Implementación y comparación de GA y PSO para el problema de Enrutamiento Unicast.

Germán R. Hüttemann
Universidad Nacional de Asunción, Facultad Politécnica,
Asunción, Paraguay
ghuttemann@gmail.com

y

Marcelo D. Rodas
Universidad Nacional de Asunción, Facultad Politécnica,
Asunción, Paraguay
rodas.marcelo@gmail.com

Abstract

The Genetics Algorithms (GA) like the Particles Swarm Optimization (PSO), are frequently used with satisfactory results to solve several routing problems. Specifically, this document presents two solutions to the problem of routing a set of unicast demands in a telecommunication network, using for that matter, GA and PSO. The results obtained show that PSO mostly obtain better solutions, without reaching the best solutions. However, those best results obtained, could enhance if the time established in the tests raise according to the raising of the amount of unicast demands. Also, with a good manipulation of algorithm's parameters and a good analysis of the algorithms main functions, it could be obtained better final solutions.

Keywords: Genetic Algorithms, Particles Swarm Optimization, Mono-Objective Optimization, Unicast demands.

Resumen

Tanto los Algoritmos Genéticos (GA) como los algoritmos de Optimización basada en Enjambre de Partículas (PSO), son frecuentemente utilizados satisfactoriamente, para la solución de distintos problemas de enrutamiento. Particularmente, este documento presenta dos soluciones al problema de tener que enrutar un conjunto de demandas unicast en una red de comunicación, utilizando para ello, GA y PSO. Los resultados obtenidos muestran que el PSO, en su mayoría, obtuvo mejores soluciones, sin necesariamente llegar a óptimos globales. Sin embargo, se concluyó que los mejores resultados obtenidos mejorarían si el tiempo de prueba de estos algoritmos aumentara a medida que se aumentase la cantidad de demandas solicitadas. Además, con un buen manejo de los parámetros utilizados en los algoritmos, y un buen análisis de las funciones principales de estos, se podrán llegar a obtener aún mejores resultados finales.

Palabras clave: Algoritmo Genético, Optimización basada en Enjambre de Partículas, Optimización Mono-objetivo, Demandas Unicast.

1 Introducción

Este trabajo plantea la solución del problema de tener un conjunto de demandas unicast para ser transmitidas en una red de comunicaciones. Se plantean 2 soluciones, una utilizando GA y la otra utilizando PSO. Además, se soluciona el problema como mono-objetivo.

El planteamiento de la solución, utilizando GA, puede ser considerado como una adaptación del algoritmo MMA1 realizado por Jorge Crichigno [1], en cuanto a la representación y el algoritmo en sí. Por otro lado, el planteamiento de la solución, utilizando PSO, puede ser considerado como una adaptación de la formulación tradicional del PSO.

El trabajo esta organizado de la siguiente manera: en la sección 2 se describe el problema a solucionar. En la sección 3 se presenta el modelo matemático para la solución. Luego, en la sección 4 se plantea la representación general de las soluciones, especificando como se interpreta tanto para GA como para PSO. Seguidamente, en la sección 5 se explica como se procedió en la evaluación de las soluciones encontradas. Luego, en las secciones 6 y 7 se detallan ambas soluciones propuestas. En la sección 8 se presentan los resultados experimentales. Finalmente, en las secciones 9 y 10 se presentan las conclusiones obtenidas y los trabajos futuros, respectivamente.

2 Descripción del Problema

El problema consiste en la selección de las mejores rutas para un conjunto de demandas unicast (CDU), pudiendo tener muchas demandas con orígenes distintos y también destinos distintos. Se tiene como único objetivo el de minimizar el costo total de las rutas a tomar para satisfacer las demandas, pero además, cumpliendo la restricción de que el ancho de banda de los enlaces que componen las rutas no sean rebasados.

Las condiciones a tener en cuenta para la evaluación de las rutas a ser seleccionadas para cada demanda son el costo de la ruta, y la capacidad máxima de los enlaces de ésta.

3 Modelo Matemático

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

- R_+ conjunto de los números reales positivos.
- $(a, b) \in E$ enlace entre los nodos a y b, donde $a, b \in V$.
- $z_{ab} \in R_+$ capacidad del enlace (a, b), en Mbps.
- $c_{ab} \in R_+$ costo del enlace (a, b).
- T_{sd} conjunto de rutas con origen s y destino d, donde $s, d \in V$ y $s \neq d$.
- $R_{sd}^i \in T_{sd}$ i-ésima ruta de origen s y destino d, donde $s, d \in V$ y $s \neq d$.
- H_{sd} demanda de origen s y destino d, donde $s, d \in V$ y $s \neq d$.
- $k_{sd}^i \in R_+$ suma total de costos (c_{ab}) de los enlaces en la ruta R_{sd}^i.
- $\phi_{sd} \in R_+$ ancho de banda requerido por la demanda H_{sd}, en Mbps.
- $F(a, b) \in R_+$ suma total de anchos de banda de las demandas (ϕ_{sd}) que pasan por el enlace (a, b).

Usando las notaciones definidas arriba, el problema planteado anteriormente se puede realizar minimizando la siguiente función objetivo:

Costo de rutas utilizadas para satisfacer las demandas unicast:

$$C = \sum k_{sd}^i \quad (1)$$

Sujeta a la restricción de capacidad en los enlaces:

$$F(a,b) \leq z_{ab}; \forall (a,b) \in E \quad (2)$$

4 Representación General de las Soluciones

Para la solución del problema, planteamos una estructura de datos que es un simple arreglo de una dimensión, tal como se indica en la Figura 1. La idea general es que este arreglo represente al conjunto de rutas que satisfagan el conjunto de las *N* demandas unicast. De esta forma, cada celda del arreglo representa la ruta que satisface a alguna de las demandas solicitadas. Como se podrá imaginar, para esta representación de la solución, inicialmente es necesaria la generación de las tablas de rutas para cada una de las demandas. Luego, cada celda del arreglo hace referencia a alguna de las rutas de la tabla correspondiente, a través de un entero positivo que representa el índice de dicha ruta dentro de la tabla.

Esta representación general es utilizada para ambas soluciones presentadas, solo cambian conceptualmente cada parte de la estructura de datos. Para la solución basada en GA se lo denomina Cromosoma y para la solución basada en PSO se lo denomina Partícula.

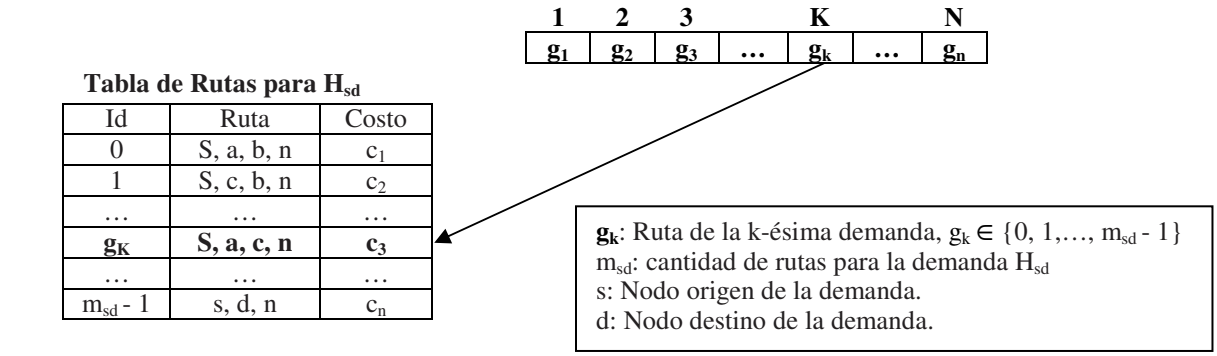
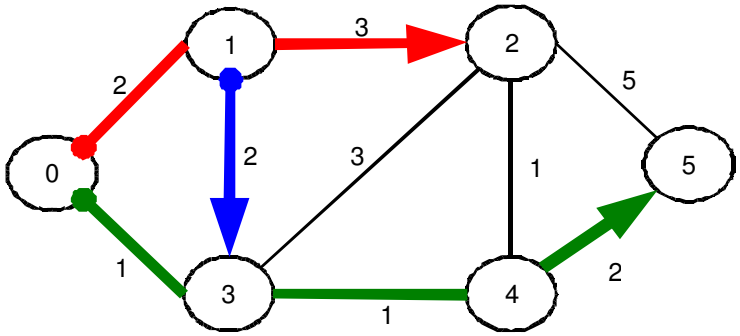


Figura 1. Representación Gráfica del cromosoma.



Demandas Unicast		
Origen	Destino	Ancho de banda (Mbps)
0	2	0.5
0	5	0.6
1	3	1.2

Figura 2. Grafo de una pequeña red de comunicación y la tabla de demandas unicast.

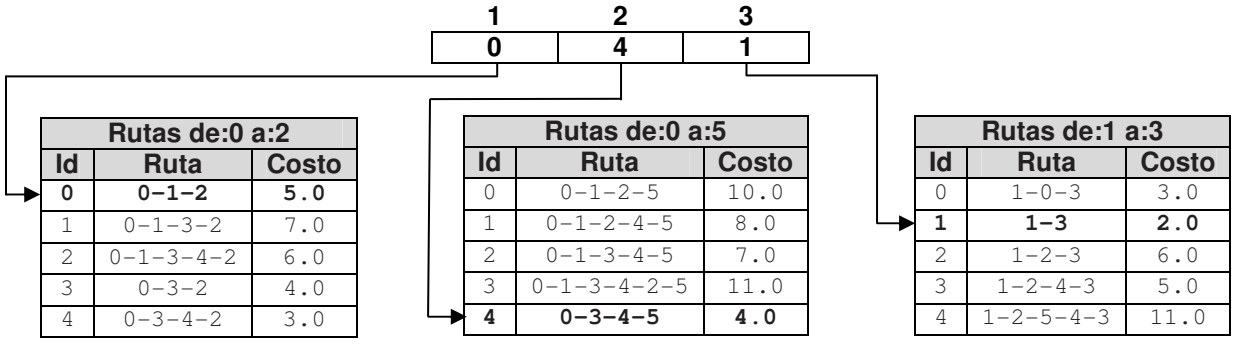


Figura 3. Representación para el Ejemplo de la Figura 2.

4.1 Cromosoma

Cuando la estructura de datos planteada se utiliza para la solución con GA se la denomina Cromosoma. Además, cada elemento de la estructura se denomina gen, siendo la tabla correspondiente la que delimita sus valores posibles (genotipo) y estos valores representan a la ruta (con su costo) en la tabla correspondiente (fenotipo).

Como ejemplo, se puede ver la Figura 3, donde el cromosoma tiene 3 genes y cuyos genotipos se definen por los índices a las posibles rutas en la tabla de rutas de sus correspondientes demandas. Además, como se puede observar en la Figura 2 y Figura 3, el gen 1 representa a la demanda de origen 0 y destino 2 y así sucesivamente.

Es importante resaltar que en el grafo de la Figura 2 el ancho de banda de de cada enlace es de 1.5 Mbps. También se puede notar que la mejor respuesta para el gen 1 sería la ruta 4 de la tabla correspondiente, dando en ese caso la mejor solución para las demandas planteadas. Obsérvese que con la solución óptima hay solapamiento de enlaces para las dos primeras demandas, sin embargo, cada enlace es capaz de soportar ambas demandas simultáneamente, cosa que no siempre podría ocurrir.

4.2 Partícula

Cuando la estructura de datos planteada se utiliza para la solución con PSO se la denomina Partícula. Además, cada elemento de la estructura se refiere a una de las coordenadas que determinan la posición de la partícula en el espacio de solución, siendo la tabla correspondiente la que delimita sus valores posibles (rango de la coordenada) y estos valores representan a la ruta asociada, junto a su respectivo costo.

Cada Partícula se representa por su posición en el espacio de solución, siendo la dimensión de éste igual a la cantidad de demandas solicitadas. Por ejemplo, en la Figura 2, el espacio es de 3 dimensiones ya que hay 3 demandas.

5 Evaluación de las Soluciones

El primer control realizado sobre los valores de entrada del algoritmo es cuando se cargan los valores para las tablas de rutas correspondientes a las demandas, donde solo se cargan rutas que de forma independiente cumplen con la restricción de capacidad de la demanda a la que está relacionada. De esta forma, se evita utilizar rutas que, sin importar con que otras rutas se seleccionen, van a dar una respuesta inválida.

Otro control realizado es el control de validez de una solución, que se realiza debido a la restricción de capacidad de los enlaces del grafo. Este problema se da debido a que dos o más rutas pueden utilizar los mismos enlaces, causando que la capacidad de alguno de ellos no sea suficiente para satisfacer a todas las rutas que pasan por él.

La evaluación de las soluciones obtenidas, o cálculo de fitness, se realiza en dos fases. Primero, se controla lo indicado en el párrafo anterior. Luego, si la respuesta es inválida se responde con el número de enlaces distintos que hacen inválida a la solución, cambiado de signo (signo negativo); en caso contrario, se suman los costos de cada enlace de la solución y se responde con la inversa del costo total. En segundo lugar, se suma la respuesta calculada anteriormente con el número total de arista del grafo.

De esta forma la peor respuesta (que daría una solución inválida) tendrá fitness 0, es decir, la solución que abarca a todos los enlaces del grafo pero que no puede cumplir con el requerimiento de ancho de banda de la demandas solicitadas. La mejor solución inválida tendría un fitness igual a la cantidad total de aristas del grafo menos 1. Por otro lado, las soluciones válidas van desde el número total de aristas del grafo (solución con costo igual a infinito), hasta infinito (solución con costo igual a cero).

Formalmente, dado $|E|$, cantidad de aristas del grafo *G = (V, E)*, la ecuación 3 representa la fórmula de evaluación de una solución x_1 válida, y la ecuación 4 representa la fórmula de evaluación de una solución x_2 inválida.

$$f_{val}(x_1) = |E| + \frac{1}{C}, \text{ donde } C = \sum k_{sd}^{i} \tag{3}$$

$$f_{inval}(x_2) = |E| + q, \text{ donde } q = -\sum_{F(a,b)>z_{ab}} (a,b) \tag{4}$$

Los rangos que pueden tomar cada una de las fórmulas de las ecuaciones 3 y 4 son, respectivamente:

$$f_{val}(x_1) \in \left[|E|, \infty\right]$$

Los fitness que están en este rango son los que satisfacen la restricción expuesta en la ecuación 2

$$f_{inval}(x_2) \in \left[0, |E| - 1\right]$$

Es importante resaltar, que el fitness es un valor que indica que tan buena es la solución, y por ende, mientras más grande sea, mejor será la solución. Además, los valores extremos del fitness para las respuestas válidas son expresados de forma matemática, ya que encontrar un costo cero o costo infinito en la práctica sería muy improbable. También, es importante no confundir el fitness con el costo, ya que el costo se refiere a la suma total de los costos de ruta de la solución y el fitness es una medida de valorización de cada solución encontrada.

6 Detalles de la implementación con GA

Como mencionamos en la introducción, el planteamiento de la solución utilizando GA, puede ser considerado como una adaptación del algoritmo MMA1 [1]. La principal diferencia se debe a la interpretación del esquema de representación, ya que MMA1 se plantea para un problema multicast de *n* demandas con origen único y en nuestro problema es de *n* demandas, cada una con un origen y destino.

Además, los operadores de cruce, mutación y reemplazo son distintos, coincidiendo sí en la selección, para la que se utiliza el torneo binario con reposición. El cruce se realiza con cruce uniforme [3] y la mutación se realiza con probabilidad P_{mut} para cada individuo, como una variación de la mutación uniforme [3], en la que en vez de mutar a un solo gen, se muta a todos ellos. El reemplazo, en nuestro caso, al ser un problema mono-objetivo, lo realizamos reemplazando la población actual por toda la nueva generación de individuos.

Para entender el cruce uniforme, se puede ver el ejemplo de la Figura 4(a). Para realizar el cruce lo que se hace es seleccionar dos padres de la población. Luego, por cada gen del *Padre 1* se selecciona randómicamente entre 2 valores (Ej.: 0 y 1) si sale el primer valor (0), el gen correspondiente del *Padre 1* es para *H1* (hijo 1) y el gen correspondiente del *Padre 2* es para *H2* (hijo 2); en caso contrario, el gen correspondiente del *Padre 1* es para *H2* y el gen correspondiente del *Padre 2* es para *H1*.

Para entender la mutación uniforme, se puede ver la Figura 4(b). En este ejemplo para realizar la mutación primeramente solo se realiza la mutación con una probabilidad P_{mut}. Luego, una vez que se decide que se mutará al individuo, para cada gen del mismo se elige randómicamente entre los valores permitidos por la tabla de rutas correspondiente y ese nuevo valor es asignado al gen. Debido a que no se realiza un control de si el nuevo valor elegido es igual al anterior, uno o más genes pueden quedar inalterados, como es el caso del tercer gen del ejemplo.

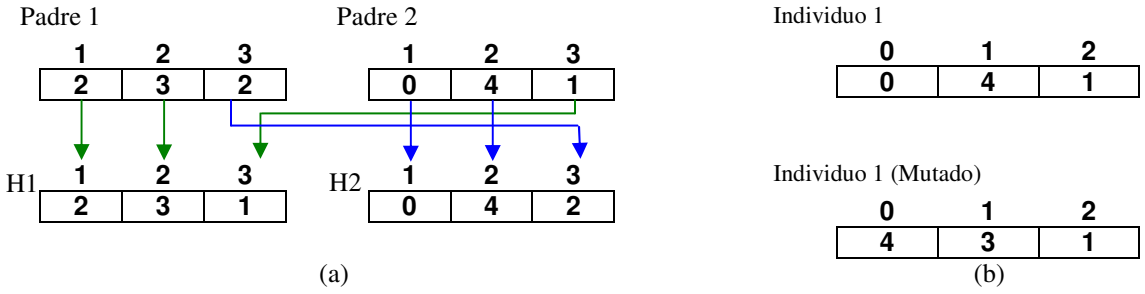


Figura 4. (a) Ejemplo de Cruce uniforme. (b) Ejemplo de Mutación uniforme.

Una de las operaciones adicionales es la función de reinicio, que consiste en reinicializar la población si se da el caso de que exista un cierto porcentaje P_{inv} de individuos inválidos en la población. Para las pruebas realizadas, los valores de P_{mut}, P_{inv} y del tamaño de la población fueron, respectivamente, 0.20, 80% y 32 individuos. El pseudocódigo del algoritmo GA se presenta a continuación:

```
INICIO
  Leer demandas y rutas
  t = 0
  P(t) = Inicializar población
  Descartar individuos iguales de P(t)
  Evaluar fitness de individuos de P(t)
  GBest = Guardar mejor individuo de P(t)

  MIENTRAS no se cumpla criterio de parada HACER
    P(t+1) = Seleccionar individuos de P(t)
    P(t+1) = Cruzar individuos de P(t+1)
    P(t+1) = Mutar individuos de P(t+1)
    Ejecutar Subrutina SUB_GA

    SI porcentaje de individuos inválidos en P(t+1) > Pinv ENTONCES
      P(t+1) = Inicializar población
      Ejecutar Subrutina SUB_GA
    FINSI

    t = t + 1
  FINMIENTRAS

  Imprimir resultados
FIN


SUBRUTINA SUB_GA
INICIO
  Descartar individuos iguales de P(t+1)
  Evaluar fitness de individuos de P(t+1)

  SI mejor individuo de P(t+1) es mejor que Gbest ENTONCES
    Gbest = Guardar mejor individuo de P(t+1)
  FINSI
FIN
```

Figura 5. Pseudocódigo del algoritmo basado en GA.

7 Detalles de la implementación con PSO

Como mencionamos en la introducción, el planteamiento de la solución utilizando PSO, puede ser considerado como una adaptación de la formulación tradicional de PSO. Inicialmente nos basamos en lo planteado en [2] para el problema del Cajero Viajante (TSP), sin embargo, esta evaluación de la nueva posición, para nuestro problema, dependía en gran medida de la población inicial y no existía la suficiente diversidad (no se lograba recorrer todo el espacio solución). Finalmente, implementamos la formula tradicional del PSO también planteada en [2], con algunas modificaciones. En nuestro caso, no se tiene en cuenta la velocidad actual, y el parámetro cognitivo (α) y el parámetro social (β), son números reales en el rango [0,1], tal que $\alpha + \beta = 1$.

Considerando un enjambre de *M* partículas que se mueven en un espacio *N*-dimensional, cada partícula *i* conoce su posición actual $X_i = [x_{i0}, x_{i1},..., x_{iN-1}]$ y la mejor posición en la que se ha encontrado $P_i = [p_{i0}, p_{i1},..., p_{iN-1}]$, o *mejor posición personal*. Además, toda partícula conoce la mejor posición encontrada por el enjambre $G = [g_0, g_1,..., g_{N-1}]$, o *mejor posición global*.

El procedimiento para actualizar la posición de cada partícula *i* se realiza conforme a las ecuaciones 5 y 6. En cada iteración *t* del algoritmo, primero se calcula el desplazamiento que sufrirá cada componente *j* de la partícula *i*, en base a la distancia de la *posición actual* respecto de la *mejor posición personal* y de la *mejor posición global*, de ahí que vaya a tomarse el valor absoluto entre las diferencias calculadas. Adicionalmente, debido a que el espacio de solución es discreto y positivo, es decir, $X_i, P_i, G \in Z_+^N$ (conjunto *N*-dimensional de enteros positivos), redondeamos el valor obtenido al entero más cercano, a través de la función *round(x)*[1]. Finalmente, este desplazamiento es sumado a la *posición actual* de la partícula para obtener la *nueva posición*. El rango de valores que puede tomar la coordenada *j* de la posición de una partícula depende de la cantidad de rutas para la demanda asociada a dicha coordenada, por tanto, debemos mantener la posición dentro de este rango, lo cual se logra con la operación de módulo sobre la cantidad de rutas en cuestión.

[1] El funcionamiento de esta función puede apreciarse a través de los siguientes ejemplos: *round*(2.41) = 2, *round*(1.5) = 2

Haciendo una comparación con el modelo tradicional de [2], el desplazamiento podría considerarse como la velocidad con que la partícula se desplaza o cambia de posición.

$$d_{ij}^{t} = round\left(\alpha \cdot \left| p_{ij}^{t} - x_{ij}^{t} \right| + \beta \cdot \left| g_{j}^{t} - x_{ij}^{t} \right| \right) \quad (5)$$

$$x_{ij}^{t+1} = \left(x_{ij}^{t} + d_{ij}^{t} \right) \bmod L_j \quad (6)$$

Siendo:

- x_{ij}^{t} posición actual de la partícula *i* en la coordenada *j*
- p_{ij}^{t} mejor posición personal de la partícula *i* en la coordenada *j*
- g_{j}^{t} mejor posición global alcanzada por alguna partícula del enjambre en la coordenada *j*
- d_{ij}^{t} desplazamiento que tendrá la partícula *i* en la coordenada *j*
- x_{ij}^{t+1} nueva posición de la partícula *i* en la coordenada *j*
- L_j cantidad de rutas para la demanda asociada a la coordenada *j*

En la Figura 6 se ilustra la actualización de la posición de una partícula en un espacio de dos dimensiones, para valores de α y β iguales a 0.25 y 0.75, respectivamente. Cabe destacar que para el caso de dos dimensiones, las posiciones de las partículas solo podrán ocupar el primer cuadrante del plano cartesiano, lo cual se asegura con la ecuación 5.

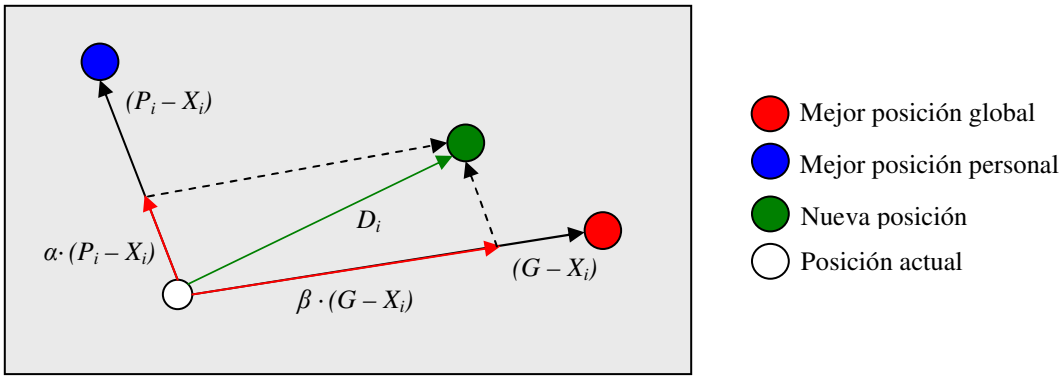


Figura 6. Ilustración de la actualización de la posición de la partícula *i* en un espacio de dos dimensiones.

Considerando los ejemplos mostrados en la Figura 2 y Figura 3, pero solo tomando las dos primeras demandas (dos dimensiones), en la Figura 7 mostramos un ejemplo del movimiento de una partícula, con valores de α, β, L_1 y L_2 iguales a 0.25, 0.75, 5 y 5.

Mejor posición global (g)

1	2
4	0

Mejor posición personal (p)

1	2
1	4

Posición actual (x)

1	2
2	0

j	*t*										*t +1*
	g_j	p_j	x_j	$\lvert p_j - x_j \rvert$	$\lvert g_j - x_j \rvert$	$\alpha \cdot \lvert p_j - x_j \rvert$	$\beta \cdot \lvert g_j - x_j \rvert$	d_j	$round(d_j)$	$round(d_j) \bmod L_j$	x_j
1	4	1	2	1	2	0.25	1.5	1.75	2	2	4

Nueva posición de la partícula

1	2
4	2

Figura 7. Ejemplo de la actualización de la posición de una partícula

La estructura general del algoritmo PSO es similar al algoritmo GA, con la diferencia que el PSO no realiza selección, cruce, ni mutación, y en contrapartida calcula la nueva posición de las partículas según las ecuaciones 5 y 6. Dos operaciones adicionales fueron agregadas al algoritmo PSO para que exista mayor diversidad: la reinicialización y el descarte de partículas iguales, tal como se hizo con el algoritmo GA. Para las pruebas realizadas, los valores de P_{inv} y del tamaño del enjambre fueron, respectivamente, 80% y 32 partículas. El pseudocódigo del algoritmo PSO se presenta a continuación:

```
INICIO
  Leer demandas y rutas
  t = 0
  E(t) = Inicializar enjambre
  Descartar partículas iguales de E(t)
  Evaluar fitness de partículas de E(t)
  Pbest = Guardar mejores posiciones personales de E(t)
  Gbest = Guardar mejor posición global de E(t)

  MIENTRAS no se cumpla criterio de parada HACER
    E(t+1) = Actualizar posiciones de partículas de E(t)
    Ejecutar Subrutina SUB_PSO

    SI porcentaje de partículas inválidas en E(t+1) > Pinv ENTONCES
      E(t+1) = Inicializar enjambre
      Ejecutar Subrutina SUB_PSO
    FINSI

    t = t + 1
  FINMIENTRAS

  Imprimir resultados
FIN


SUBRUTINA SUB_PSO
INICIO
  Descartar partículas iguales de E(t+1)
  Evaluar fitness de partículas de E(t+1)

  PARA cada partícula k de E(t+1) HACER
    SI k es mejor que Pbest(k) ENTONCES
      Pbest(k) = Actualizar mejor posición personal k
    FINSI
  FINPARA

  SI mejor posición global de E(t+1) es mejor que Gbest ENTONCES
    Gbest = Guardar mejor posición global de E(t+1)
  FINSI
FIN
```

Figura 8. Pseudocódigo del algoritmo basado en PSO.

8 Resultados

Las pruebas fueron realizadas sobre una versión asimétrica de la red NSF (NSFNet, National Science Foundation Network), la cual cuenta con 14 nodos, 42 aristas y 14226 rutas en total, lo que da un promedio de aproximadamente 78 rutas por cada una de las 182 demandas unicast posibles.

Se realizó cinco grupos de ejecuciones con conjuntos de demandas de distintos tamaños: 5, 10, 15, 20 y 25. Para cada conjunto se realizaron 10 ejecuciones consecutivas de cada algoritmo y se calculó el promedio de los costos obtenidos en función al tiempo, tomando muestras cada 1 segundo.

Para un conjunto de demandas, considerando una muestra obtenida (digamos, a los 10 segundos) para cada una de las 10 ejecuciones, se daba el caso de que no todas las soluciones eran válidas, por tanto el cálculo del promedio de costos se realizó teniendo en cuenta solo soluciones válidas. Lo anterior da lugar a que pueda no existir solución válida y, como esto solo ocurría al inicio de las ejecuciones (solos los primeros segundos), las curvas de costo en función al tiempo no siempre inician en el tiempo cero.

Los gráficos del costo en función al tiempo, pueden verse en las Figuras 9, 10, 11, 12 y 13. Cada ejecución de los algoritmos tuvo una duración de 3 minutos y se cargó la máxima cantidad de rutas por demanda. Es importante tener en

cuenta que por cuestiones de eficiencia de espacio no siempre será posible cargar todas las rutas, considerando grafos que tengan 50, 100 o más nodos y una cantidad de aristas proporcional al cuadrado de la cantidad de nodos.

8.1 Ambiente de ejecución

La implementación de este trabajo, para ambos algoritmos, fue realizada con el lenguaje de programación Java, cuya versión del compilador utilizado fue 1.6.0_03.

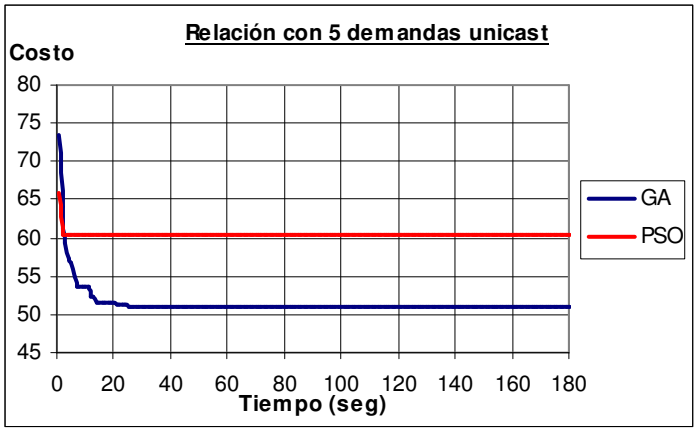
La configuración de hardware de la computadora en la que fueron realizadas las pruebas de ejecución de ambas implementaciones son las siguientes: Procesador Intel Pentium IV de 2.8GHz, Memoria RAM de 512MB, Sistema operativo Windows XP.

8.2 Prueba Nº 1

Esta prueba corresponde a la ejecución de los algoritmos GA y PSO para 5 demandas. Como se puede observar en la Figura 9, GA encuentra mejores resultados y su tendencia es más regular con respecto a PSO que se detiene rápidamente en un óptimo local. El hecho de que sea más regular, se refiere a que la pendiente de la curva varía más uniformemente.

La solución óptima para el conjunto de demandas de esta prueba tiene un costo de 51. Pudimos notar que el algoritmo GA en todas las ejecuciones encontró dicha solución, no así el algoritmo PSO, que a pesar de que tuvo un costo promedio de 60.4, logró encontrar dicha solución en algunas ocasiones.

Otro fenómeno encontrado son las porciones de curva con pendiente cero, lo que indica que el algoritmo no ha encontrado mejores soluciones durante un cierto periodo de tiempo. Esto se da porque en dicho periodo de tiempo el algoritmo probablemente esté explorando otras áreas menos prometedoras. Este fenómeno se da en mayor o menor medida en todas las pruebas. Particularmente para el PSO en esta prueba, este fenómeno se mantuvo hasta el final de la ejecución, pudiendo haber mejorado con un mayor tiempo de ejecución, no así el algoritmo GA ya que éste encontró la mejor solución.



(a)

Demandas Unicast		
Origen	Destino	Ancho de banda (Mbps)
1	3	0.2
1	9	0.31
12	11	0.1
0	10	0.5
6	5	0.3

(b)

Figura 9. (a) Gráfico de costo en función del tiempo para 5 demandas. (b) Demandas utilizadas para la prueba Nº 1.

8.3 Prueba Nº 2

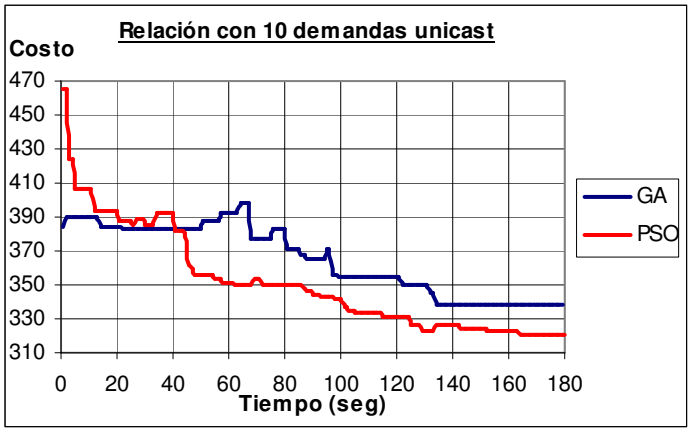
Esta prueba corresponde a la ejecución de los algoritmos GA y PSO para 10 demandas. Como se puede observar en la Figura 10, PSO encuentra mejores resultados y el comportamiento de ambos es inestable, lo cual se debe a lo expuesto en los siguientes dos párrafos.

La tendencia general de cualquier ejecución es que los costos de soluciones válidas vayan disminuyendo a medida que trascurre el tiempo. En algunas ejecuciones se demora un cierto tiempo en encontrar la primera solución válida, tiempo que puede variar entre ejecuciones. Antes de encontrar la primera solución válida, la curva de costo en función al tiempo no necesariamente cumplirá con la tendencia de las soluciones válidas, lo que ocasionaría que el costo pueda aumentar y disminuir intercaladamente según trascurre el tiempo.

En forma global, teniendo en cuenta varias ejecuciones, el hecho de que varíe entre ellas el tiempo en el que se encuentra la primera solución válida, ocasiona que el costo promedio aumente en vez de que disminuya o se mantenga. Teniendo en cuenta que solo utilizamos las soluciones válidas para calcular el costo promedio, una ejecución que demoró cierto tiempo t_2 en encontrar su primera solución válida, hará aumentar el costo promedio calculado (hasta ese momento) por las ejecuciones que encontraron su primera solución en un tiempo $t_1 < t_2$, debido a que su valor

probablemente será mayor a dicho promedio, ya que normalmente los primeros valores válidos encontrados tendrán un costo relativamente alto.

La solución óptima para el conjunto de demandas de esta prueba tiene un costo de 107. Pudimos notar que ninguno de los algoritmos se asomó a dicha solución, quedando en costos promedios de 338.55 y 320.33 y en mejores costos de 291 y 205, para GA y PSO, respectivamente.



(a)

Demandas Unicast		
Origen	Destino	Ancho de banda (Mbps)
1	5	0.3
3	5	0.2
12	5	0.5
0	10	0.2
6	5	0.4
10	13	0.2
7	13	0.4
1	12	0.1
9	13	0.2
10	4	0.3

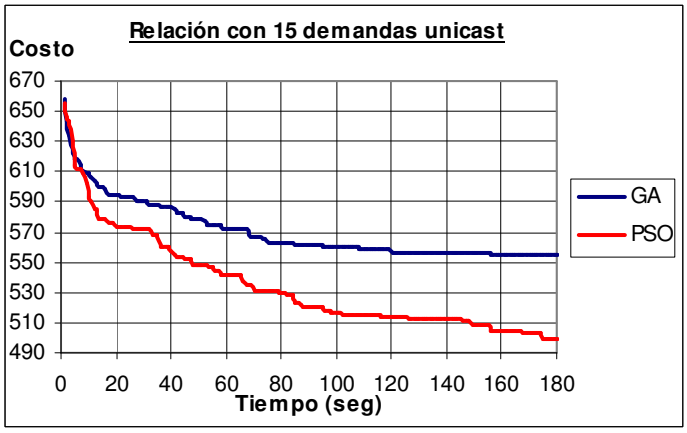
(b)

Figura 10. (a) Gráfico de costo en función del tiempo para 10 demandas. (b) Demandas utilizadas para la prueba Nº 2.

8.4 Prueba Nº 3

Esta prueba corresponde a la ejecución de los algoritmos GA y PSO para 15 demandas. Como se puede observar en la Figura 11, PSO encuentra mejores resultados y la tendencia de ambos es bastante regular.

La solución óptima para el conjunto de demandas de esta prueba tiene un costo de 184. Pudimos notar que ninguno de los algoritmos se asomó dicha solución, quedando en costos promedios de 554.7 y 499.7 y en mejores costos de 535 y 465, para GA y PSO, respectivamente.



(a)

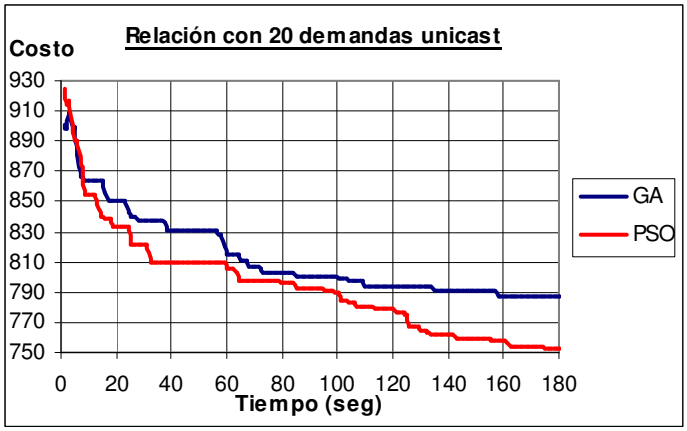
Demandas Unicast		
Origen	Destino	Ancho de banda (Mbps)
1	5	0.17
3	5	0.09
12	5	0.21
0	10	0.05
6	5	0.04
10	13	0.2
7	13	0.02
1	12	0.1
9	13	0.03
10	4	0.1
2	7	0.02
3	5	0.09
0	13	0.1
5	3	0.18
4	9	0.06

(b)

Figura 11. (a) Gráfico de costo en función del tiempo para 15 demandas. (b) Demandas utilizadas para la prueba Nº 3.

8.5 Prueba Nº 4

Esta prueba corresponde a la ejecución de los algoritmos GA y PSO para 20 demandas. Como se puede observar en la Figura 12, PSO encuentra mejores resultados y la tendencia de ambos es bastante regular.



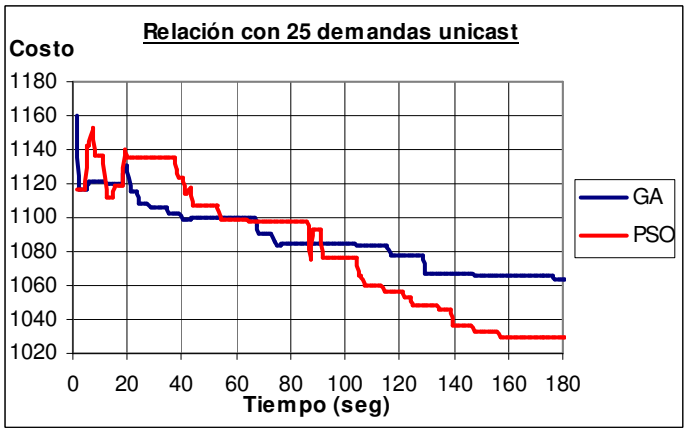
Demandas Unicast		
Origen	Destino	Ancho de banda (Mbps)
1	5	0.17
3	5	0.09
12	5	0.2
0	10	0.05
6	5	0.04
10	13	0.19
7	13	0.02
1	12	0.1
9	13	0.03
10	4	0.1
2	7	0.02
3	5	0.09
0	13	0.1
5	3	0.18
4	9	0.04
8	7	0.01
4	9	0.02
10	11	0.03
5	6	0.02
1	8	0.01

(a) (b)

Figura 12. (a) Gráfico de costo en función del tiempo para 20 demandas. (b) Demandas utilizadas para la prueba Nº 4.

8.6 Prueba Nº 5

Esta prueba corresponde a la ejecución de los algoritmos GA y PSO para 25 demandas. Como se puede observar en la Figura 13, PSO encuentra mejores resultados hacia el final, sin embargo, este es muy irregular entre ejecuciones. Por otro lado el GA es más regular pero no llega a encontrar mejor solución que el PSO. Las irregularidades encontradas se deben al problema explicado en la prueba Nº 2.



Demandas Unicast		
Origen	Destino	Ancho de banda (Mbps)
1	5	0.17
3	5	0.09
12	5	0.2
0	10	0.05
6	5	0.04
10	13	0.19
7	13	0.02
1	12	0.1
9	13	0.03
10	4	0.1
2	7	0.02
3	5	0.09
0	13	0.1
5	3	0.08
4	9	0.04
8	7	0.01
4	9	0.02
10	11	0.03
5	6	0.02
1	8	0.01
3	2	0.01
3	5	0.02
10	5	0.02
12	13	0.03
12	0	0.05

(a) (b)

Figura 13. (a) Gráfico de costo en función del tiempo para 25 demandas. (b) Demandas utilizadas para la prueba Nº 5.

9 Conclusiones

Si bien la cantidad y variedad de las pruebas realizadas no haya sido lo suficiente para un trabajo de este tipo y aunque las mismas se hayan limitado solo a una topología (NSFNet), los resultados obtenidos muestran que el algoritmo PSO obtiene mejores soluciones en cuatro de las cinco pruebas realizadas.

Un hecho resaltante es que en las pruebas Nº 2 y Nº 3 no se haya llegado a encontrar las soluciones óptimas, tal como se logró en la prueba Nº 1. Teniendo en cuenta la tendencia de ambos algoritmos a ir mejorando según trascurre el tiempo, consideramos que esto mejoraría si el tiempo de ejecución fuese aumentando conforme aumenta el tamaño del conjunto de demandas. Cabe destacar que para una misma topología, a medida que crece la cantidad de demandas, también lo hace el espacio de solución, por lo que recorrerlo enteramente demora más tiempo.

Por otra parte, para el problema de enrutamiento abordado en este trabajo, notamos que poder realizar un buen recorrido del espacio solución no es nada trivial, ya que no existe una relación aparente de "similitud" entre dos soluciones. Por ejemplo, al modificar una arista en una de las rutas de una solución se puede tener un mejor costo en la ruta pero esta modificación puede llegar a invalidar la solución, y por ende, estas dos soluciones no serían "similares". Esto ocasiona que en ocasiones se genere un alto porcentaje de individuos inválidos en la población, debido a lo cual fue necesario utilizar la reinicialización de la población.

10 Trabajos futuros

Como trabajo futuro se plantea realizar un mayor análisis del impacto de las operaciones de cruce, mutación y reinicialización en la convergencia y "evolución" de los algoritmos hacia mejores soluciones. También podría considerarse una mejor representación para los individuos, de tal manera que pequeñas modificaciones a nivel unitario no impliquen cambios muy grandes a nivel global.

En cuanto al algoritmo PSO, creemos que sería interesante aplicar conceptos de vecindad, como una manera de lidiar con la convergencia prematura y como mecanismo de aumentar la diversidad. Además, sería interesante poder probar la aplicabilidad del enfoque propuesto en este trabajo a otros problemas con naturaleza combinatoria.

Finalmente, nos gustaría realizar las pruebas con tiempos escalonados, es decir, conforme aumentan las demandas aumentar también el tiempo de ejecución. Creemos que esto haría que se obtengan mejores resultados con cada algoritmo.

Referencias

[1] Jorge Crichigno. Enrutamiento Multicast utilizando Optimización Multiobjetivo. Universidad Católica "Nuestra Señora de la Asunción", Facultad de Ciencias y Tecnología, Proyecto final de tesis para el título de Ingeniero en Electrónica. Asunción, Paraguay, 2004.

[2] Joaquín Quinto Lima Molinari. Aplicación de Optimización de Enjambre de Partículas al Problema de Cajero Viajante Bi-Objetivo. Universidad Nacional de Asunción, Facultad Politécnica, Proyecto final de tesis para el título de Ingeniero en Informática. San Lorenzo, Paraguay, 2006.

[3] Dr. Carlos A. Coello Coello. Introducción a la Computación Evolutiva (Notas de Curso). CINVESTAV-IPN, Departamento de Ingeniería Eléctrica, Sección de Computación. México, D.F., 2004.