Comparación de Algoritmos Evolutivos Multi-Objetivos en un ambiente Multicast.

Francisco Talavera[+], Joel Prieto[+], Jorge Crichigno[+*], Benjamín Barán[+*]
ftalavera@yahoo.com.ar, jprieto@telesurf.com.py, jcrichigno@cnc.una.py, bbaran@cnc.una.py
Ciencias y Tecnología - Universidad Católica Nuestra Señora de la Asunción[+]
Centro Nacional de Computación - Universidad Nacional de Asunción[*]

Resumen

En problemas de optimización multi-objetivo, se plantean dos o más funciones objetivo que se optimizarán al mismo tiempo, buscando el conjunto de las mejores soluciones de compromiso, o conjunto Pareto. Este es el caso de la ingeniería de tráfico multicast, que pretende optimizar costo y retardo entre otras posibles métricas.

Como han sido publicados numerosos Algoritmos Evolutivos Multiobjetivos o MOEAs, no queda aún claro cual es el que presenta mejor desempeño para el problema considerado. Por esta razón, en este trabajo hacemos una comparación experimental entre 5 alternativas: NSGA, NSGA2, SPEA, SPEA2 y cNSGA2, con el fin de determinar cual es la más apropiada para resolver problemas de enrutamiento multicast. Resultados experimentales demuestran que algoritmos como el SPEA y SPEA2 logran un excelente desempeño en este tipo de problemas.

Palabras Claves: Redes, Multicast, Algoritmos Evolutivos Multiobjetivos, Dominancia Pareto.

1. Introducción

Una transmisión Multicast consiste en el envío simultáneo o concurrente de datos desde una fuente a un conjunto de destinos componentes de una red de computadoras [1]. En estos últimos años, los algoritmos de enrutamiento Multicast se han vuelto mucho más importantes debido a la creciente utilización de nuevas aplicaciones de transmisión punto a multipunto, como lo son las transmisiones de radio y televisión, video bajo demanda, teleconferencias y enseñanza a distancia.

A partir de esta creciente tendencia, el retardo desde la fuente a cada uno de los destinos se torna una variable de vital importancia en trasmisiones Multicast de audio y/o video [2]. Otro punto importante en la optimización del enrutamiento Multicast son los "costos" del árbol, entendiéndose por "costos" otras métricas a ser minimizadas como: el número de saltos (*hop count*), utilización del ancho de banda, etc. De esta forma, la ingeniería de tráfico multicast queda planteada como un problema multi-objetivo.

Investigaciones realizadas sobre el principio de *survival of the fittest* (ley de supervivencia de los más aptos) observado en la naturaleza, dieron como resultado simulaciones computacionales que resultaron muy útiles para solucionar problemas complejos [4], dando origen a los Algoritmos Evolutivos (EAs). Los EAs han sido muy populares en tareas de búsqueda y de optimización en los últimos años, con un desarrollo constante de nuevos algoritmos. En particular, los Algoritmos Evolutivos Multi-Objetivos (MOEAs) permiten resolver problemas multi-objetivos (MOP), encontrando un conjunto completo de soluciones *Pareto* en una sola ejecución [3], convirtiéndolos en un candidato natural para resolver el problema del enrutamiento multicast.

En la literatura se tienen trabajos que comparan el rendimiento de algunos Algoritmos. En [12] se analiza el rendimiento de NSGA-II, SPEA-II y NSGA-II con elitismo controlado en diseño de sistemas de seguridad.

En [10] se estudia el pasado, presente y posible futuro de los Algoritmos Evolutivos Multi-objetivos, brindando un esquema y comentarios, de los algoritmos de primera generación, NSGA, NPGA, MOGA, y los de segunda generación, SPEA, SPEA2, PAES, NSGA2, NPGA2, micro-GA. Así mismo, se proponen posibles disciplinas en la que se pueden aplicar los Algoritmos Genéticos.

El problema especifico de la optimización multicast ya esta siendo tratado en la literatura. Así, en [11] se presenta un Algoritmo de Enrutamiento Multicast Multiobjetivo MMA1 que a diferencia del presentado, posee una codificación distinta

En el presente trabajo compararemos cinco algoritmos evolutivos multi-objetivos. Ellos son:

- **NSGA** (*Non dominated Sort Genetic Algorithm*)[5]
- **NSGA2** (*Non dominated Sort Genetic Algorithm 2*)[6]
- **cNSGA2** (*Controlled Non dominated Sort Genetic Algorithm 2*)[7]
- **SPEA** (*Strength Pareto Evolutionary Algorithm*)[8]
- **SPEA2** (*Strength Pareto Evolutionary Algorithm 2*)[9]

El resto del documento se encuentra organizado de la siguiente manera. Una definición general de un problema de optimización multi-objetivo es presentada en la Sección 2. La formulación del problema y las funciones objetivo son dadas en la Sección 3. Una breve introducción a los algoritmos evolutivos y la codificación utilizada se presenta en la Sección 4. En la Sección 5, una breve descripción de cada uno de los algoritmos. Pruebas y resultados experimentales de los algoritmos son mostrados en la Sección 6. Finalmente, la conclusión y los trabajos futuros son expuestos en la Sección 7.

.

2. Problemas de Optimización Multiobjetivo

Un Problema de Optimización Multiobjetivo (MOP) general incluye un conjunto de n parámetros (variables de decisión), un conjunto de k funciones objetivo, y un conjunto de m restricciones. Las funciones objetivo y las restricciones son funciones de las variables de decisión. Luego, el MOP puede expresarse como:

$$\begin{array}{ll} \text{Optimizar} & \boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x})) \\ \text{sujeto a} & \boldsymbol{e}(\boldsymbol{x}) = (e_1(\boldsymbol{x}), e_2(\boldsymbol{x}), \ldots, e_m(\boldsymbol{x})) \geq \boldsymbol{0} \\ \text{donde} & \boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \boldsymbol{X} \\ & \boldsymbol{y} = (y_1, y_2, \ldots, y_k) \in \boldsymbol{Y} \end{array} \qquad \textbf{(3.1)}$$

siendo $\boldsymbol{x}$ el vector de decisión e $\boldsymbol{y}$ el vector objetivo. El espacio de decisión se denota por $\boldsymbol{X}$, y al espacio objetivo por $\boldsymbol{Y}$. Optimizar, dependiendo del problema, puede significar igualmente, minimizar o maximizar. El conjunto de restricciones $\boldsymbol{e(x)} \geq \boldsymbol{0}$ determina el conjunto de soluciones factibles $\boldsymbol{X}_f \subset \boldsymbol{X}$ y su correspondiente conjunto de vectores objetivo factibles $\boldsymbol{Y}_f \subset \boldsymbol{Y}$.

El problema de optimización consiste en hallar la $\boldsymbol{x}$ que tenga el "mejor valor" de $\boldsymbol{f(x)}$. En general, no existe un único "mejor valor", sino un conjunto de soluciones óptimas. Entonces, un

nuevo concepto de optimalidad debe ser establecido para MOPs. Dados dos vectores de decisión ***u***, ***v*** $\in \boldsymbol{X_f}$, puede darse una de las tres condiciones siguientes:

$$\begin{aligned} &\boldsymbol{f(u)} = \boldsymbol{f(v)} \text{ si y solo si } \forall i \in \{1, 2, \ldots, k\}: f_i(\boldsymbol{u}) = f_i(\boldsymbol{v}) \\ &\boldsymbol{f(u)} \leq \boldsymbol{f(v)} \text{ si y solo si } \forall i \in \{1, 2, \ldots, k\}: f_i(\boldsymbol{u}) \leq f_i(\boldsymbol{v}) \\ &\boldsymbol{f(u)} < \boldsymbol{f(v)} \text{ si y solo si } \boldsymbol{f(u)} \leq \boldsymbol{f(v)} \wedge \boldsymbol{f(u)} \neq \boldsymbol{f(v)} \end{aligned} \qquad \textbf{(3.2)}$$

En un contexto de minimización, esta situación se expresa con los siguientes símbolos y términos:

$\mathbf{u} \succ \mathbf{v}$ (***u*** domina a ***v***)	si y solo si	$\boldsymbol{f(u)} < \mathbf{f(v)}$	
$\boldsymbol{v} \succ \boldsymbol{u}$ (***v*** domina a ***u***)	si y solo si	$\boldsymbol{f(v)} < \boldsymbol{f(u)}$	**(3.3)**
$\boldsymbol{u} \sim \boldsymbol{v}$ (***u*** y ***v*** no son comparables)	si y solo si	$\boldsymbol{f(u)} \not\prec \boldsymbol{f(v)} \wedge \boldsymbol{f(v)} \not\prec \boldsymbol{f(u)}$	

Alternativamente, $\boldsymbol{u} \rhd \boldsymbol{v}$ denota que ***u*** domina o es igual a ***v***.

Dado un vector de decisión $\boldsymbol{x} \in \boldsymbol{X_f}$, se dice que ***x*** es no dominado respecto a un conjunto $\boldsymbol{V} \subseteq \boldsymbol{X_f}$ si y solo si $\boldsymbol{x} \succ \boldsymbol{v}$ o $\boldsymbol{x} \sim \boldsymbol{v}$, $\forall \boldsymbol{v} \in \boldsymbol{V}$. En caso que ***x*** sea no dominado respecto a todo el conjunto $\boldsymbol{X_f}$, y solo en ese caso, se dice que ***x*** es una solución Pareto óptima. Por lo tanto, el conjunto Pareto óptimo $\boldsymbol{X_{true}}$ puede ser definido formalmente de la siguiente manera:

$$\boldsymbol{X_{true}} = \{ \boldsymbol{x} \in \boldsymbol{X_f} \mid \boldsymbol{x} \text{ es no dominado con respecto a } \boldsymbol{X_f} \} \qquad \textbf{(3.4)}$$

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

3. Formulación del Problema

La red es modelada como un grafo dirigido $G = (V, E)$, donde V es el conjunto de nodos y E es el conjunto de enlaces. Sea $(i,j) \in E$ el enlace entre los nodos i y j. Sea z_{ij}, c_{ij}, d_{ij} y $t_{ij} \in \Re^+$ la capacidad, el costo, el retardo y el tráfico actual (en bps) del enlace (i,j).Un grupo multicast está dado por un nodo fuente $s \in V$, un conjunto de nodos destinos $N \subseteq V - \{s\}$ y una demanda de tráfico $\phi \in \Re^+$ (en bps). Sea $T(s,N)$ un árbol multicast con origen en s y conjunto de nodos destinos $N \subset V$; $p_T(s, n) \subseteq T(s,N)$ representa el camino que conecta el nodo fuente s y el nodo destino $n \in N$, y $d(p_T(s, n))$ el retardo de este camino, dado por la suma de los retardos de los enlaces que conforman el camino. Esto es, $d(p_T(s,n)) = \sum_{(i,j) \in p_T(s,n)} d_{ij}$.

El problema de enrutamiento multicast puede ser formulado como un MOP en el cual se desea construir el árbol multicast $T(s,N)$ que minimice las siguientes funciones objetivo:

1- Utilización máxima de los enlaces del árbol: $\alpha_T = \underset{(i,j) \in T}{Max} \{ (\phi + t_{ij}) / z_{ij} \}$ **(4.1)**

2- Costo del árbol: $C_T = \phi \sum_{(i,j) \in T} c_{ij}$ **(4.2)**

3- Retardo máximo de extremo a extremo: $D_M = \underset{n \in N}{Max} \{ d(p_T(s,n)) \}$ **(4.3)**

4- Retardo medio: $D_A = \frac{1}{|N|} \sum_{n \in N} d(p_T(s,n))$ **(4.4)**

sujeto a restricciones de capacidad en los enlaces: $\phi + t_{ij} \leq z_{ij}$; $\forall (i,j) \in T(s,N)$

4. Algoritmos Evolutivos MultiObjetivos

Los algoritmos considerados para este trabajo son los Algoritmos Evolutivos MultiObjetivos o MEAs que se inician con un conjunto de configuraciones aleatorias llamada población inicial. Cada individuo (cromosoma) en la población representa una solución al problema de optimización. En cada generación, los individuos son evaluados usando una función de adaptabilidad (*fitness*). Basados en este valor, algunos individuos, llamados padres, son seleccionados. La probabilidad de selección de un individuo está relacionada con su adaptabilidad, de forma a asignar mayor probabilidad de selección a los mejores individuos. Luego, un número de operadores genéticos son aplicados a los padres para producir nuevos individuos que formarán parte de la nueva población. El proceso continua intentando obtener soluciones cada vez mejores hasta que un criterio de parada sea satisfecho.

Para este trabajo, cada individuo es representado directamente como el conjunto de los enlaces de un árbol [2,13], Como se vera en la figura 2.

La inicializacion aquí propuesta genera $|P|$ individuos aleatoriamente, donde P es la población. Cada individuo es obtenido con la construcción de árboles basados en el algoritmo de Prim [13]. Empezando con un nodo fuente s, el algoritmo expande el árbol eligiendo un nuevo enlace en cada iteración, añadiendo de esta forma un nuevo nodo al árbol. La elección del enlace se hace aleatoriamente. De esta forma, el algoritmo, denominado PrimRST (*Prim Random Steiner Tree*), inicializa la población evolutiva P. Una población de soluciones no dominadas P_{nd} es inicializada con aquellos individuos no dominados pertenecientes a la primera generación de P. El pseudocódigo del procedimiento PrimRST es mostrado en la Figura 1.

```
PrimRST(G(V,E), s, N)   //recibe la red G(V, E) y el grupo s, N.
T_p = {φ};   //Inicialización del árbol a crear (vacío)
V_c = {s};   //Nodos conectados a T_p
A = {(s, j) ∈ E | j ∈ V};   //Enlaces candidatos a ser añadidos a T_p
Mientras (N ∪ {s} ⊄ V_c)
    Elegir aleatoriamente un enlace (i, j) ∈ A; i ∈ V_c;
    A = A − {(i, j)};
    Si j ∉ V_c entonces              // conectar j al árbol T_p
        T_p = T_p ∪ {(i, j)}
        V_c = V_c ∪ {j};
        A = A ∪ {(j, w) ∈ E | w ∉ V_c};
    Fin si
Fin Mientras
Borrar los enlaces inútiles de T_p
Retornar T_p
```

Figura 1 Procedimiento utilizado para la construcción de un árbol T_p.

El operador genético, utilizado consiste de dos pasos:

Paso 1: Se debe identificar los enlaces comunes de ambos padres y copiarlos al individuo hijo. De acuerdo al procedimiento de selección de los algoritmos evolutivos, los individuos con mejor *fitness* tienen mayor probabilidad de ser seleccionados. Entonces, los enlaces de los padres posiblemente sean “buenos”. Sin embargo, considerar solo estos enlaces puede conducir a un individuo hijo que consiste en sub-árboles separados, y por lo tanto algunos nuevos enlaces deben ser añadidos;

Paso 2: conectar los sub-árboles hasta formar un árbol multicast. En esta etapa, los sub-árboles son conectados de forma aleatoria, y cada sub-árbol tiene asignado un nodo raíz. El algoritmo de interconexión añade en cada iteración un enlace cuyo nodo origen forma parte de un sub-árbol. Dos sub-árboles son conectados cuando el enlace elegido tiene como nodo destino la raíz de uno de los sub-árboles – T_1 – y como origen un nodo perteneciente al otro sub-árbol – T_2 –. La raíz del nuevo sub-árbol (compuesto por los dos sub-árboles) es el nodo raíz de T_2. El algoritmo termina cuando todo el grupo multicast es interconectado. Luego, los enlaces no utilizados en la interconexión del grupo son borrados del árbol.

Este proceso se puede observar en la figura 2

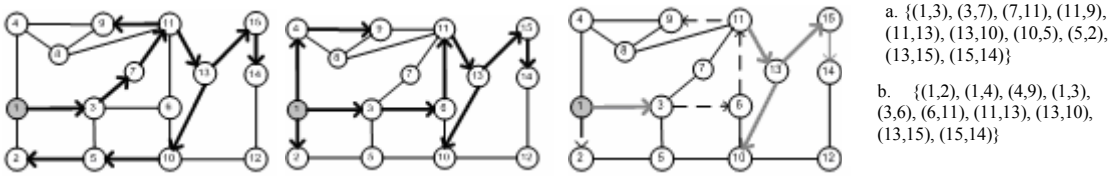

Figura 2. (a), (b) Individuos seleccionados, sobre los cuales se aplica el operador genético propuesto. (c) Resultado del operador de cruzamiento. Los enlaces discontinuos son seleccionados aleatoriamente para unir los sub-árboles, mientras que los enlaces continuos pertenecen a ambos padres y son heredados por el nuevo individuo. (d) Representación de los árboles de las figuras (a) y (b).

5. Descripción de los Algoritmos

En el presente capítulo se presentan brevemente las características principales de los cinco algoritmos evolutivos multiobjetivos a ser comparados. Para mayor información acerca de la implementación de ellos, se sugiere leer las publicaciones referenciadas.

6.1 NSGA

Este algoritmo fue propuesto por Srinivas y Deb [5]. Se basa en la clasificación de individuos en varias capas o frentes. La clasificación consiste en agrupar a todos los individuos no dominados en un frente, con un valor de *fitness* (o adaptabilidad) igual para todos los individuos. Este valor es proporcional al tamaño de la población, para así proporcionar un potencial reproductivo igual para todos los individuos de este frente. Entonces el grupo de individuos clasificados es ignorado y otro frente de individuos no dominados es considerado. El proceso continúa hasta que se clasifican a todos los individuos en la población. Puesto que los individuos en el primer frente tienen el valor de *fitness* mayor, consiguen siempre más copias que el resto de la población [10]. El pseudocódigo de este algoritmo se presenta en la Figura 3.

```
- Leer grupo multicast a enrutar.
- Inicializar población P.
  Hacer {
            Frente = 1
            Hacer {
                        - Identificar los individuos no dominados.
                        - Asignar el fitness.
                        - Asociar los individuos al frente actual
                        - Frente = Frente + 1
            } mientras no hayan sido clasificados todos los individuos

            - Reproducir de acuerdo al fitness
            - Aplicar Operadores genéticos
} mientras el criterio de parada no sea alcanzado.
```

Figura 3. Pseudocódigo del NSGA

6.2 SPEA

Este algoritmo fue introducido por el Zitzler y Thiele [8]. El SPEA utiliza un archivo que contiene las soluciones no dominadas encontradas (población externa de no dominados P_{nd}). En cada generación, se copian los individuos no dominados de P a Pnd y se borra de este las soluciones dominadas. Para cada individuo en el sistema externo, se computa un valor de fuerza (*strength*) es proporcional al número de las soluciones a las cuales cada individuo domina.

En SPEA, el *fitness* de cada miembro de la población actual se computa según las fuerzas de todas las soluciones no dominadas externas que la dominen [10]. El pseudocódigo de este algoritmo se presenta en la figura 4.

```
- Leer grupo multicast a enrutar.
- Inicializar población P.
Hacer {
            - Evaluar Individuos de P.
            - Marcar soluciones no dominadas de P.
            - Actualizar el conjunto de soluciones no
            dominadas P_N
            - Calcular la adaptabilidad de los individuos de P y
            P_N
            - Seleccionar individuos del conjunto P_N
            - Aplicar los operadores de cruzamiento y
            mutación.
} mientras el criterio de parada no sea alcanzado.
```

Figura 4. Pseudocódigo del SPEA

6.3 SPEA2

SPEA2 tiene las siguientes diferencias principales con respecto a su precursor [9]: (1) incorpora una estrategia fina de asignación del *fitness* que considera, para cada individuo, el número de los individuos que lo dominan y el número de los individuos por los cuales es dominado; (2) utiliza la técnica del "vecino más cercano" para la valoración de la densidad, dirigiendo la búsqueda en forma más eficiente. El pseudocódigo de este algoritmo se presenta en la figura 5.

```
- Leer grupo multicast a enrutar.
- Inicializar población P.
Hacer {
      - Evaluar Individuos de P.
      - Marcar soluciones no dominadas de P.
      - Actualizar el conjunto de soluciones no dominadas P_N
      - Calcular la adaptabilidad de los individuos de P y P_N
      - Seleccionar individuos del conjunto P ∨ P_N
      - Aplicar los operadores de cruzamiento y mutación.
} mientras el criterio de parada no sea alcanzado.
```

Figura 5. Pseudocódigo del SPEA2

6.4. NSGA2

DEB et al. [8] propusieron una versión revisada del NSGA[5], llamada NSGA2, que es computacionalmente más eficiente. Además, es elitista y no necesita especificar ningún parámetro adicional. El NSGA2 no utiliza una memoria externa como los algoritmos anteriores (SPEA y SPEA2). El mecanismo elitista consiste en elegir los mejores $|P|$ individuos de la unión de las poblaciones padre e hijo. El pseudocódigo de este algoritmo se presenta en la figura 6.

```
- Leer grupo multicast a enrutar.
- Inicializar población P.
- Ordenar P, considerando dominancia.
- Evaluar Individuos de P.
- Aplicar operadores genéticos a P, para tener Q
- t=0                    //cantidad de generaciones
Hacer {
        - R = P U Q.
        - ordenar R, considerando dominancia y obtener frentes F_i
        - I=1
        Mientras |P_t+1| < N {          // N numero de individuos en P
                - Calcular adaptabilidad de cada individuo en F_i
                - P_t+1 = P_t+1 U F_i
        I = I +1
        }
        - Ordenar P_t+1, por dominancia.
        - Elegir los primeros N elementos de P_t+1
        - Aplicar operadores genéticos a P_t+1, para tener Q_t+1
        - t = t +1
} mientras el criterio de parada no sea alcanzado.
```

Figura 6. Pseudocódigo del NSGA2

6.5. cNSGA2

Deb y Goel [7] propusieron una variación del NSGA 2, llamado *Controlled Non-dominated Sorting Genetic Algorithms*. En contraposición al NSGA 2, que elige los N primeros elementos de P_{t+1}, el cNSGA 2, utiliza una proporción geométrica para elegir n_i individuos de cada frente i, siendo $n_i = r \, . \, n_{i-1}$, donde r es la razon geométrica. El pseudocódigo de este algoritmo se presenta en la figura 7.

```
- Leer grupo multicast a enrutar.
- Inicializar población P.
- Ordenar por no dominados P.
- Evaluar Individuos de P.
- Aplicar operadores genéticos a P, para tener Q
- t=0                    //cantidad de generaciones
Hacer {
        - Rt = Pt U Qt.
        - F = ordenar por no dominados Rt
        - I=1
        Hasta |Pt+1| < N {                      // N numero de individuos en P
                - Asignar la distancia Fi
                - Ordenar Fi
                - Pt+1 = Pt+1 U Fi [ni:0]       //se asignan los ni elementos de Fi
        I = I +1
        }
        - Aplicar operadores genéticos a Pt+1, para tener Qt+1
        - t++
} mientras el criterio de parada no sea alcanzado.
```

Figura 7. Pseudocódigo del cNSGA2

6. Pruebas 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. Los números sobre cada enlace de Z=6Mbps representan el retardo del mismo (ver figura 8).

Se realizó un conjunto de 4 pruebas, en las cuales se varió la cantidad de nodos destino entre 9 y 24, en intervalos de 5. La cantidad de ejecuciones de cada prueba fue 15. En la Tabla 1 se presentan los grupos multicast utilizados para cada prueba.

Tabla 1: Grupos multicast utilizados durante las pruebas

	S (fuente)	**N** (Nodos destino)
Prueba 1	{5}	{0, 1, 8, 10, 22, 32, 38, 43, 53}
Prueba 2	{4}	{0, 1, 3, 5, 9, 10, 12, 23, 25, 34, 37, 41, 46, 52}
Prueba 3	{4}	{0, 1, 3, 5, 6, 9, 10, 12, 17, 22, 23, 25, 34, 37, 41, 46, 47, 52, 54}
Prueba 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}

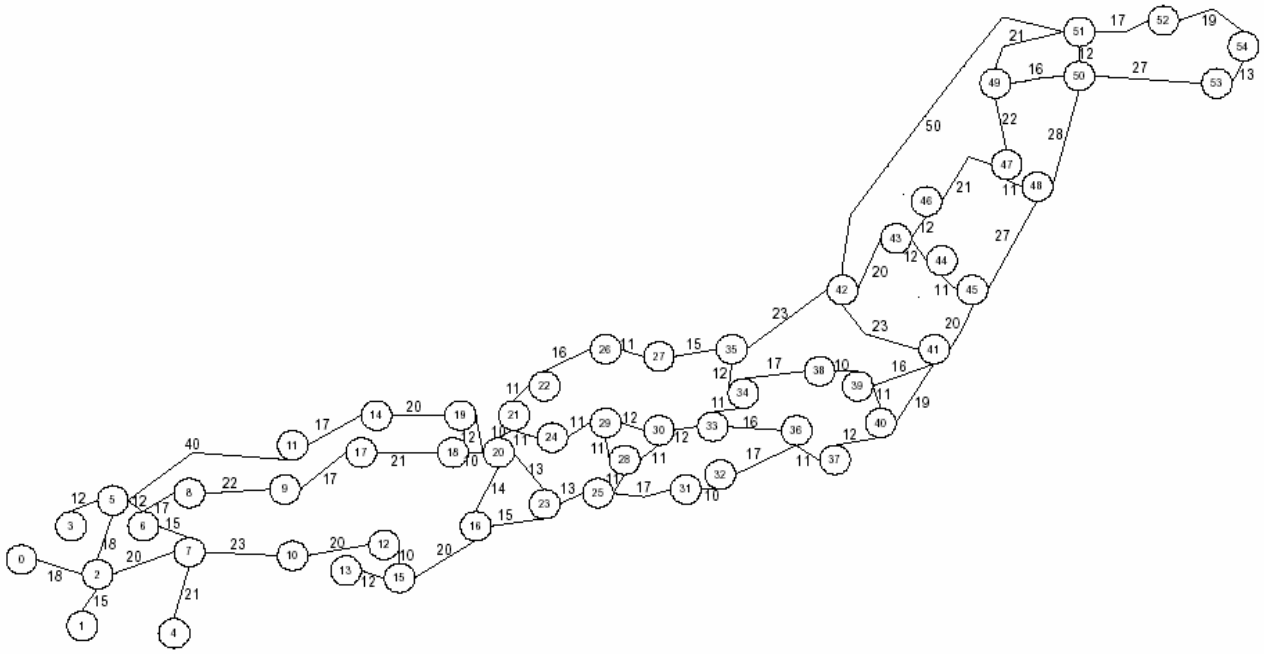

Figura 9. Red de la Nipon Telegraph and Telephone, Co

En cuanto a los parámetros de los algoritmos, se consideró para todos los casos $|P|$=50 y como criterio de parada se utilizó el número máximo de generaciones, el cual fue fijado en 100.

Las ejecuciones del algoritmo se realizaron en una NoteBook HP-Compaq NX9010 Pentium4 2.8 GHz, 512MB de memoria RAM, compilador Borland C++ 5.02.

6.1. Procedimiento de comparación

El procedimiento de comparación utilizado fue:

1. Cada uno de los 5 algoritmos se corrió quince veces.
2. Se obtuvo para cada algoritmo el conjunto de soluciones no dominadas: P1, P2,…, P15.
3. Se creó para cada algoritmo una superpoblación, donde $P_T = \bigcup_{i=1}^{15} P_i$
4. De cada Superpoblación se extrajeron las soluciones no-dominadas, formando así el frente pareto de cada algoritmo, como sigue:
   - $\boldsymbol{Y}_{\text{NSGA}}$ (frente pareto de las 15 corridas del algoritmo NSGA)
   - $\boldsymbol{Y}_{\text{NSGA2}}$ (frente pareto de las 15 corridas del algoritmo NSGA2)
   - $\boldsymbol{Y}_{\text{cNSGA2}}$ (frente pareto de las 15 corridas del algoritmo CNSGA2)
   - $\boldsymbol{Y}_{\text{SPEA}}$ (frente pareto de las 15 corridas del algoritmo SPEA)
   - $\boldsymbol{Y}_{\text{SPEA2}}$ (frente pareto de las 15 corridas del algoritmo SPEA2)
5. Se obtuvo el conjunto de soluciones encontradas como sigue

   $\hat{\boldsymbol{Y}} = \boldsymbol{Y}_{\text{NSGA}} \vee \boldsymbol{Y}_{\text{NSGA2}} \vee \boldsymbol{Y}_{\text{cNSGA2}} \vee \boldsymbol{Y}_{\text{SPEA}} \vee \boldsymbol{Y}_{\text{SPEA2}}$

6. Del conjunto $\hat{\boldsymbol{Y}}$ se obtienen las soluciones No Dominadas, y así se forma una aproximación del Frente Pareto Optimo $\hat{\boldsymbol{Y}}_{true}$.

6.2. Resultados obtenidos

La primera tabla de cada prueba, expone la cantidad de soluciones de cada algoritmo, que se encuentran en Y_{true} [$\in Y_{true}$], Las que son dominadas por Y_{true} [$Y_{true} \succ$], el número de soluciones encontradas [$|Y_{algoritmo}|$] y el porcentaje de soluciones en Y_{true} [$\%(\in Y_{true})$].

La segunda tabla de cada prueba presenta la cantidad de soluciones tales que si $\boldsymbol{p} \succ \boldsymbol{q}$, $\forall\ \boldsymbol{p} \in \boldsymbol{Y}_{\text{ALGORITMO}_i}$, y $\boldsymbol{q} \in \boldsymbol{Y}_{\text{ALGORITMO}_j}$. El subíndice i corresponde a la fila y el subíndice j corresponde a la columna. Entonces, si x_{ij} es un elemento de esta tabla, el valor de x_{ij} es la cantidad de soluciones que el algoritmo i domina al del algoritmo j.

Prueba 1

	$\in Y_{true}$	$Y_{true} \succ$	$\lvert Y_{algoritmo}\rvert$	$\%(\in Y_{true})$
$\boldsymbol{Y}_{\text{cNSGA2}}$	9	0	9	100
$\boldsymbol{Y}_{\text{SPEA2}}$	9	0	9	100
$\boldsymbol{Y}_{\text{NSGA}}$	6	1	7	85,7
$\boldsymbol{Y}_{\text{NSGA2}}$	9	0	9	100
$\boldsymbol{Y}_{\text{SPEA}}$	9	0	9	100

Tabla I: Comparación de las soluciones de los algoritmos con Y_{true}

Y_i \ Y_j	$\boldsymbol{Y}_{\text{cNSGA2}}$	$\boldsymbol{Y}_{\text{SPEA2}}$	$\boldsymbol{Y}_{\text{NSGA}}$	$\boldsymbol{Y}_{\text{NSGA2}}$	$\boldsymbol{Y}_{\text{SPEA}}$
$\boldsymbol{Y}_{\text{cNSGA2}}$		0	1	0	0
$\boldsymbol{Y}_{\text{SPEA2}}$	0		1	0	0
$\boldsymbol{Y}_{\text{NSGA}}$	0	0		0	0
$\boldsymbol{Y}_{\text{NSGA2}}$	0	0	1		0
$\boldsymbol{Y}_{\text{SPEA}}$	0	0	1	0	

Tabla II: Cantidad de soluciones en las cuales un algoritmo domina a otro

Se puede observar que los algoritmos CNSGA2, SPEA2, SPEA y NSGA2, llegan a todas las soluciones de Y_{true}, mientras que el NSGA no alcanza todas las soluciones de Y_{true}. Esto se observa en la Tabla II arriba, por la presencia de un "1" en la columna que corresponde al NSGA.

Prueba 2

	$\in Y_{true}$	$Y_{true} \succ$	$\lvert Y_{algoritmo}\rvert$	$\%(\in Y_{true})$
$\boldsymbol{Y}_{\text{cNSGA2}}$	22	0	22	100
$\boldsymbol{Y}_{\text{SPEA2}}$	22	0	22	100
$\boldsymbol{Y}_{\text{NSGA}}$	11	3	14	78.5
$\boldsymbol{Y}_{\text{NSGA2}}$	22	0	22	100
$\boldsymbol{Y}_{\text{SPEA}}$	22	0	22	100

Tabla I: Comparación de las soluciones de los algoritmos con Y_{true}

Y_i \ Y_j	$\boldsymbol{Y}_{\text{cNSGA2}}$	$\boldsymbol{Y}_{\text{SPEA2}}$	$\boldsymbol{Y}_{\text{NSGA}}$	$\boldsymbol{Y}_{\text{NSGA2}}$	$\boldsymbol{Y}_{\text{SPEA}}$
$\boldsymbol{Y}_{\text{cNSGA2}}$		0	3	0	0
$\boldsymbol{Y}_{\text{SPEA2}}$	0		3	0	0
$\boldsymbol{Y}_{\text{NSGA}}$	0	0		0	0
$\boldsymbol{Y}_{\text{NSGA2}}$	0	0	3		0
$\boldsymbol{Y}_{\text{SPEA}}$	0	0	3	0	

Tabla II: Cantidad de soluciones en las cuales un algoritmo domina a otro

Se puede observar que el SPEA2, NSGA2, SPEA y CNSGA3 no poseen ninguna solución que sea dominada por el conjunto Y_{true}, mientras que el NSGA es el algoritmo que contiene mayor cantidad de soluciones dominadas. Con este cuadro comparativo ya se puede apreciar el notable mejor rendimiento que tienen los cuatro algoritmos citados, cuando comparamos con el NSGA.

Prueba 3

	$\in Y_{true}$	$Y_{true} \succ$	$\lvert Y_{algoritmo} \rvert$	$\%(\in Y_{true})$
Y_{cNSGA2}	23	0	23	100
Y_{SPEA2}	24	0	24	100
Y_{NSGA}	11	3	14	78.5
Y_{NSGA2}	22	2	24	91.6
Y_{SPEA}	24	0	24	100

Tabla I: Comparación de las soluciones de los algoritmos con Y_{true}

Y_i \ Y_j	Y_{cNSGA2}	Y_{SPEA2}	Y_{NSGA}	Y_{NSGA2}	Y_{SPEA}
Y_{cNSGA2}		0	3	2	0
Y_{SPEA2}	0		3	2	0
Y_{NSGA}	0	0		0	0
Y_{NSGA2}	0	0	2		0
Y_{SPEA}	0	0	3	2	

Tabla II: Cantidad de soluciones en las cuales un algoritmo domina a otro

Se puede apreciar que el SPEA2, el SPEA y el CNSGA2, poseen 100% de efectividad en sus soluciones, pero cabe destacar que CNSGA2 si bien tiene 100% de efectividad en sus soluciones, no tiene a todas las soluciones de Y_{true}. Si bien, el NSGA2 tiene 24 soluciones, 2 de ellas son dominadas por Y_{true}. Por lo cual alcanza un 91,6% de efectividad. El NSGA, siendo inferior a los demás, posee solo un 78.5% de efectividad en las soluciones. En resumen, se puede concluir que para esta prueba los mejores algoritmos son el SPEA y el SPEA2 dado que encuentran la mayor cantidad de soluciones Pareto.

Prueba 4

	$\in Y_{true}$	$Y_{true} \succ$	$\lvert Y_{algoritmo} \rvert$	$\%(\in Y_{true})$
Y_{cNSGA2}	15	5	20	75
Y_{SPEA2}	17	1	18	94.4
Y_{NSGA}	9	5	14	64.2
Y_{NSGA2}	13	5	18	72.2
Y_{SPEA}	17	1	18	94.4

Tabla I: Comparación de las soluciones de los algoritmos con Y_{true}

Y_i \ Y_j	Y_{cNSGA2}	Y_{SPEA2}	Y_{NSGA}	Y_{NSGA2}	Y_{SPEA}
Y_{cNSGA2}		1	5	2	1
Y_{SPEA2}	5		5	5	1
Y_{NSGA}	4	0		3	0
Y_{NSGA2}	3	0	5		0
Y_{SPEA}	5	1	5	5	

Tabla II: Cantidad de soluciones en las cuales un algoritmo domina a otro

Se observa que el SPEA2 y el SPEA llegan a todas las soluciones en Y_{true}, mientras que el NSGA2 72.2% (13/18) y el CNSGA2 75%(15/20). El NSGA es el algoritmo que alcanza la menor cantidad de soluciones en Y_{true}.

El cNSGA2, obtiene un mayor número de soluciones que el resto de los algoritmos, sin embargo el 25% de ellas son dominadas por el conjunto Y_{treu}. Nuevamente, sobresalen los algoritmos SPEA y SPEA2.

7. Conclusión y Trabajos Futuros.

Se puede observar que los algoritmos NSGA2, CNSGA2, SPEA, SPEA2 poseen el mismo rendimiento cuando el grupo musticast es pequeño (Prueba 1 y 2). No obstante, el NSGA no llega a todas las soluciones en Y_{true}.

En la Prueba 3 y 4, se puede notar una diferencia entre los últimos cuatro algoritmos, notándose un mejor rendimiento de dos algoritmos, el SPEA y el SPEA2.

Es importante destacar que el Algoritmo cNSGA2, posee un parámetro R. Variando el mismo, pudimos obtener resultados tan buenos como el SPEA2, pero para cada problema deberíamos variar el mismo para obtener un resultado óptimo. El NSGA es absolutamente inferior a los demás algoritmos.

En trabajos futuros, se incluirán otras funciones objetivo como la utilización de los enlaces y el número de saltos de extremo a extremo. Así mismo, se espera realizar simulaciones con variaciones dinámicas del tráfico.

Agradecimiento

Se agradece a la Universidad Católica "Nuestra Señora de la Asunción", por su apoyo al proyecto "Ingeniería de Tráfico utilizando optimización multi-objetivo" que sirve de marco a la realización del presente trabajo.

Referencias

[1] A. Tanenbaum, *Computer Networks*, Prentice Hall, 2003.

[2] C. P. Ravikumar, y R. Bajpai, "Source-based delay bounded multicasting in multimedia networks", *Computer Communications*, Vol. 21, 1998, pg. 126-132.

[3] D. Van Veldhuizen, "Multiobjective Evolutionary Algorithms: Classifications, Analysis, and New Innovations", *Ph.D. dissertation, School of Engineering*, Air Force Institute of Technology.

[4] D. B. Fogel, editor Evolutionary Computation. "The Fossil Record Selected Reading on the History of Evolutionary Algorithms". *The institute of Electrical and Electric Engineers*, New York.

[5] N. Srinivas "Multiobjetive optimization using nondominated sorting in genetic algorithms". Master tesis, Indian Institute of Technology, Kuanpur. Inda, (1994).

[6] K. Deb, S. Agrawal, A. Pratap, y T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objetive Optimization: NSGAII. http://www.lania.mx/~ccoello/EMOO/EMOOconferences.html.

[7] K. Deb y T. Goel. Controlled Elitist Non-Dominated Sorting Genetic Algorithms for Better Convergence. http://www.lania.mx/~ccoello/EMOO/EMOOconferences.html.

[8] 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, pp. 257-271.

[9] E. Zitzler, Marco Laumanns, y Lothar Thiele. SPEA2. Improving the Strength Pareto Evolutionary Algorithm for MultObjetive Optimization. http://www.lania.mx/~ccoello/EMOO/EMOOconferences.html.

[10] C. Coello Coello. Evolutionary Multiobjetive Optimization: Current and Future Challenges. http://www.lania.mx/~ccoello/EMOO/EMOOconferences.html.

[11] J. Crichigno y B. Barán, "Multiobjective Multicast Routing Algorithm". *IEEE ICT'2004*, Ceará, Brasil, 2004.

[12] A. Dias, De Vasconcelos J. "Multiobjective Genetic Algorithms Applied to Solve Optimization Problems". IEE Transactions on Magnetics, Vol. 38, No. 2, March 2002