Enrutamiento Multicast Multiobjetivos basado en Colonia de Hormigas

Diego Pinto
Centro Nacional de Computación - Universidad Nacional de Asunción
Ciencias y Tecnología - Universidad Nuestra Señora de la Asunción
Asunción, Paraguay
dpinto@cnc.una.py

Hugo Estigarribia
Centro Nacional de Computación - Universidad Nacional de Asunción
Asunción, Paraguay
hestigarribia@cnc.una.py

Benjamín Barán
Centro Nacional de Computación - Universidad Nacional de Asunción
Ciencias y Tecnología - Universidad Católica Nuestra Señora de la Asunción
Asunción, Paraguay
bbaran@cnc.una.py

Resumen

Los algoritmos de optimización basados en Sistemas de Colonias de Hormigas (*Ant Colony Optimization* - ACO) son métodos meta-heurísticos recientes, inspirados en el comportamiento de colonias de hormigas reales. Este trabajo propone un algoritmo multiobjetivo, para la resolución del problema de Enrutamiento Multicast en Ingeniería de Tráfico, denominado *Multiobjective Ant Colony System* (MOACS), que está basado en ACO y que es utilizado para la construcción del *árbol multicast*, en el contexto de transmisión de datos en redes de computadoras. El MOACS optimiza de manera simultánea tres parámetros: el costo del *árbol multicast*, el retardo promedio y el retardo máximo (de origen a destino) y obtiene como resultado un conjunto de soluciones óptimas, denominadas conjunto Pareto. Este conjunto de soluciones óptimas es calculado en una sola ejecución del algoritmo sin la necesidad de considerar decisiones a priori.
Los resultados experimentales obtenidos con el MOACS fueron comparados con el *Multiobjective Multicast Algorithm* (MMA) que demostró ser un excelente algoritmo evolutivo basado en el SPEA. Esta comparación muestra que el MOACS presenta mejores soluciones que las arrojadas por el MMA, indicando que los algoritmos basados en colonias de hormigas son muy prometedores en la resolución de problemas multiobjetivos.

Palabras claves: **Algoritmo Evolutivo, Ingeniería de Tráfico, Enrutamiento Multicast, Optimización Multiobjetivo, Frente Pareto, Colonias de Hormigas.**

Abstract

The optimization algorithms based on Ant Colony Optimization (ACO) are recent meta-heuristic methods, inspired by the behavior of real ant colonies. This work presents a multiobjective algorithm; for the resolution the Multicast Routing problem in the Traffic Engineering and it is denominated *Multiobjective Ant Colony System* (MOACS). The MOACS is based in ACO and it is used in the construction of the *multicast tree*, in the context of the data transmission and computers network. The MOACS simultaneously optimizes the cost of the *multicast tree*, the average delay and the maximum end-to-end delay and it obtains as result a group of good solutions without the considerations of *a priori* restrictions.
The experimental results obtained with MOACS were compared with the *Multiobjective Multicast Algorithm* (MMA), which have demonstrated to be an excellent evolutionary algorithm based in the SPEA. This comparison shows that the MOACS presents better solutions that those obtained by the MMA, indicating that the algorithms based on the ant colonies are very promising in the resolution of multi objective problems.
Keywords: **Evolutionary Algorithms, Traffic Engineering, Multicast Routing, Multiobjective Optimization, Pareto Front , Ant Colony Optimization**.

1 Introducción

Una transmisión m*ulticast* consiste en el envío simultáneo o concurrente de datos desde una fuente a un conjunto de nodos destinos componentes de una red de computadoras [12]. En estos últimos años el enrutamiento *multicast* se ha vuelto más importante debido a la creciente utilización de nuevas aplicaciones de transmisión punto a multipunto, como lo son la transmisión de radio y televisión, vídeo bajo demanda, teleconferencias y enseñanza a distancia, entre otras. Tales aplicaciones generalmente tienen requerimientos mínimos que buscan garantizar una calidad del servicio.

Considerando esta creciente tendencia, el retardo desde la fuente a cada uno de los destinos se torna una variable de vital importancia en transmisiones Multicast de audio y/o vídeo [9]. En la Ingeniería de Tráfico Multicast un punto importante son los "costos" del árbol, entendiéndose por "costos" otras métricas a ser minimizadas como: el número de saltos (*hop count*), la utilización del ancho de banda, etc. De esta forma, la Ingeniería de Tráfico Multicast puede ser abordada y planteada como un problema multiobjetivo.

La Optimización por Colonia de Hormigas (ACO) es una metaheurística, propuesta por Dorigo et al. [4], que se inspira directamente en el comportamiento de las colonias de hormigas reales para solucionar problemas de optimización combinatoria. Se basa en una colonia de hormigas artificiales, esto es, unos agentes computacionales simples que trabajan de manera cooperativa y se comunican mediante rastros de feromona artificiales en la búsqueda de mejores soluciones.

Varios algoritmos basados en ACO tratan el enrutamiento multicast como un problema mono-objetivo, minimizando el costo del árbol sujeto a múltiples restricciones. En [8] Y. Liu y J. Wu proponen la construcción de un árbol multicast, donde solamente se minimiza el *costo del árbol* con restricciones en el grado de cada nodo. Por otro lado, Gu et al. consideran múltiples métricas de Calidad de Servicio como restricciones y minimizando también el *costo del árbol* [7].

Se puede notar claramente que los algoritmos anteriores tratan el problema de Ingeniería de Tráfico Multicast como un problema mono-objetivo con varias restricciones. Nótese que la principal desventaja del enfoque basado en restricciones es la necesidad de una cota máxima (o mínima) definida *a priori*, que puede descartar buenas soluciones.

En este trabajo se propone resolver el problema de Ingeniería de Tráfico Multicast, considerando el algoritmo *Multiobjective Ant Colony System* (MOACS), presentado por M. Schaerer y B. Barán [10]. Este algoritmo optimiza varias funciones objetivos simultáneamente, con resultados experimentales que demostraron que es el mejor algoritmo multiobjetivo basado en *ACO* para el Problema del Cajero Viajante (*TSP*) bi-objetivo [6].

Por otra parte, fueron comparados los resultados obtenidos con el MOACS con los del algoritmo *Multiobjective Multicast Algorithm* (MMA) [3] que está basado en el *Strength Pareto Evolutionary Algorithm* (*SPEA*) [14]. El MMA optimiza simultáneamente, tres funciones objetivos para el caso estático en [1] mientras que en [2] optimiza cuatro objetivos para el caso dinámico.

En resumen, este trabajo toma uno de los mejores algoritmos de colonias de hormiga multiobjetivo (MOACS) y lo adapta para su utilización en la Ingeniería de Tráfico Multicast y también compara el MOACS con un algoritmo evolutivo recientemente publicado (MMA), especialmente diseñado para el problema en cuestión.

2 Formulación del Problema

La red es modelada como un grafo dirigido $G = (V, E)$, donde V es el conjunto de vértices y E el conjunto de arcos. Los vértices del grafo representan nodos de la red, y los arcos representan los enlaces entre los nodos. Sea:

$(i,j) \in E$: Enlace entre los nodos i y j; $i, j \in V$.

$c_{ij} \in \Re^+$: Costo del enlace (i,j).

$d_{ij} \in \Re^+$: Retardo del enlace (i,j).

$s \in V$: Nodo origen un grupo multicast.

$N_r \subseteq V\text{-}\{s\}$: Conjunto de nodos destinos de un grupo multicast.

$\phi \in \Re^+$: Demanda de tráfico del grupo multicast, en bps.

$T(s,N_r)$: Árbol multicast con origen en s y conjunto de destinos N_r.

$p_T(s, n) \subseteq T(s,N_r)$: Camino que conecta el nodo fuente s y un nodo destino $n \in N_r$.

$d(p_T(s, n))$: Retardo del camino $p_T(s,n)$, dado por la suma de los retardos de los enlaces que conforman el camino. Esto es,

$$d\left(p_T\left(s,n\right)\right) = \sum_{(i,j)\in p_T(s,n)} d_{ij} \qquad (1)$$

Usando las notaciones definidas arriba, el problema de enrutamiento multicast puede ser definido como un Problema Multi-Objetivo (*MOP*) [15], que trata de hallar un conjunto de árboles multicast $T(s,N)$[1] minimizando las siguientes funciones objetivos:

a. Costo del árbol:

$$f_1(T) = \phi \cdot \sum_{(i,j)\in T} c_{ij} \tag{2}$$

b. Retardo máximo de extremo a extremo:

$$f_2(T) = \underset{n\in N_r}{Max} \left\{ d\left(p_T(s,n)\right) \right\} \tag{3}$$

c. Retardo medio:

$$f_3(T) = \frac{1}{|N_r|} \cdot \sum_{n\in N_r} d\left(p_T(s,n)\right) \tag{4}$$

Dentro del marco presentado y dadas las soluciones T y T' para un mismo grupo multicast (s,N_r), con sus correspondientes vectores objetivos:

$$x = \left[f_1(T) \quad f_2(T) \quad f_3(T) \right] \qquad \text{y} \qquad z = \left[f_1(T') \quad f_2(T') \quad f_3(T') \right]$$

Puede darse una de las siguientes tres condiciones en el contexto de dominancia [15]:

$$\begin{aligned} &x \succ z \ (x \text{ domina a } z) && \text{si y solo si } x_i \le z_i \wedge x_i \ne z_i \ \forall i \in \{1,2,3\} \\ &z \succ x \ (z \text{ domina a } x) && \text{si y solo si } z_i \le x_i \wedge z_i \ne x_i \ \forall i \in \{1,2,3\} \\ &x \sim z \ (x \text{ y } z \text{ son no-comparables}) && \text{si y solo si } x_i \not\succ z_i \wedge z_i \not\succ x_i \ \forall i \in \{1,2,3\} \end{aligned} \tag{5}$$

Alternativamente, $x \rhd z$ denota que $x \succ z$ o $z \sim x$.

Dado una solución $T \in X_f$ *(espacio de decisión factible)*, se dice que T es no-dominado respecto a un conjunto $Q \subseteq X_f$ si y solo si $T \rhd T'$, $\forall T' \in Q$. En caso que T sea no-dominado respecto a todo el conjunto X_f, y solo en ese caso, se dice que T es una solución Pareto óptima. Por lo tanto, el conjunto Pareto óptimo X_{true} puede ser definido formalmente de la siguiente manera:

$$X_{true} = \{ T \in X_f \mid T \text{ es no-dominado con respecto a } X_f \} \tag{6}$$

El correspondiente conjunto de vectores objetivo $Y_{true} = f(X_{true})$ constituye el Frente Pareto óptimo.

3 Multiobjective Ant Colony Optimization

El *Multiobjective Ant Colony Optimization (MOACS)* propuesto por M. Schaerer y B. Barán en [9], es una generalización del *ACS* [5]. Este enfoque utiliza en cada generación una colonia de hormigas para la construcción de m soluciones T, luego el Frente Pareto conocido Y_{know} [15] es actualizado con las soluciones no-dominadas, y termina el ciclo modificando la matriz de feromonas τ_{ij}. En la Figura 1 se presenta el procedimiento general del MOACS:

```
Inicio (MOACS)
   Leer grupo multicast (s,Nr) y demanda de tráfico ϕ
   τij = τ0        ∀(i,j)∈V
   Repetir hasta cumplir condición de parada
      Desde k=1 hasta m repetir
         T = Construir Árbol, con Pseudocódigo 3(Figura 3)
         Si (T ⊀ {Tx | Tx ∈ Yknow}) entonces
            Yknow = Yknow ∪ T − {Ty | T≻Ty} ∀Ty ∈ Yknow
         Fin Si
         Si (Yknow fue modificado) entonces
            τij = τ0        ∀ (i,j)∈V
         Sino
            Actualización global de τij, con Pseudocódigo 2.
         Fin Si
      Fin Desde
   Fin Repetir
Fin
```

Figura 1. Procedimiento general del *MOACS (Pseudocódigo 1)*

[1] En lo que resta de este trabajo $T \equiv T(s,N_r)$ para mayor simplicidad.

donde:

τ_{ij}	Matriz de feromonas.
τ_0	Nivel de feromonas inicial.
Y_{know}	Frente Pareto conocido.
T	Árbol multicast en construcción.

La actualización de τ_{ij} depende del estado de Y_{know}. Si éste fue modificado, entonces es inicializado para mejorar la exploración. En otro caso, la explotación con las soluciones de Y_{know} son utilizadas para una actualización global de τ_{ij}, así como se presenta en el siguiente procedimiento indicado en la Figura 2.

```
Inicio (Actualización global de τij)
    Repetir para todo T ∈ Yknow
        Repetir para todo (i,j) ∈ T
            τij =(1-ρ).τ0 +ρ.Δτ
        Fin Repetir
    Fin Repetir
Fin
```

Figura 2. Actualización global de τ_{ij} *(Pseudocódigo 2)*

donde:

$$\Delta\tau = \frac{1}{\sum_{\forall T \in Y_{know}} (f_1(T) + f_2(T) + f_3(T))} \tag{7}$$

donde:

$f_1(T)$	Costo normalizado[2] de T, dada por la ecuación (2).
$f_2(T)$	Retardo medio normalizado de T, dada por la ecuación (3).
$f_3(T)$	Retardo máximo normalizado de T, dada por la ecuación (4).
$\rho \in (0,1]$	Parámetro de decremento del nivel de feromona.

Cada solución T, se construye iniciando la hormiga en el nodo fuente el grupo multicast, y a cada paso elige otro nodo aún no visitado, según una regla pseudo-aleatoria [10]. Este proceso sigue hasta alcanzar todos los nodos destinos multicast. En la Figura 3, se presenta el procedimiento para hallar una solución T.

```
Inicio (Construir Árbol)
   T = ∅
   N = "s"
   Dr = ∅
   Repetir hasta que (N = ∅ o Dr = Nr)
      i = nodo elegido aleatoriamente de N
      Construir conjunto Ni
      Si (Ni = ∅) entonces
         /* Se elimina nodo i sin vecinos factibles */
         N = N – i
      Sino
         j = nodo elegido de Ni, con regla pseudo-aleatoria.
         T = T ∪ (i,j)
         N = N ∪ j
         Si (j ∈ Nr) entonces
            Dr = Dr ∪ j
         Fin Si
      Fin Si
      /* Se actualiza en línea τij */
      τij =(1-φ).τ0 +φ.τ0
   Fin Repetir
   /* Se elimina los enlaces no utilizados */
   Podar Árbol T
Fin
```

Figura 3. Procedimiento Construir Árbol (*Pseudocódigo 3*)

[2] La normalización consiste en dividir cada función objetivo por su máximo valor escogido a priori.

donde:

N	Lista de nodos de partida correspondiente al árbol en construcción T.
N_i	Lista de nodos vecinos factibles al nodo i.
N_r	Conjunto de nodos destinos de un grupo multicast.
D_r	Conjunto de nodos destinos ya alcanzados.
φ	Parámetro de decremento del nivel de feromona.

4 Multiobjective Multicast Algorithm

El *Multiobjective Multicast Algorithm* (MMA) propuesto en [1] por J. Crichigno y B. Barán, se basa en el *Strength Pareto Evolutionary Algorithm* (SPEA) [14]. El algoritmo propuesto mantiene una población P y un conjunto externo de soluciones (individuos) Pareto P_{nd}. El procedimiento general se inicia con una población de individuos aleatorios y se evolucionan estos individuos a soluciones óptimas, las cuales son incluidas en P_{nd}. La Figura 4 muestra el algoritmo propuesto [1].

Construcción de tablas de enrutamiento.

Sea:

$N_r = \{n_1, n_2, ..., n_{|N|}\}$ Conjunto de nodos destinos n_i del grupo multicast.

```
Inicio (MMA)
   Leer grupo multicast (s,Nr) y demanda de tráfico ϕ
   Construir tablas de enrutamiento
   Inicializar P y Pnd
   Repetir hasta que algún criterio de parada se cumpla
      Descartar soluciones idénticas
      Evaluar individuos de P
      Actualizar conjunto no-dominado de Pnd
      Calcular fitness
      Seleccionar soluciones
      Aplicar Cruce y Mutación
   Fin Repetir
Fin
```

Figura 4. Procedimiento general del MMA (*Pseudocódigo 4*)

Para cada nodo destino $n_i \in N_r$, una tabla de enrutamiento es construida. Esta tabla consiste de R caminos cortos y R caminos de bajos costos. Nótese que R es un parámetro del algoritmo.

Cada cromosoma (o solución T) es representado por una cadena de longitud $|N_r|$, en el cual el elemento (gen) g_i representa el camino entre el nodo fuente "s" y un nodo destino n_i *del grupo multicast*. La relación entre un cromosoma, gene y tablas de enrutamiento es mostrado en la Figura 5 (gráfico tomado de [1]).

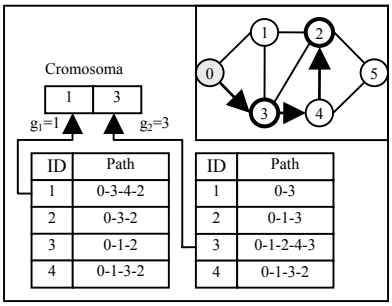

Figura 5. Relación entre cromosoma gen y tablas de enrutamientos.

Nótese que en cada generación del procedimiento general del *MMA* existen seis sub-procedimientos que evolucionan las poblaciones P y P_{nd}, las cuales se explican brevemente a continuación.

Descartar soluciones idénticas. En cada nueva población P, pueden generarse soluciones idénticas. La aplicación del operador de cruce en dos soluciones idénticas genera la misma solución, por lo que la habilidad de buscar nuevas soluciones queda reducida.

En la *Evaluación de individuos P,* se calcula el vector objetivo de cada cromosoma de *P*, usando las funciones de costo, retardo medio y máximo, definidos en la sección 2.
En la *Actualización de individuos no-dominados de* P_{nd}, cada solución no-dominada de *P* es comparada con los individuos de P_{nd}. Aquellas soluciones de *P* no-dominadas por alguna solución de P_{nd}, son copiadas en P_{nd}; por lo tanto, si alguna solución de P_{nd} es dominado por alguna solución de *P*, ésta es removida de P_{nd}.
El *Calculo de fitness* de cada individuo es obtenido usando el procedimiento del *SPEA* [14], y el operador de *Selección de soluciones* es aplicado sobre la unión del conjunto de *P* y P_{nd}, para seleccionar buenos individuos y generar una nueva población *P*.
Al final de cada generación los operadores de *Cruce* y *Mutación* son aplicados a cada par de individuos. Todo el conjunto de procedimientos anteriores explicados se aplica hasta que algún criterio de parada se cumpla.

5 Resultados Experimentales

Las pruebas experimentales, se realizaron con la topología de red de la *NTT* (*Nipon Telegraph and Telephone, Co*). Esta red consta de 55 nodos y 144 enlaces direccionados [11]. Los números sobre cada enlace representan el retardo del mismo (ver Figura 6).

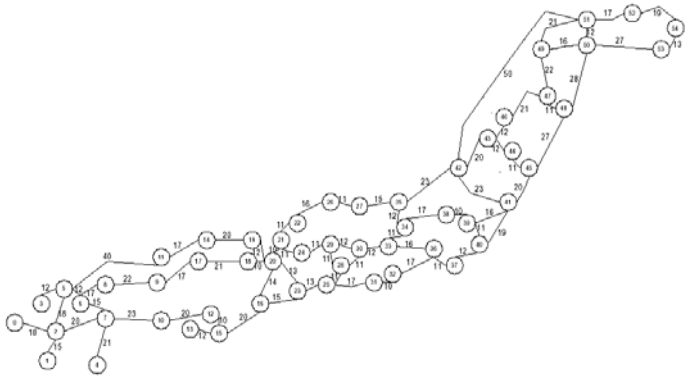

Figura 6. Red de la Nipon Telegraph and Telephone, Co

Se realizó un conjunto de 4 pruebas, con 9,14,19 y 24 destinos, conforme se muestra en la Tabla 1. Cada prueba consiste en 3 corridas con tiempos de ejecución de: 40, 160 y 320 segundos.

Tabla 1. Grupos multicast utilizados para las pruebas

	(*s*)Nodo Fuente	(N_r) Nodos destinos	$\|N_r\|$
Grupo 1	{5}	{0, 1, 8, 10, 22, 32, 38, 43, 53}	9
Grupo 2	{4}	{0, 1, 3, 5, 9, 10, 12, 23, 25, 34, 37, 41, 46, 52}	14
Grupo 3	{4}	{0, 1, 3, 5, 6, 9, 10, 12, 17, 22, 23, 25, 34, 37, 41, 46, 47, 52, 54}	19
Grupo 4	{4}	{0, 1, 3, 5, 6, 9, 10, 11, 12, 17, 19, 21, 22, 23, 25, 33, 34, 37, 41, 44, 46, 47, 52, 54}	24

Las ejecuciones del algoritmo se realizaron en una PC con procesador AMD-K6 3D de 350 MHz, 120 MB de memoria RAM, usando un compilador Borland C++ 5.02.

5.1 Procedimientos de comparación

El procedimiento de comparación utilizado para cada grupo multicast fue el siguiente:

a. Cada uno de los 2 algoritmos se corrió cinco veces, para calcular los promedio.
b. Se obtuvo para cada algoritmo el conjunto de soluciones no dominadas: $Y_1, Y_2, \ldots, Y_5$, para cada una de las cinco corridas.
c. Se creó para cada algoritmo una superpoblación, donde $Y_T = \bigcup_{i=1}^{5} Y_i$.

d. De cada superpoblación Y_T se extrajeron las soluciones no dominadas, formando así el frente Pareto calculado por cada algoritmo, como sigue:
   Y_{MOACS} (frente Pareto obtenido con las 5 corridas del *MOACS*)
   Y_{MMA} (frente Pareto obtenido con las 5 corridas del *MMA*)

e. Se obtuvo el conjunto $\hat{Y}$ de soluciones encontradas como sigue:

$$\hat{Y} = Y_{MOACS} \vee Y_{MMA}$$

f. Del conjunto $\hat{Y}$ se eliminan las soluciones dominadas, y así se forma una aproximación del Y_{true}, que en estas pruebas es llamado Y_{apr}[3]. En la Tabla 2 se presenta la cantidad de soluciones $T \in Y_{apr}$ halladas para cada grupo multicast.

Tabla 2. Cantidad de soluciones óptimas halladas para cada grupo multicast

	Grupo 1	Grupo 2	Grupo 3	Grupo 4
$\lvert Y_{apr} \rvert$	9	18	24	18

donde :

$\lvert Y_{apr} \rvert$ representa la cardinalidad de Y_{apr}.

5.2 Resultados Obtenidos

Las tablas impares de cada prueba presentan la cantidad de soluciones (promedio de las 5 corridas) de cada algoritmo, que se encuentran en Y_{apr} denotado en tablas como [$\in Y_{apr}$], las que son dominadas por Y_{apr} como [$Y_{apr}\succ$]. El número de soluciones encontradas [$|Y_{alg}|$] y [%($\in Y_{apr}$)] representa el porcentaje de soluciones pertenecientes a Y_{apr}.
Los ítems siguientes dan un ejemplo de lo arriba presentado para la Tabla 3, considerando el algoritmo *MMA*:

a. La casilla que pertenece a la fila de Y_{MMA} y la columna de [$\in Y_{apr}$] indica que 5.8 soluciones halladas en promedio por *MMA*, pertenecen al Y_{apr} del grupo 1.
b. La casilla que pertenece a la fila de Y_{MMA} y la columna de [$Y_{apr}\succ$] indica que 0 soluciones halladas en promedio por *MMA* son dominadas por el Y_{apr} del grupo 1.
c. La casilla que pertenece a la fila de Y_{MMA} y la columna de [$|Y_{alg}|$] sencillamente indica la cantidad de soluciones que en promedio fueron halladas por *MMA*, que en este caso es 5.8.
d. La casilla que pertenece a la fila de Y_{MMA} y la columna de [%($\in Y_{apr}$)] indica el porcentaje de soluciones halladas por *MMA* que pertenecen al Y_{apr} del grupo 1, que en esta prueba es 64%. Donde [%($\in Y_{apr}$)] se calcula como el cociente de las soluciones [$\in Y_{apr}$] y la cantidad de soluciones de Y_{apr}, obtenidas de la Tabla 2. En este caso es 100•5.8/9=64.

Las tablas pares de cada experimento presentan la cobertura entre algoritmos [13]. Entonces, dentro de este contexto, la tabla de cobertura representa en promedio la cantidad de soluciones $q \succ p$, donde $p \in Y_i$ mientras $q \in Y_j$, y sea a_{ij} un elemento de esta tabla, tal que i representa a las filas y j representa a las columnas.

Experimento 1. Resultados experimentales para el grupo multicast 1

Tabla 3. Comparación de las soluciones con Y_{apr}

	Tiempo de corrida = 40 seg.			
	$\in Y_{apr}$	$Y_{apr}\succ$	$\lvert Y_{alg} \rvert$	%($\in Y_{apr}$)
Y_{MOACS}	8.8	0	8.8	98%
Y_{MMA}	5.8	0	5.8	64%

Tabla 4. Cobertura entre algoritmos

	Y_j	
Y_i	Y_{MOACS}	Y_{MMA}
Y_{MOACS}		0
Y_{MMA}	0	

Tabla 5. Comparación de las soluciones con Y_{ap}.

	Tiempo de corrida = 160 seg.			
	$\in Y_{apr}$	$Y_{apr}\succ$	$\lvert Y_{alg} \rvert$	%($\in Y_{apr}$)
Y_{MOACS}	9	0	9	100%
Y_{MMA}	5.2	0	5.2	57%

Tabla 6. Cobertura entre algoritmos

	Y_j	
Y_i	Y_{MOACS}	Y_{MMA}
Y_{MOACS}		0
Y_{MMA}	0	

[3] Nótese que para fines prácticos $Y_{apr} \approx Y_{true}$, es decir Y_{apr} es una excelente aproximación de Y_{true}, sino idéntico.

Tabla 7. Comparación de las soluciones con Y_{apr}

	Tiempo de corrida = 320 seg.			
	$\in Y_{apr}$	$Y_{apr} \succ$	$\lvert Y_{alg} \rvert$	$\%(\in Y_{apr})$
Y_{MOACS}	9	0	9	100%
Y_{MMA}	5.8	0	5.8	64%

Tabla 8. Cobertura entre algoritmos

	Y_j	
Y_i	Y_{MOACS}	Y_{MMA}
Y_{MOACS}		0
Y_{MMA}	0	

- Se observa en las Tablas 3, 5 y 7 que el *MOACS* encuentra prácticamente todas las soluciones de Y_{apr}, con una pequeña diferencia del 2% para 40 segundos de corrida, superando ampliamente al *MMA*.
- Por otra parte, nótese que todas las soluciones halladas por ambos algoritmos pertenecen a Y_{apr} (columnas [$\in Y_{apr}$] y [$Y_{apr} \succ$]), y esto se ve reflejado en las Tablas 4, 6 y 8 donde las coberturas dan como resultado cero.

Experimento 2. Resultados experimentales para el grupo multicast 2

Tabla 9. Comparación de las soluciones con Y_{apr}

	Tiempo de corrida = 40 seg.			
	$\in Y_{apr}$	$Y_{apr} \succ$	$\lvert Y_{alg} \rvert$	$\%(\in Y_{apr})$
Y_{MOACS}	13.8	6.4	20.2	76%
Y_{MMA}	8.4	3	11.4	46%

Tabla 10. Cobertura entre algoritmos

	$\boldsymbol{Y_j}$	
Y_i	Y_{MOACS}	Y_{MMA}
Y_{MOACS}		0
Y_{MMA}	0	

Tabla 11. Comparación de las soluciones con Y_{ap}.

	Tiempo de corrida = 160 seg.			
	$\in Y_{apr}$	$Y_{apr} \succ$	$\lvert Y_{alg} \rvert$	$\%(\in Y_{apr})$
Y_{MOACS}	16	5.6	21.6	89%
Y_{MMA}	11.6	3.8	15.4	64%

Tabla 12. Cobertura entre algoritmos

	$\boldsymbol{Y_j}$	
Y_i	Y_{MOACS}	Y_{MMA}
Y_{MOACS}		0.6
Y_{MMA}	0	

Tabla 13. Comparación de las soluciones con Y_{apr}

	Tiempo de corrida = 320 seg.			
	$\in Y_{apr}$	$Y_{apr} \succ$	$\lvert Y_{alg} \rvert$	$\%(\in Y_{apr})$
Y_{MOACS}	16	6	22	89%
Y_{MMA}	13.6	4	16.6	75%

Tabla 14. Cobertura entre algoritmos

	Y_j	
Y_i	Y_{MOACS}	Y_{MMA}
Y_{MOACS}		0.4
Y_{MMA}	0	

- Observe que también en estos experimentos el *MOACS* supera ampliamente al *MMA*. Obteniendo mayor cantidad de soluciones pertenecientes al Y_{apr}, como mayor cantidad de soluciones calculadas (Tablas 9, 11 y 13).
- Para tiempos de cómputos medios y altos las soluciones del *MOACS* dominan a algunas soluciones del *MMA* (Tablas 12 y 14).

Experimento 3. Resultados experimentales para el grupo multicast 3

Tabla 15. Comparación de las soluciones con Y_{apr}

	Tiempo de corrida = 40 seg.			
	$\in Y_{apr}$	$Y_{apr} \succ$	$\lvert Y_{alg} \rvert$	$\%(\in Y_{apr})$
Y_{MOACS}	13.6	1.4	15	56%
Y_{MMA}	10	0.8	10.4	41%

Tabla 16. Cobertura entre algoritmos

	Y_j	
Y_i	Y_{MOACS}	YP_{MMA}
Y_{MOACS}		0.8
Y_{MMA}	0.2	

Tabla 17. Comparación de las soluciones con Y_{apr}

	Tiempo de corrida = 160 seg.			
	$\in Y_{apr}$	$Y_{apr} \succ$	$\lvert Y_{alg} \rvert$	$\%(\in Y_{apr})$
Y_{MOAC}	17.6	0.6	18.2	73%
Y_{MMA}	11	0	11	45%

Tabla 18. Cobertura entre algoritmos

	Y_j	
Y_i	Y_{MOACS}	Y_{MMA}
Y_{MOACS}		0
Y_{MMA}	0	

Tabla 19. Comparación de las soluciones con Y_{apr}

	Tiempo de corrida = 320 seg.			
	$\in Y_{apr}$	$Y_{apr} \succ$	$\lvert Y_{alg} \rvert$	$\%(\in Y_{apr})$
Y_{MOACS}	19.4	0.6	19.8	80%
Y_{MMA}	11.6	0.4	11.1	47%

Tabla 20. Cobertura entre algoritmos

	Y_j	
Y_i	Y_{MOACS}	Y_{MMA}
Y_{MOACS}		0.4
Y_{MMA}	0	

- En estos experimentos el *MOACS* alcanza en promedio un mayor porcentaje de soluciones pertenecientes a Y_{apr} que las alcanzadas por *MMA* (Tablas 15, 17 y 19).
- Lo anteriormente dicho implica claramente que el *MOACS* halla mayor cantidad de soluciones.
- Con respecto a la cobertura entre algoritmos, el *MOACS* en promedio domina más soluciones pertenecientes al *MMA* (Tablas 16 y 20).

Experimento 4. Resultados experimentales para el grupo multicast 4

Tabla 21. Comparación de las soluciones con Y_{apr}

	Tiempo de corrida = 40 seg.			
	$\in Y_{apr}$	$Y_{apr} \succ$	$\lvert Y_{alg} \rvert$	$\%(\in Y_{apr})$
Y_{MOACS}	4	7.6	11.6	22%
Y_{MMA}	2.6	0.6	3.2	14%

Tabla 22. Cobertura entre algoritmos

	Y_j	
Y_i	Y_{MOACS}	Y_{MMA}
Y_{MOACS}		0.2
Y_{MMA}	1.6	

Tabla 23. Comparación de las soluciones con Y_{apr}

	Tiempo de corrida = 160 seg.			
	$\in Y_{apr}$	$Y_{apr} \succ$	$\lvert Y_{alg} \rvert$	$\%(\in Y_{apr})$
Y_{MOACS}	12.2	0.6	14.8	67%
Y_{MMA}	4.2	0.6	4.8	23%

Tabla 24. Cobertura entre algoritmos

	Y_j	
Y_i	Y_{MOAC}	Y_{MMA}
Y_{MOAC}		0.4
Y_{MMA}	0.2	

Tabla 25. Comparación de las soluciones con Y_{apr}

	Tiempo de corrida = 320 seg.			
	$\in Y_{apr}$	$Y_{apr} \succ$	$\lvert Y_{alg} \rvert$	$\%(\in Y_{apr})$
Y_{MOACS}	14	2.6	16.6	77%
Y_{MMA}	4.4	1.2	5.6	24%

Tabla 26. Cobertura entre algoritmos

	Y_j	
Y_i	Y_{MOACS}	Y_{MMA}
Y_{MOAC}		0.8
Y_{MMA}	0.2	

- En este último experimento que se caracteriza por tener el grupo multicast con mayor cantidad de destinos y en el cual el *MOACS* también demostró ser mejor que el *MMA*. El *MOACS* obtuvo mayor cantidad de soluciones que pertenecen a Y_{apr} en todos los tiempos de corrida.
- Nótese también que las soluciones del *MOACS* en promedio dominan más soluciones del *MMA* para 160 y 320 seg. de corridas (Tabla 24 y 26), pero en 40 seg. de corrida el *MMA* tuvo mayor dominancia sobre soluciones del *MOACS*.

Tabla 27. Promedios generales de los experimentos

Tabla (a). Comparación de las soluciones con Y_{apr}

	$\in Y_{apr}$	$Y_{apr} \succ$	$\lvert Y_{alg} \rvert$	$\%(\in Y_{apr})$
Y_{MOACS}	14.1	3.5	17.8	69.9%
Y_{MMA}	8.6	1.6	9.9	42.1%

Tabla (b). Cobertura entre algoritmos

	Y_j	
Y_i	Y_{MOACS}	Y_{MMA}
Y_{MOAC}		0.4
Y_{MMA}	0.2	

- Se puede notar que el *MOACS* en promedio es superior al *MMA*, conforme a las tablas de promedios generales Tabla 27.

6 Conclusiones

Teniendo en cuenta los resultados de las experiencias llevadas a cabo, las mismas nos indican según las conclusiones anteriores y las tablas de promedios generales que el *MOACS* es el mejor con 69,9% de soluciones que pertenecen a Y_{apr} encontradas, muy superior al *MMA*. Además el Y_{MOACS} tiene mayor dominancia sobre soluciones de Y_{MMA}.
Estos resultados de esta manera muestran que los *Algoritmos basados en Colonia de Hormigas* son prometedores para la resolución del problema de Ingeniería de Tráfico Multicast.
En trabajos futuros, se incluirán otras funciones objetivos como la utilización de la carga en los enlaces con restricciones de carga, y se espera realizar simulaciones con variaciones dinámicas del tráfico. También serán utilizados otros algoritmos *ACO* para estas pruebas.

Referencias

[1] J. Crichigno y B. Barán, "Multiobjective Multicast Routing Algorithm". IEEE ICT'2004, Ceará, Brasil, 2004.
[2] J. Crichigno y B. Barán, "A Multicast Routing Algorithm Using Multiobjective Optimization", IEEE ICT'2004, Ceará, Brasil, 2004.
[3] J. Crichigno y B. Barán, "Multiobjective Multicast Routing Algorithm for Traffic Engineering". IEEE ICCCN'2004, Chicago,US, 2004.
[4] M. Dorigo y G. Di Caro. "The Ant Colony Optimization meta-heuristic". En D. Corne, M. Dorigo, y F. Glover, editores, New Ideas in Optimization, páginas 11-32. McGraw Hill, London, UK, 1999.
[5] M. Dorigo y L. M. Gambardella. "Ant Colony System: A cooperative learning approach to the traveling salesman problem". IEEE Transactions on Evolutionary Computation, 1: 1, páginas 53-66, 1997.
[6] C. García-Martínez, O. Cordón y F. Herrera, "An Empirical Analysis of Multiple Objective Ant Colony Optimization Algorithms for the Bi-criteria 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. Tomo 3172 de LNCS. Springer-Verlag, Bruselas, septiembre 2004.
[7] J. Gu, C. Chu, X. Hou y Q. Gu. "A heuristic ant algorithm for solving QoS multicast routing problem". Evolutionary Computation, 2002. CEC '02. Vol. 2, páginas 1630-1635.
[8] Y. Liu y J. Wu. "The degree-constrained multicasting algorithm using ant algorithm". IEEE 10th International Conference on Telecommunications". 2003.
[9] P. Ravikumar, y R. Bajpai, "Source-based delay bounded multicasting in multimedia networks", Computer Communications, Vol. 21, 1998, páginas 126-132.
[10] M. Schaerer y B. Barán. **"**A Multiobjective Ant Colony System For Vehicle Routing Problem With Time Windows**",** IASTED International Conference on Applied Informatics, Innsbruck, Austria, 2003.
[11] R. Sosa y B. Barán, "A New Approach for Antnet Routing", (ICCCN'2000), USA, 2000.
[12] A. Tanenbaum, Computer Networks, Prentice Hall 4º Edition, 2003.
[13] F. Zitzler, K. Deb y L. Thiele. Comparison of Multiobjective Evolutionary Algoritms. Empirical Results. Evolutionary Computation, 8(2): 173-195, Summer 2000.
[14] E. Zitzler, y L. Thiele, "Multiobjective Evolutionary Algorithms: A comparative Case Study and the Strength Pareto Approach", IEEE Trans. Evolutionary Computation, Vol. 3, No. 4, 1999, páginas 257-271.
[15] D. A. Van Veldhuizen. "Multiobjective Evolutionary Algorithms: Clasifications, Analyses and New Innovations". Ph. D. thesis Air Force Institute of Technology, 1999.