peor a mejor de acuerdo a la longitud del tour.
% $P_{worst} = P[0]$.

Pseudocódigo de : *Nueva solución.*
$P_{new}[0]$ = Seleccionar una ciudad al azar (*i*). %Inicio del recorrido.
x=1.
Mientras x<n %Recorrido sin terminar.

P[*x*] = Seleccionar una ciudad mediante la ecuación (1) y lista $C_{i.}$
$x = x+1$.

Fin mientras.

Pseudocódigo de: *Actualizar población* (P_{new}, *P*).
$P[0] = P_{new.}$
P = Ordenar *P* de peor a mejor de acuerdo a la longitud del tour % $P_{worst} = P[0]$.

Pseudocódigo de: *Actualizar matriz de feromonas.*
τ = *Inicializar matriz de feromonas* ().
$x = 0$.
Mientras x < m

Repetir para cada arco (*i, j*) de *P*[*x*].

$\tau_{ij} = \tau_{ij} + O/m.$

Fin repetir.
$x = x+1$.

Fin mientras.

Pseudocódigo de: *Fast 2-opt.*

Parámetros de entrada: matriz C, l, P_{new}.
Inicializar lista "Don't look bits" ().
Mientras algún $DLB_j \neq 0$.
Elegir una ciudad al azar: $P_{\text{new}}(i)$ que tenga $DLB_i = 0$.
$N_1 = P_{\text{new}}(i)$, $N_2 = P_{\text{new}}(i+1)$.
c =1, salida = 0.
Mientras $DLB_i = 0$ y salida = 0
$N_k = P_{new}(C_i(c))$, $N_{k+1} = P_{new}(C_i(c)+1)$.
Si $l(N_1, N_2) + l(N_k, N_{k+1}) > l(N_1, N_k) + l(N_2, N_{k+1})$.
Cambiar recorridos$(P_{new}, N_1, N_2, N_k, N_{k+1})$.
$DBL(C_i(c)) = 0$.
salida = 1.
Fin si.
$c = c + 1$.
SI $c > l$. %Ya se utilizaron las l ciudades
$DLB_i = 1$.
Fin si.
Fin mientras.
Fin mientras.

Pseudocódigo de: *Inicializar lista "Don't Look bits".*
DBL= vector nulo de longitud n.

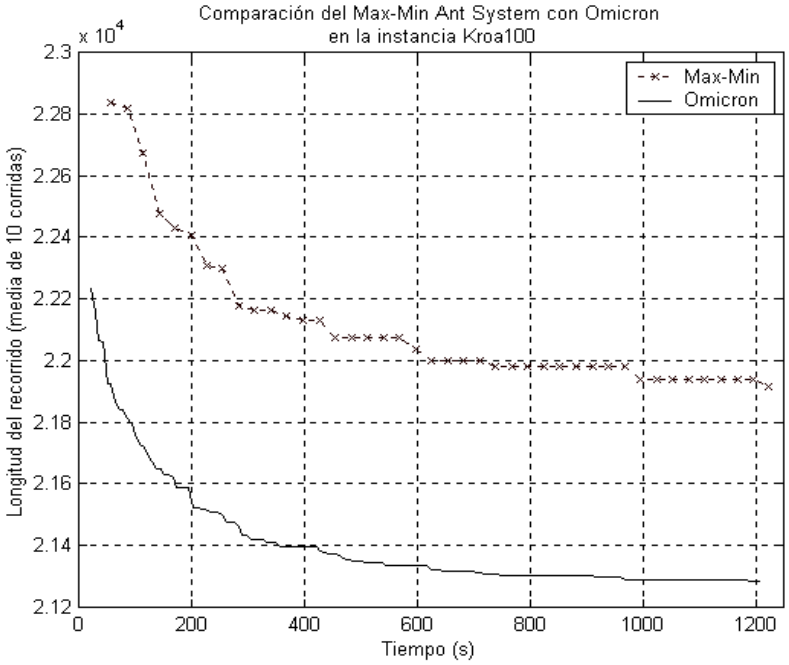
Pseudocódigo de: *Cambiar recorridos* $(P_{new}, N_1, N_2, N_k , N_{k+1})$.
P_{new} = [*Sendero*$(N_1, .., N_{k+1}.)$, *Sendero*$(N_2 , ..., N_k)$]*.

*Observación: *Sendero*$(N_1, .., N_{k+1}.)$ representa al semirrecorrido dirigido que sale de N_1 y llega a N_{k+1}.

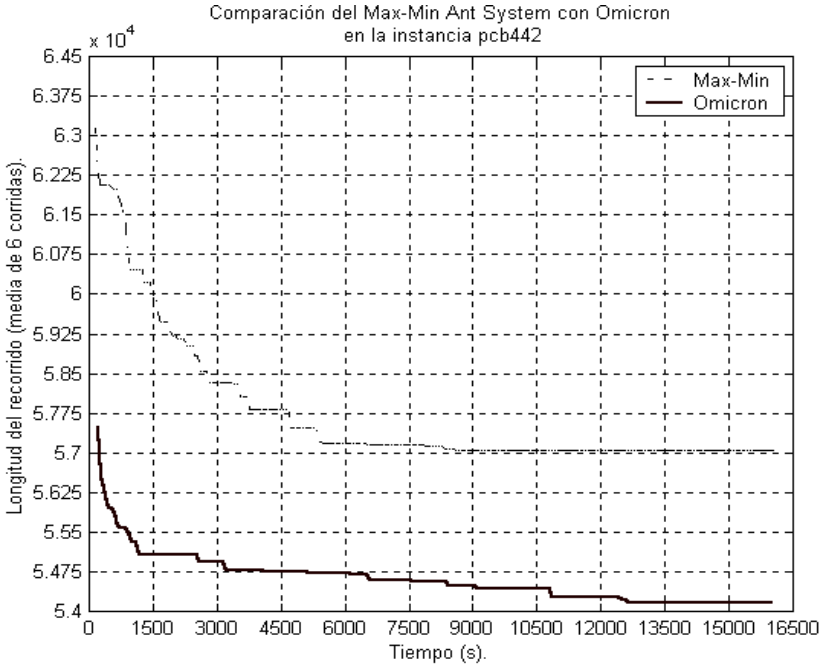
4- Resultados experimentales.

Para la comparación de desempeño todos los parámetros del *MMAS* fueron extraídos de [3], en especial α=1, β=2 y p = 20, siendo $p = |C_i|$. Los mismos valores de α, β y p fueron usados en *OA* también. Los demás parámetros del *OA* que fueron hallados empíricamente en trabajos anteriores [2] son: O = 600, m = 25 y K = 1000. Para el "*Fast 2-opt"* se utilizó un conjunto con l = 20 por estar dentro de los valores sugeridos por Johnson y McGeoch [5] (veasé sección 3.4).
Las observaciones fueron registradas en función del tiempo, dado que el concepto de iteración o generación es diferente para los algoritmos analizados.

Para la instancia Kroa100 del TSPLIB [4] se muestra el promedio de 10 corridas de 1200 segundos (figura 4) y para la instancia Pcb442 se promediaron 6 corridas de 15000 segundos cada una (figura 5). Para el estudio de la instancia Kroa100 se determinó el tiempo de 1200 segundos porque en ese tiempo se halló el óptimo en 8 de las 10 corridas con *Ómicron*. Para la instancia Pcb442 se fijó el tiempo máximo en 15000 segundos por ser de un orden mayor de magnitud, misma relación de magnitud que existe entre la cantidad de ciudades de las instancias. Además, los parámetros de *Ómicron* aún no se hicieron variables en el tiempo, razón por la cual no se analizó su desempeño en corridas muy largas.



Figura(4): Longitud en función del tiempo para la instancia Kroa100



Figura(5): Longitud en función del tiempo para la instancia Pcb442.

En general, *OA* produce mejores resultados que el *MMAS* desde el principio y converge más rápidamente hacia resultados ligeramente mejores como se puede apreciar en las figuras 4 y 5. En la instancia Kroa100 *OA* llegó al óptimo en 8 de las 10 las corridas, mientras que el *MMAS* no lo consiguió en ninguna ocasión. En la instancia Pcb442 ninguno de los algoritmos llegó al óptimo en el tiempo preestablecido para la corrida, pero el *OA* llegó a mejores soluciones en esa etapa inicial en todas las corridas.

Gómez y Barán propusieron en [7] que la mayor razón del éxito de *OA* es su habilidad de buscar en una zona central Ω_P de P donde usualmente se hallan mejores soluciones [7], inclusive se espera que r^* se encuentre en Ω_P. Esta suposición se basa en la creencia de que el *TSP* tiene un espacio de soluciones globalmente convexo como fue sugerido en estudios anteriores [3, 6, 7, 8].

OA concentra su búsqueda de nuevas soluciones en Ω_P. Esto se debe a que en la construcción de una nueva solución P_{new}, una mayor probabilidad es asignada a los arcos de cada P_x. En consecuencia se espera que P_{new} tenga una gran cantidad de arcos de $P_x \in P$ [7].

Debido a la importancia que tiene P, se decidió analizar las características de las poblaciones generadas por *OA* para la instancia Pcb442 por ser ésta la mayor.

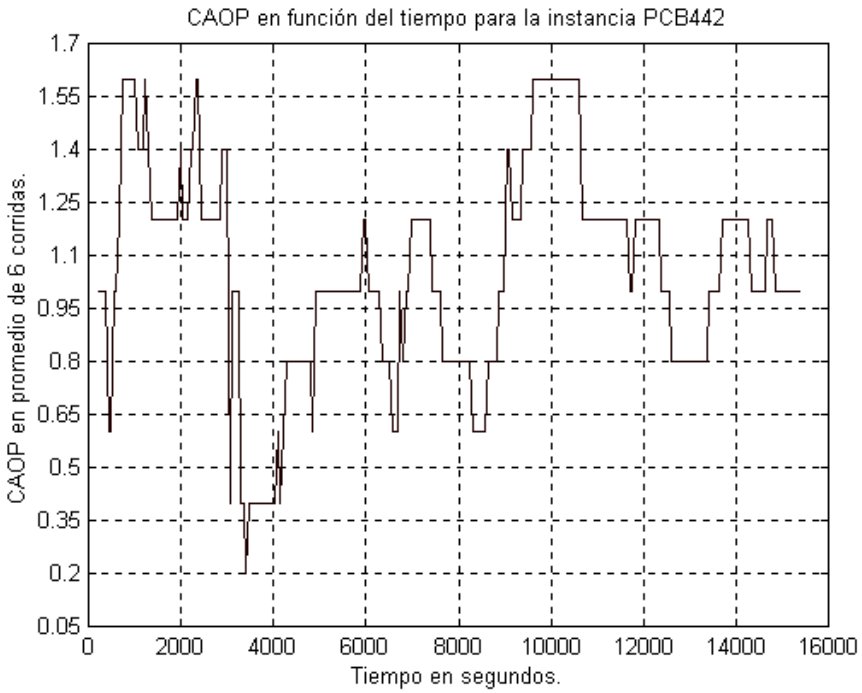
Como se muestra en la figura 6 las poblaciones generadas por *OA* mantienen en promedio un *CAOP* menor a 2, es decir, mantienen la información necesaria para hallar r^* o al menos un recorrido muy bueno a sólo un par de arcos de distancia.

En la tabla 1 se muestran las características de P para las 6 corridas hechas con *OA*. $|E|$ denota la cantidad total de arcos de S, $|E_{(P)}^*|$ es la cantidad de arcos diferentes hallados en P. El porcentaje de $|E|$ que $|E_{(P)}^*|$ representa nos da una idea de la relación entre los tamaños de S y Ω_P. Este bajo porcentaje indica la capacidad de *OA* de reducir considerablemente el espacio de búsqueda.

$|E_{(P)}|$ indica la cantidad de arcos que componen a la población P ($|E_{(P)}| = m * n$). $|E_{(P)}, r^*|$ indica la cantidad de arcos del óptimo hallados en los diferentes individuos de P. El alto porcentaje de $|E_{(P)}|$ que $|E_{(P)}, r^*|$ representa, demuestra que los arcos de r^* tienen asociados una alta probabilidad de ser elegidos para formar parte de P_{new}.

Número de corrida:	$\|E\|$	$\|E_{(P)}^*\|$	Porcentaje de $\|E\|$:	$\|E_{(P)}\|$	$\|E_{(P)}, r^*\|$	Porcentaje de $\|E_{(P)}\|$
1	97461	1186	1,22%	11050	7570	68,51%
2	97461	1170	1,20%	11050	7556	68,38%
3	97461	1191	1,22%	11050	7576	68,56%
4	97461	1148	1,18%	11050	8102	73,32%
5	97461	1133	1,16%	11050	7534	68,18%
6	97461	1183	1,21%	11050	7425	67,19%
Promedio:	97461	1168,5	**1,20**%	11050	7627,12	**69,02**%

Tabla 1: Resumen de la cantidad de arcos encontrados en la población P generada por *Ómicron* para la instancia Pcb442. Muestra la cantidad de arcos diferentes que componen P para cada corrida, $|E_{(P)}^*|$, así como la cantidad de arcos de r^* que se encuentran en los individuos que componen a P, $|E_{(P)}, r^*|$.



Figura(6): *CAOP* en función del tiempo para la instancia Pcb442

5- Conclusiones.

El Omicron *ACO* (*OA*) supera apreciablemente en desempeño al *MMAS* para instancias pequeñas [2] y para instancias mayores converge más rápidamente a soluciones mejores (ver figuras 4 y 5). Esto ocurre aún cuando se utilizaron parámetros adecuados al *MMAS* sin ser optimizados para el *OA*. Además, *OA* reduce notablemente el espacio de búsqueda sin perder substancialmente la información necesaria para hallar la solución óptima.
Para futuros trabajos se buscará ajustar los parámetros del *OA* para implementarlo en instancias de gran tamaño en presencia de un optimizador local. Además, se implementará al *OA* con parámetros variables en el tiempo y se probará su desempeño en corridas muy largas.

Referencias.

[1] Dorigo M. y Di Caro G. The Ant Colony Optimization Meta-heuristic. In David Corne, Marco Dorigo and Fred Glover, editors, New Ideas in Optimization. McGraw-Hill, Londres, 1999.

[2] Gómez O. y Barán B. *Ómicron ACO*, a ser publicado en: Conferencia Latinoamericana de Informática CLEI2004 septiembre 2004, Arequipa, Perú.

[3] Stützle T. y Hoos H. *MAX-MIN* Ant System. Future Generation Computer Systems, 16(8):889-914, junio 2000.

[4] TSPLIB. Accesible en: http://www.iwr.uni-heidelberg.de/iwr/comopt/software/TSPLIB95/

[5] 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: Nueva York, 1997.

[6] Gómez O., Barán B. y Gardel P. Estudio del Espacio de Soluciones del *Traveling Salesman Problem*. a ser publicado en: Conferencia Latinoamericana de Informática CLEI2004 septiembre 2004, Arequipa, Perú.

[7] Gómez O. y Barán B. Reasons of ACO's Success in TSP. En M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, y T. Stützle, editores, Proceedings of ANTS 2004 - Fourth International Workshop on Ant Colony Optimization and Swarm Intelligence. A ser publicado, tomo 3172 de LNCS. Springer-Verlag, Bruselas, septiembre 2004.

[8] Boese, K.D. Cost versus Distance in the Traveling Salesman Problem. Technical Report 950018, University of California, Los Angeles, Computer Science Department 1995.