# COMPARACIÓN DE ALGORITMOS EVOLUTIVOS MULTI-OBJETIVOS EN UN AMBIENTE MULTICAST.

**Francisco Talavera Solalinde**

**PROYECTO FINAL DE TESIS**

**Orientador:**
**Prof. Benjamín Barán, D.Sc.**

**Ingeniería Electrónica**
**Facultad de Ciencias y Tecnología**
**Universidad Católica**
**"Nuestra Señora de la Asunción"**

**Asunción – Paraguay**
**Febrero del 2005**

***Este trabajo está dedicado a mis padres y a mis hermanas. Espero poder retribuirles en alguna medida su esfuerzo, su apoyo, su comprensión y sobre todo el amor que he recibido de ellos desde siempre.***

# *Agradecimientos*

*Al profesor Benjamín Barán, modelo de profesional y por sobre todo, excelente persona. Sin su constante apoyo y orientación este proyecto no hubiera llegado a buen puerto. Su excepcional capacidad de superación frente a los problemas e inigualable visión positiva han inyectado vida y dinamismo al quehacer diario de las personas que tenemos la gran suerte de pertenecer a su grupo de investigación.*

*A la Lic. Blanca Troche de Trevisán, Directora del Centro Nacional de Computación, por haber hecho posible que en dicha institución puedan ser llevados a cabo trabajos de investigación de esta naturaleza. Con personas como ella sin dudas nuestro querido país tendrá un futuro luminoso, de pleno desarrollo y madurez.*

*A los queridos compañeros del Centro Nacional de Computación, Ing. Jorge Crichigno, Lic. Joel Prieto, Lic. Diego Pinto, Pedro Gardel, M. Sc. Osvaldo Gomez y Lic. José Fernández. Deseo agradecerles especialmente por su constante apoyo y colaboración en las diferentes instancias del desarrollo del presente trabajo, así también por el excelente ambiente de trabajo del que me han permitido ser parte.*

*Finalmente una mención especial a todos mis profesores, compañeros y amigos de la Facultad de Ciencias y Tecnología, con quienes he compartido estos años de formación. Mi deseo es que el conocimiento y la experiencia que hemos adquirido a través de nuestro paso por esta facultad nos sirvan para forjarnos una vida digna, honesta y plena, al servicio de nuestras familias y de nuestro país.*

# Índice General

# Índice de Figuras

# Índice de Tablas

# Siglas y Abreviaturas

| | |
|---|---|
| $(i,j)$ | enlace entre los nodos $i$ y $j$. |
| $\alpha$ | utilización máxima de los enlaces de la red. |
| $\alpha C$ | política estática alfa costo. |
| $\alpha D_A$ | política estática alfa retardo. |
| $\alpha_c$ | promedio de la utilización de los enlaces actual |
| $\alpha_d$ | promedio de la desviación estándar de $\alpha_T$ |
| $\alpha_p$ | utilización máxima promedio. |
| $\alpha_T$ | utilización máxima de los enlaces de un árbol. |
| $\phi$ | demanda de tráfico del grupo multicast. |
| $\phi_{\Delta t}$ | demanda constante en el lapso de tiempo $\Delta t$. |
| $\Psi$ | número de pedidos multicast. |
| $\pi$ | número de grupos multicast activos en un instante. |
| $a$ ,$b$ $y$ $c$ | valores utilizados en la política semi-estática. |
| $C\alpha$ | política estática costo alfa. |
| $c_{ij}$ | costo del enlace $(i,j)$. |
| $C_c$ | promedio del costo actual |
| $C_p$ | costo promedio. |
| $C_d$ | promedio de la desviación estándar de $C_T$ |
| $C_T$ | costo del árbol. |
| $d(p_T(s, n))$ | retardo del camino $p_T(s, N)$, dado por la suma de los retardos de los enlaces que conforman el camino. |
| $D_A$ | retardo promedio de un árbol. |

| | |
|---|---|
| $D_A\alpha$ | política estática retardo alfa. |
| DC | política dinámica al origen de coordenadas. |
| $D_c$ | promedio del retardo actual |
| $D_d$ | promedio de la desviación estándar de $D_A$ |
| $d_{ij}$ | retardo (delay) del enlace $(i,j)$. |
| $D_M$ | retardo máximo extremo a extremo. |
| $D_m$ | límite de retardo definido por el administrador. |
| DP | política dinámica del peor caso aceptable. |
| $D_p$ | retardo promedio. |
| $D_t$ | demanda acumulada. |
| $\overline{D}$ | demanda promedio. |
| E | conjunto de arcos. |
| G | red modelada como un grafo dirigido |
| *N* | conjunto de nodos destinos del grupo multicast. |
| *Na* | número de grupos no aceptados. |
| $p_T(s, n)$ | camino que conecta el nodo fuente $s$ y el nodo destino $n \in N$. |
| P | población. |
| $P_t$ | conjunto de soluciones no dominadas de los algoritmos comparados. |
| *s* | nodo origen del grupo multicast. |
| *SE* | semi-estáticas. |
| $T(s,N)$ | árbol multicast con origen en $s$ y conjunto de nodos destinos $N$. |
| $t_{ij}$ | tráfico actual fluyendo a través del enlace $(i,j)$. |
| $\hat{Y}$ | unión de Frentes Pareto en la comparación estática. |
| $\boldsymbol{Y}_{cNSGA2}$ | frente Pareto del algoritmo cNSGA2. |
| $\boldsymbol{Y}_{NSGA}$ | frente Pareto del algoritmo NSGA. |

| | |
|---|---|
| $\boldsymbol{Y}_{\text{NSGA2}}$ | frente Pareto del algoritmo NSGA2. |
| $\boldsymbol{Y}_{\text{SPEA}}$ | frente Pareto del algoritmo SPEA. |
| $\boldsymbol{Y}_{\text{SPEA2}}$ | frente Pareto del algoritmo SPEA2. |
| $\hat{\boldsymbol{Y}}_{true}$ | aproximación del frente Pareto óptimo. |
| V | conjunto de nodos en la red. |
| $z_{ij}$ | capacidad del enlace $(i,j)$. |

# 1 Multicast y Redes de Computadoras

## 1.1 Introducción

La fusión de las computadoras y las comunicaciones ha tenido una profunda influencia en la forma en que los sistemas de cómputo se organizan. El concepto de "centro de computo" como cuarto con una gran computadora a la cual los usuarios traían sus trabajos para procesar es ahora totalmente obsoleto. El viejo modelo de una sola computadora que atendía todas las necesidades de computación de la organización ha sido reemplazado por uno en el cual un gran número de computadoras separadas pero interconectadas hacen el trabajo. Estos sistemas se llaman **redes de computadoras** [TAN03].

La Internet tiene sus orígenes en la ARPANET, que era una red experimental de datos financiada por los Estados Unidos en los comienzos de la década de los 60. El objetivo principal del gobierno de los Estados Unidos era la creación de una red robusta con capacidad de mantenerse funcionando aún cuando parte de la misma esté sujeta a fallas. Con esta idea, la ARPANET fue construida basándose en un modelo de datagramas, en el cual cada paquete de datos es enviado en forma independiente a su destino. La red de datagramas, además de estar basada en una idea simple, tiene la habilidad de adaptarse automáticamente a cambios en la topología [TAN03].

Durante muchos años, Internet ha sido empleada por investigadores para realizar intercambio de información. Las aplicaciones más populares eran el acceso remoto, la transferencia de archivos y el correo electrónico (e-mail), para este tipo de aplicaciones el modelo de datagramas funciona bien. No obstante, con el tiempo se han creado un conjunto de nuevas aplicaciones con diferentes requerimientos de calidad de servicio (QoS) distintos a aquellos para los cuales Internet fue creada, como ser un requerimiento de ancho de banda mínimo garantizado para que ciertas aplicaciones funcionen bien. El modelo de datagramas en el cual está basado Internet tiene muy poca capacidad de manejo de tráfico de datos dentro de la red y consecuentemente no puede garantizar una capacidad de ancho de banda mínima en bps a los usuarios. Cuando una aplicación

intenta alcanzar un sitio Web o hacer una llamada a través de Internet, algunas partes de la red pueden estar tan congestionadas que los paquetes no pueden llegar a destino y simplemente son desechados. Por lo tanto, las aplicaciones no pueden funcionar en forma adecuada [WAN01].

Un problema que aparece en las redes de computadoras es el retardo que sufren los paquetes hasta alcanzar al destino. Cuando las aplicaciones como transmisiones de radio o TV tienen un retardo que sobrepasa ciertos niveles, estas carecen de utilidad. Entonces, para que las redes de computadoras actuales sean capaces de manejar tráfico de datos considerando parámetros de QoS, no sólo deben ser capaces de garantizar una cantidad de ancho de banda mínimo a las aplicaciones, sino que deben proveer el servicio teniendo en cuenta el retardo a los nodos destinos [WAN01].

Las redes basadas en el modelo de datagramas históricamente han ofrecido un servicio conocido como servicio del mejor esfuerzo (*best-effort*). *Best-effort* simplemente representa el servicio más simple que una red puede proveer; no hay ninguna forma de asignación de recursos en la red. Este servicio consiste en que todos los paquetes son tratados de igual forma, sin niveles de servicio, requerimientos, reservas o garantías. Este servicio solo ofrece hacer el mejor esfuerzo para lograr la comunicación. Cuando un enlace está congestionado, los paquetes simplemente son descartados. Los objetivos principales de este modelo fueron la escalabilidad y el mantenimiento de la conexión bajo fallas. Los algoritmos encargados de seleccionar un camino (*enrutar)* entre dos nodos de la red simplemente seleccionan el camino más corto considerando una métrica simple como número de saltos o retardo. Claramente este enfoque no es adecuado para soportar asignación de recursos como ser una capacidad mínima garantizada. Por ejemplo, para llevar a cabo una reserva de recursos, es necesario encontrar un camino desde el origen al destino con cierta cantidad de ancho de banda a lo largo del camino. Pero el protocolo *Internet Protocol* (IP) utilizado en las redes de datagramas no cuenta con dicha información. Por lo tanto, el uso de dichos algoritmos puede conllevar a una alta tasa de rechazos de solicitudes de conexión y a una mala distribución del tráfico sobre la red. Esto es, mucho tráfico fluyendo por aquellos enlaces que conforman los caminos más cortos, creando cuellos de botella, mientras otros enlaces son apenas utilizados [WAN01].

Entonces queda claro que la optimización de recursos requiere capacidades adicionales a aquellas proveídas por el modelo de datagramas. Si deseamos obtener una mejor utilización de la red, es necesario tener un control explícito sobre los caminos que deben atravesar los datos. Entonces, el flujo total de datos sobre la red puede ser acomodado de forma a minimizar los recursos utilizados, maximizar la probabilidad de enrutamiento de toda la demanda de tráfico y minimizar el costo que ello implica, considerando los parámetros de QoS que cada demanda de tráfico solicita [STA01].

Para seleccionar explícitamente los caminos, se puede utilizar MPLS (*MultiProtocol Label Switching*). Una red MPLS consiste en un conjunto de nodos llamados *Label Switch Routers* (LSR), que tienen la capacidad de enrutar paquetes utilizando etiquetas, las cuales son adheridas a cada paquete IP. Las etiquetas definen un flujo de paquetes entre dos nodos de la red, o, en el caso de una transmisión punto a multipunto, entre el nodo origen y el conjunto de nodos destinos. Por cada flujo, denominado *Forwarding Equivalent Class* (FEC), un camino específico (*Label Switch Path* - LSP) a través de la red es definido. La Figura 1.1 muestra un dominio MPLS, el cual puede representar un sistema autónomo (*Autonomous System* - AS) administrado por alguna entidad privada, como ser un proveedor de servicios de Internet (*Internet Service Provider* - ISP) [STA01].

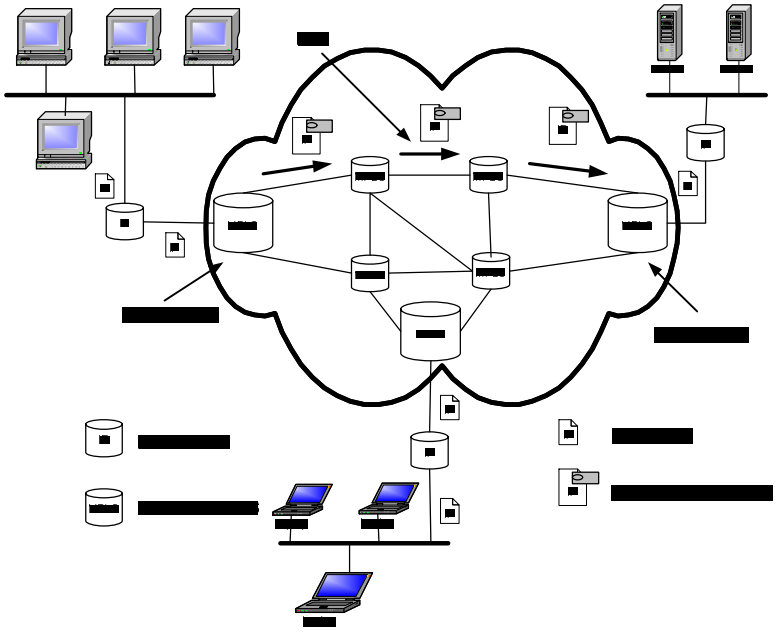


**Figura 1.1.** Dominio MPLS

El modelo DiffServ (*Differentiated Services*) en vez de distinguir flujos individuales LSP se clasifica los paquetes en categorías (según el tipo de servicio solicitado). A cada

categoría le corresponde un SLA (Service Level Agreement). Los usuarios pueden contratar o solicitar un determinado caudal en la categoría que deseen. Los routers tratan cada paquete según su categoría (que viene marcada en la cabecera del paquete).

## 1.2 Ingeniería de Tráfico

La elección dinámica de caminos para cada una de las demandas de tráfico en una red de computadoras, considerando el balanceo de carga de forma a evitar el congestionamiento de los enlaces, atendiendo los parámetros de QoS y optimizando los recursos de la red, es conocida como ingeniería de tráfico [STA01].

Existe un creciente número de aplicaciones punto a multipunto surgidas en Internet denominadas *multicast*, en las cuales se debe transmitir de un origen a varios destinos, entre estas aplicaciones, se pueden mencionar como ejemplo el video bajo demanda (*Video on-demand* - VoD), las tele-conferencias, las transmisiones de radio y TV, y la educación a distancia. En consecuencia, se ha incrementado el interés en algoritmos de enrutamiento explícito punto a multipunto para ingeniería de tráfico. Esto se debe principalmente a que la pila de protocolos tradicionalmente usada en las redes de datagramas TCP/IP no puede hacer ingeniería de tráfico. Estos inconvenientes pueden ser superados utilizando por ejemplo MPLS y RSVP-TE (*Resource Reservation Protocol – Traffic Engineering*) [OOM02]. RSVP-TE es un protocolo de señalización utilizado para la reserva de recursos de *ancho de banda* a lo largo de un camino que conecta un nodo de ingreso y otro de egreso, lo cual permite garantizar servicios con QoS. De esta forma, la ingeniería de tráfico multicast en una red MPLS puede llevarse a cabo en dos pasos:

1 Encontrando el camino desde el nodo origen a cada uno de los nodos destinos utilizando un algoritmo apropiado para tal efecto.
2 Reservando los recursos a lo largo de los caminos hallados en el paso 1, utilizando RSVP-TE [DON02].

Los caminos deben ser elegidos de forma a acomodar todo el tráfico a un costo mínimo en términos de recursos, considerando parámetros de QoS como retardo máximo de extremo a extremo y retardo medio, y balanceando la carga de forma a evitar congestionamientos en la red. De este hecho se deriva que la ingeniería de tráfico

multicast debe considerar más de una métrica, y por lo tanto, el problema de enrutamiento multicast en redes de computadora puede ser formulado como un Problema de Optimización Multiobjetivo (*Multiobjective Optimization Problem* – MOP) [COE96].

Motivado por la necesidad creciente por parte de los administradores de AS de contar con herramientas informáticas eficaces para dicha tarea, y 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, cNSGA2, SPEA y SPEA2, en un ambiente estático y en un ambiente de tráfico dinámico, donde entran y salen grupos multicast, considerando que los MOEAs proporcionan un conjunto de soluciones de compromiso. En la literatura tampoco queda claro la política de elección de la solución a escoger en un contexto dinámico, por lo que se propone un conjunto de 7 políticas de selección de soluciones.

Habiendo introducido conceptos esenciales utilizados a lo largo del libro, el Capítulo 1 continúa de la siguiente manera: en la Sección 1.3 se da la definición formal de multicast, unicast y broadcast, y se ejemplifica la diferencia entre cada una de estas formas de comunicación. En la Sección 1.4 se definen las métricas de optimización consideradas en este trabajo. Posteriormente, en la Sección 1.5 se presentan los trabajos relacionados a problemas de enrutamiento multicast en redes de computadoras. Por último, en la Sección 1.6 se da una breve introducción al enfoque multiobjetivo considerado en este trabajo para la resolución del problema de enrutamiento multicast en redes de computadoras.

## 1.3 Multicast

Seleccionar una ruta entre dos nodos de una red es una de las tareas más importantes en las redes de computadoras. Esta es llevada a cabo por los algoritmos de enrutamientos, que se encargan de seleccionar la ruta más adecuada.

Las formas de comunicación tradiciones son básicamente tres y se definen a continuación [KOM93a].

1. Unicast: Transmisión de datos desde una fuente a un destino.
2. Broadcast: Transmisión de datos desde una fuente a todos los posibles destinos.
3. Multicast: Transmisión de datos desde una fuente a un conjunto de destinos.

Implementando unicast o broadcast se puede realizar una transmisión multicast. No obstante, estas soluciones no son eficientes. Enviar datos a todos los nodos de la red para alcanzar a un subconjunto puede producir en la mayoría de los casos un desperdicio grande de recursos. Esta situación es mostrada en la Figura 1.2, donde el nodo 0 desea transmitir a los nodos 2, 3 y 6, pero eventualmente también transmite a otros nodos que no desean la información, como el nodo 4 y 5.

Alcanzar cada uno de los nodos destinos desde el origen utilizando secuencia de unicast también en la mayoría de los casos puede ser indeseable. El principal motivo nuevamente es el desperdicio de recursos utilizados. Por ejemplo, los caminos 0 a 3 y 0 a 6 comparten el enlace que conecta los nodos 0 y 6. En este caso, el mismo dato atravesaría el enlace dos veces. La Figura 1.3 muestra dicha situación.

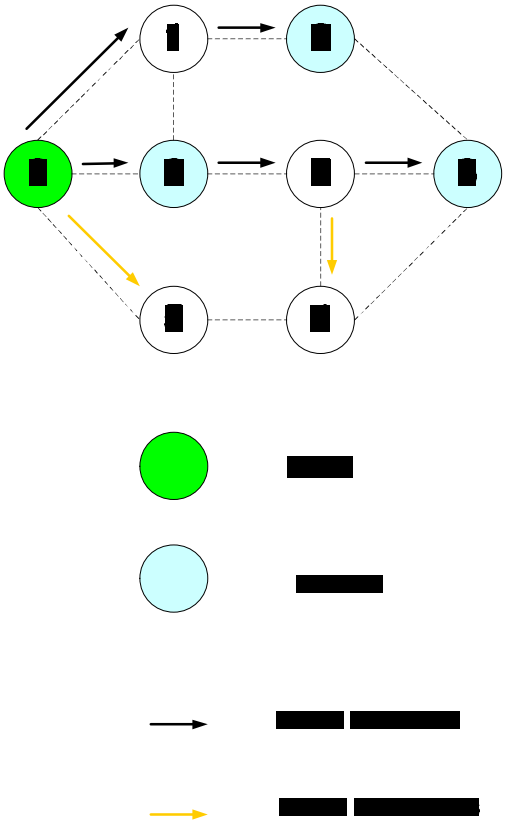


**Figura 1.2.** Ejemplo de utilización de broadcast para realizar multicast

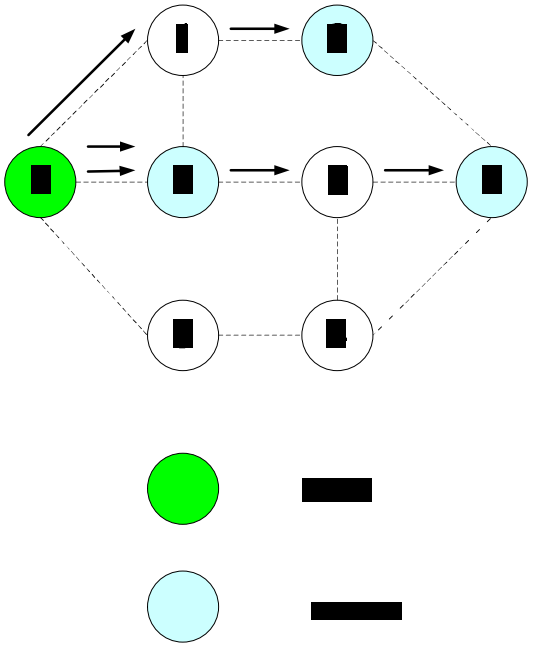


**Figura 1.3.** Una secuencia de conexiones unicast para realizar multicast

Un árbol de expansión es la estructura apropiada para realizar multicast. El árbol de expansión consta de un nodo fuente desde el cual cada uno de los nodos destinos puede ser alcanzado, a través de un solo camino. Es decir, consiste en un grafo sin ciclos [KOM93a]. El árbol de expansión multicast evita el desperdicio de recursos como lo hacen las técnicas mencionadas anteriormente. La Figura 1.4 muestra un árbol que puede ser utilizado para la distribución de datos desde el nodo 0 a los nodos 2, 3 y 6. Note que los datos atraviesan los enlaces que pertenecen al árbol una sola vez. Además, no se producen envíos innecesarios de datos.

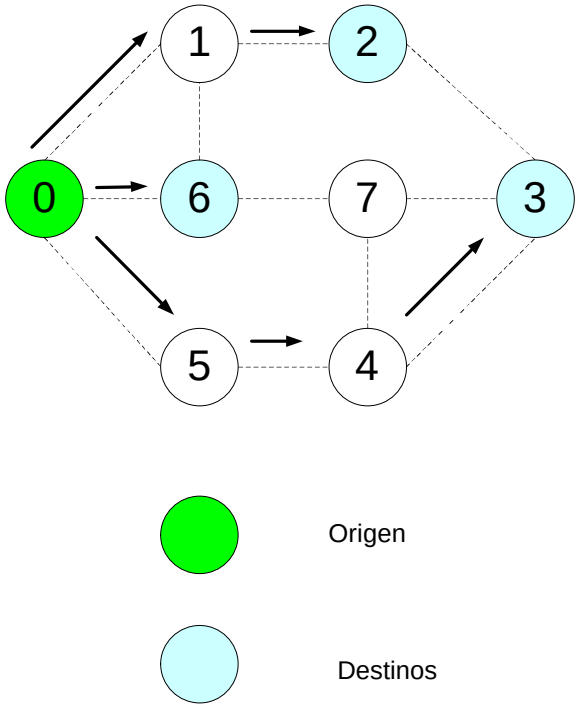



**Figura 1.4.** Un árbol multicast con origen en 0 y destinos en 2, 3 y 6

## 1.4 Métricas de Optimización Multicast

Un algoritmo de enrutamiento multicast se encarga de construir un árbol multicast que conecta el nodo origen a cada uno de los destinos de la comunicación multicast. El mismo puede optimizar un conjunto de distintas métricas. Además, la aplicación para la cual se construye el árbol puede tener diversos requerimientos de QoS como retardo máximo de extremo a extremo limitado, retardo medio acotado o ancho de banda mínimo requerido. Seguidamente son presentadas las distintas métricas a ser consideradas en la ingeniería de tráfico multicast, tratada en este trabajo.

### 1.4.1 Utilización Máxima de los Enlaces

La utilización máxima de los enlaces es uno de los objetivos principales de la ingeniería de tráfico multicast, consiste en encontrar los caminos con una cierta

capacidad entre el nodo origen y el conjunto de nodos destinos, evitando el congestionamiento en la red y balanceando la carga en la misma. El enfoque tradicional utilizado en la práctica para lograr dicho objetivo es el enrutamiento a través de los caminos con menor utilización de los enlaces, donde la utilización de un enlace está definida como el tráfico que fluye a través de él, dividido su capacidad. Por su parte, la utilización máxima de un camino ($\alpha_p$) se define como la utilización máxima de un enlace en el camino del origen al destino [SEO02]. El enrutamiento a través de dichos enlaces evita la congestión de una red sobrecargada, también reduce la pérdida de paquetes y el retardo total [DON04]. Además, la probabilidad de rechazos de posteriores solicitudes de tráfico disminuye [SEO02]. Dado que los algoritmos de enrutamiento multicast deben conectar el nodo origen con cada uno de los destinos, es deseado que el árbol multicast esté compuesto por aquellos enlaces de menor utilización. Esto es, si la utilización máxima de los enlaces de un árbol es definido como la utilización de su enlace más utilizado o utilización máxima de sus enlaces ($\alpha_T$), se desea hallar el árbol que minimice $\alpha_T$ [SEO02]. A continuación se presenta un ejemplo en el cual se demuestra la utilidad del enrutamiento considerando la métrica $\alpha_T$.

**Ejemplo 1.1.** Considere la Figura 1.5 (a), donde cada enlace tiene una capacidad de 10 Mbps y un tráfico actual asignado, en Mbps, dado en la Figura. Suponga que el nodo 2 desea hacer una transmisión de 1 Mbps a los nodos 5 y 6, y que el árbol hallado para dicha transmisión es el mostrado en la Figura 1.5 (b). En dicha Figura también se muestra la utilización de cada enlace luego de haber sido enrutada la citada demanda multicast. Si bajo estas circunstancias es generada una nueva solicitud multicast de 2 Mbps con origen en el nodo 2 y destinos en los nodos 0 y 1, dicha solicitud sería bloqueada por falta de recursos. Esta situación podría haber sido evitada si la transmisión de datos del primer grupo multicast se hiciera a través del árbol con menor $\alpha_T$. Dicho árbol es mostrado en la Figura 1.5 (c).

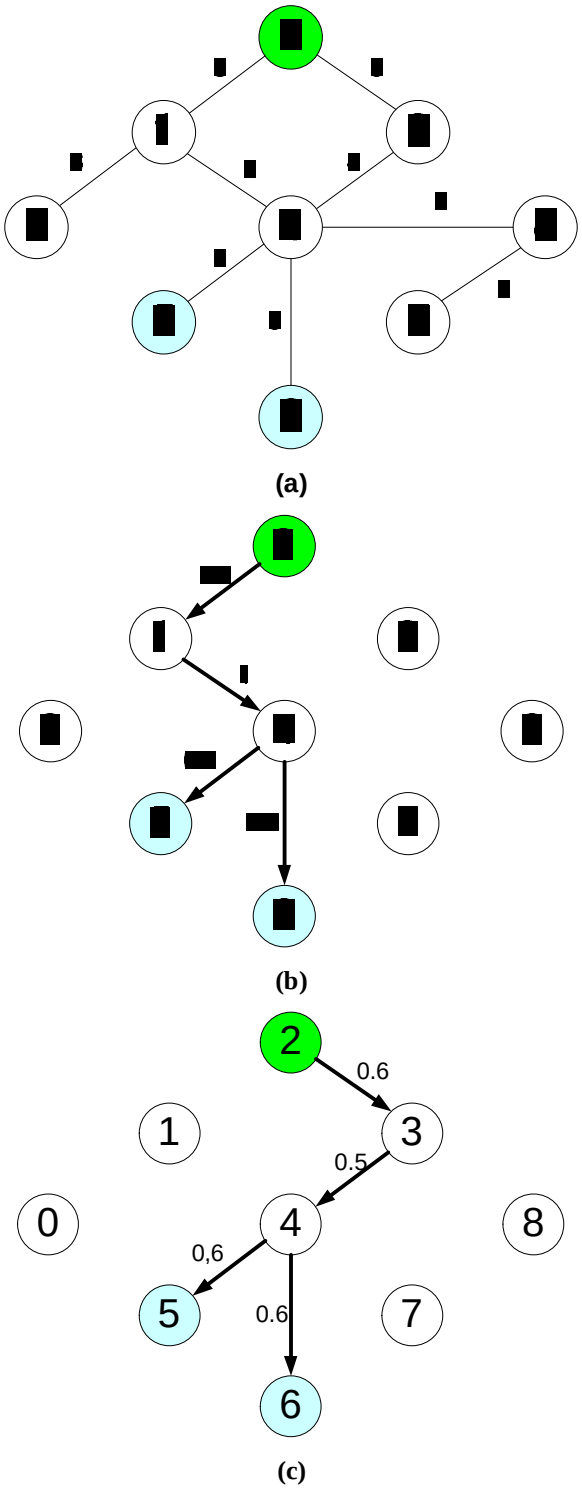



**Figura 1.5.** (a) Red de 9 nodos. Cada enlace tiene una capacidad de 10 Mbps y un tráfico actual fluyendo a través de él, en Mbps, dado en la misma Figura. (b) Árbol con origen en 2, conjunto destino {5, 6} y demanda de tráfico de 1 Mbps. La utilización de cada enlace es mostrada en la misma Figura. Para este árbol, $\alpha_T = 1$. (c) Árbol alternativo. Para esta opción, $\alpha_T = 0.6$

Un problema de asignación de flujo fue mostrado en el Ejemplo 1, el mismo puede ser optimizado si todas las demandas multicast son conocidas *a priori*. Debido al dinamismo con que son generadas las demandas de tráfico en las redes de computadoras reales, es imposible conocer todas ellas de antemano. Por lo tanto, es razonable hacer optimizaciones locales teniendo en cuenta la utilización máxima de los enlaces del árbol en el momento en que cada grupo multicast solicita su demanda de tráfico, de forma a balancear el tráfico total sobre la red de la mejor manera posible.

El enrutamiento a través de los enlaces de menor utilización es útil tanto para el balanceo de carga como para la reducción de la congestión, pero este podría desperdiciar recursos debido a la longitud de las rutas en término de saltos (i.e. suma de ancho de banda asignado en cada enlace a lo largo del camino) [SEO02]. Esta situación es mostrada en el Ejemplo 1.2.

**Ejemplo 1.2.** En la Figura 1.6 tenemos la red de la *National Science Foundation* (NSF) [BAR00]. Sobre cada enlace se tienen los valores de utilización de los enlaces (i.e. el enlace que conecta los nodos 7 y 8 tiene una utilización de 0.2). Suponga que el nodo 0 desea transmitir un flujo de datos al nodo 2. El camino de menor utilización de los enlaces está representado por un conjunto de flechas.

Note que la utilización máxima para este árbol multicast es $\alpha_p = 0.2$. Por otra parte, el camino directo desde 0 a 2 tiene una utilización máxima $\alpha_p = 0.25$, apenas superior a la alternativa mostrada en la Figura, pero un consumo de ancho de banda mucho menor. Si tenemos en cuenta el número de saltos, la solución mostrada en la Figura 1.6, no sería óptima. En dicho caso, elegiríamos la solución del camino directo 0 a 2, con un $\alpha_p$ apenas mayor. Entonces podemos concluir que una métrica a tener en cuenta es la utilización de los recursos de la red.

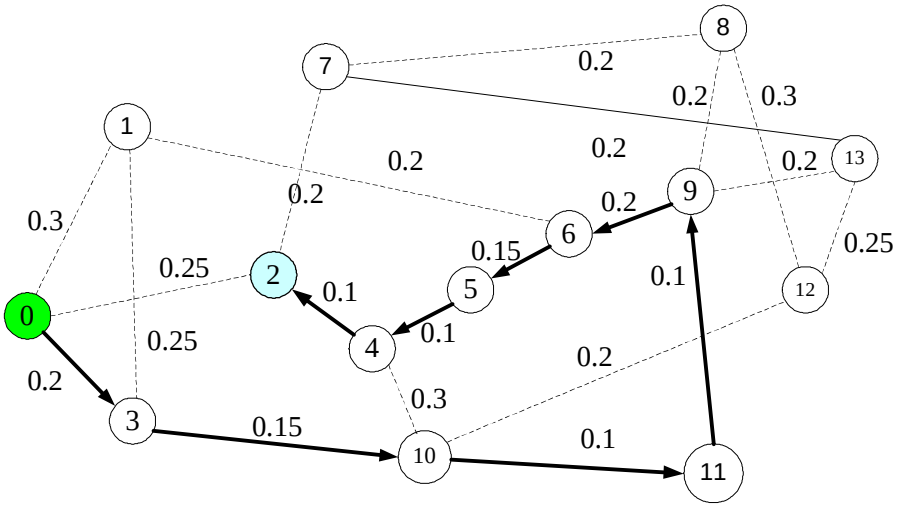



**Figura 1.6.** Red de la NSF, con la utilización de los enlaces como métrica. El camino de menor utilización de los enlaces, con $\alpha_p$ = 0.2, es mostrado en la Figura. Note que el camino directo 0-2 tiene $\alpha_p$ = 0.25, pero un costo en términos de recursos utilizados mucho menor.

### 1.4.2 Costo del Árbol Multicast

Como fue concluido en el Ejemplo 1.2 la utilización de los recursos también debería ser considerada como una métrica de optimización al construir un árbol multicast. Una forma general de optimizar dicha métrica es minimizando el costo del árbol. El costo del árbol $C_T$ está dado por la suma de los costos de los enlaces que lo componen. El mismo puede tener diferentes significados: puede representar alguna cantidad monetaria, como también ser proporcional al consumo total de ancho de banda del grupo multicast. Cualquiera sea el significado, aquellas soluciones de menor costo siempre son deseables. Por ejemplo, en una red cuyos enlaces tienen costo unitario, la minimización del costo del árbol multicast conduce al árbol con menor número de enlaces, y por lo tanto, al de menor consumo de ancho de banda.

La minimización del costo del árbol optimiza una métrica muy importante asociada a los recursos de la red (i.e. ancho de banda consumido), pero en aplicaciones como e-learning, tele-conferencias, video bajo demanda y transmisiones de audio y/o video, se necesitan árboles de bajo retardo medio y/o bajo retardo máximo de extremo a extremo. La Figura 1.7 muestra el árbol de costo mínimo para el grupo multicast conformado por el nodo 0 como origen y el conjunto destino {2, 6, 9, 11}. El retardo y el costo de cada enlace están representados respectivamente por el valor próximo al mismo en la Figura 1.7.

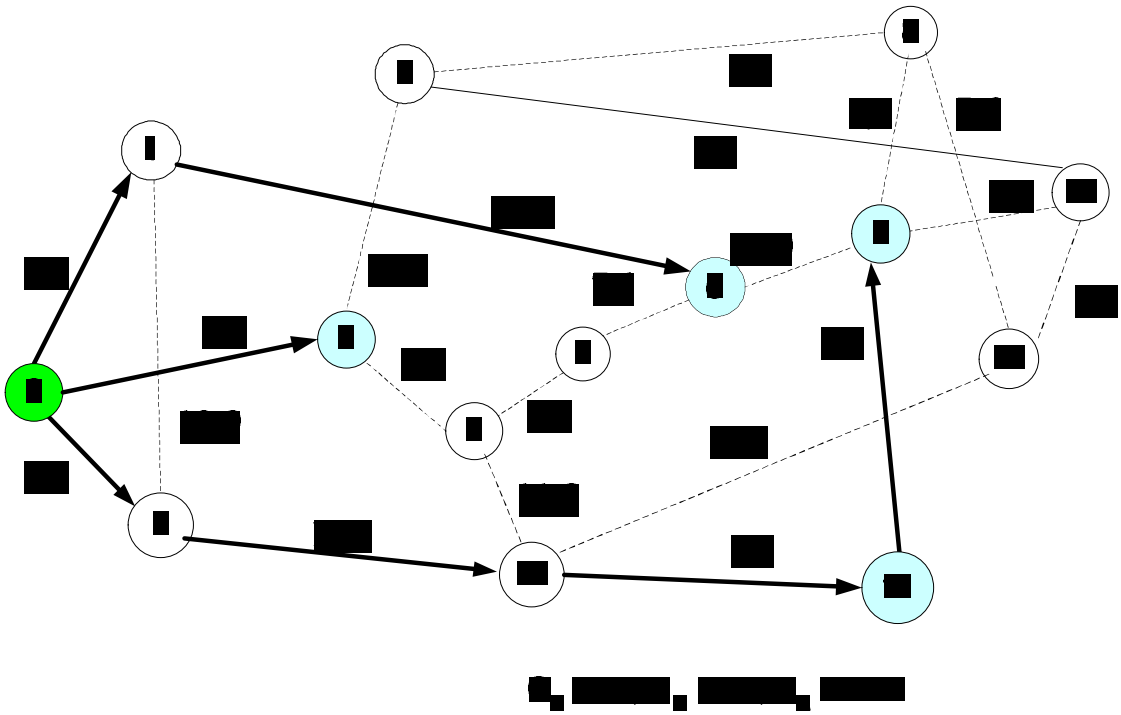


**Figura 1.7.** Árbol multicast de costo mínimo.

La solución mostrada en la figura 1.7 es óptima cuando se considera la métrica de costo. Sin embargo, esta puede no ser de utilidad para aquellas aplicaciones sensibles al retardo, como las citadas anteriormente.

### 1.4.3 Retardo Medio y Retardo Máximo de Extremo a Extremo

Enrutar a través del camino de menor retardo es la forma tradicional para optimizar las métricas de retardo (*delay*) medio ($D_A$) y retardo máximo de extremo a extremo ($D_M$). La extensión natural al caso multicast es la transmisión a través del árbol compuesto por los caminos de menor retardo desde el nodo fuente a los nodos destinos. El mismo es conocido como el árbol del camino más corto (*Shortest Path Tree* - SPT). La Figura 1.8 ilustra dicha solución para el mismo grupo multicast mostrado en la Figura 1.7. El par de números sobre cada enlace corresponde al retardo y al costo del mismo. El retardo promedio $D_A$ está dado por la suma de los retardos de los caminos a cada uno de los nodos destinos dividido el número de nodos destinos. Es importante destacar que, con estos valores de retardo, el árbol de costo mínimo de la Figura 1.7 tiene $D_M$ = 39 y $D_A$ = 21.6, mientras que el SPT tiene $D_M$ = 35 y $D_A$ = 20.8. El retardo máximo y el retardo medio

del árbol de costo mínimo son mayores que el obtenido por SPT y dicha solución podría no ser de utilidad para aquellas aplicaciones sensibles a estas métricas. En contrapartida, el costo del árbol de la Figura 1.7 es de apenas 23, mientras que el costo del SPT es 35. Claramente, existe una relación de compromiso entre estas métricas.

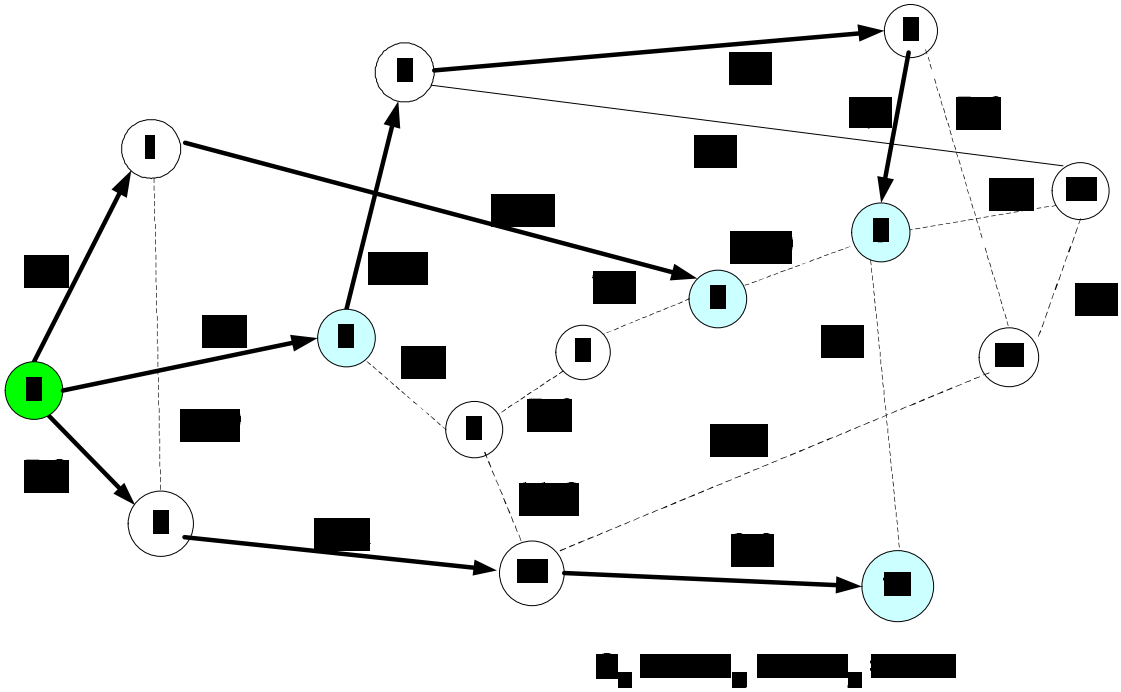


**Figura 1.8.** Árbol multicast óptimo cuando solo $D_M$ es considerado.

## 1.5 Trabajos Relacionados

En la Introducción se ha mencionado que el enfoque típico utilizado en Internet para enrutar paquetes es el enrutamiento a través del camino más corto, considerando la métrica de retardo o número de saltos. En redes de computadores la mayor parte de los algoritmos utilizados son variantes del algoritmo de Dijkstra [DIJ59]. El mismo encuentra el camino de menor costo, en términos de retardo o saltos, desde un nodo fuente a un nodo destino. Si aplicamos el algoritmo a un grupo multicast, este produciría un árbol cuyos caminos desde el nodo origen al conjunto de nodos destinos son óptimos en términos de la métrica utilizada. En Internet el estándar más utilizado es el *Open Shortest Path First* (OSPF) [STA00], soportado por la mayoría de los *Routers* hoy comercializados.

Hallar el árbol de menor costo es conocido como Problema del árbol de Steiner, y es sabido que la complejidad de dicho problema es NP-completa [KOM93a]. La dificultad de hallar el árbol de costo mínimo radica en la existencia de un número de árboles que crece exponencialmente con el número de nodos de la red, y la solución óptima es obtenida por búsqueda exhaustiva [KOM93a]. Debido al elevado tiempo de cómputo que consume hallar la solución óptima, han sido propuestos varios algoritmos heurísticos para la resolución de dicho problema. Uno de los más famosos algoritmos fue propuesto por Kou, Markowsky y Berman (algoritmo KMB) [KOU81]. Dada una red de computadoras modelada como un grafo G y un grupo multicast, KMB transforma el grafo G a otro G'. G' consiste en un grafo completamente conexo, en el que cada enlace representa el camino de menor costo en G entre los dos nodos involucrados. El número de nodos de G' es igual al tamaño del grupo multicast. Utilizando G', el algoritmo construye el árbol de expansión de menor costo. Luego, el árbol en la red original es obtenido transformando el árbol de expansión en G' a los caminos en G que ellos representan.

En [HWA00], Hwang et al. presentaron un algoritmo genético simple que minimiza el costo del árbol multicast. Los autores propusieron representar los cromosomas como cadenas de enteros, en los cuales cada elemento de la cadena es usado para indexar una tabla de rutas de un nodo destino. La longitud de los cromosomas es igual al número de receptores multicast. La tabla de rutas para un nodo destino consiste en un conjunto de $R$ posibles rutas que conectan el nodo origen y el destino, donde $R$ es un parámetro del algoritmo. Cada elemento de la cadena (gen) puede tomar un valor entero entre 1 y $R$, el cual representa (indexa) una ruta dada de la tabla de rutas al nodo destino correspondiente. La adaptabilidad (o *fitness*) de un individuo es inversamente proporcional al costo del árbol que él representa. Los autores mencionan la manera de extender la propuesta a situaciones que requieran restricciones como retardo de extremo a extremo acotado y/o ancho de banda mínimo requerido para la transmisión de datos.

Los algoritmos que solo optimizan la función de costo pueden proveer soluciones que violan las restricciones de retardo máximo impuestas por ciertas aplicaciones sensibles a las mismas. Algunos algoritmos añaden una restricción de retardo máximo al problema del árbol de costo mínimo para considerar la métrica de retardo. Este problema fue formulado por primera vez por Kompella, Pasquale y Polyzos [KOM93b]. Los autores propusieron un algoritmo heurístico denominado KPP basado en el algoritmo KMB. KPP

extiende el alcance de KMB a problemas con restricciones de retardo, teniendo en cuenta el retardo a cada nodo destino en el momento de formar la red completamente conexa. El retardo máximo permitido es determinado *a priori*. La importancia de este trabajo radica en la formulación, por primera vez, de un problema de enrutamiento multicast en redes de computadoras en el cual más de una métrica independiente debe ser considerada.

Nuevas propuestas basadas en algoritmos genéticos fueron publicadas en los últimos años [RAV98, XIA99, ZHE01, ARA02]. El primero de ellos fue propuesto por Ravikumar et al. [RAV98], quienes presentaron un algoritmo genético simple en el cual un cromosoma consiste en el conjunto de enlaces que conforman el árbol que él representa. Para cada individuo el *fitness* es directamente proporcional a la calidad del individuo, siendo un buen individuo aquel árbol de bajo costo que cumple con la restricción de retardo máximo impuesto a priori. El operador de cruzamiento consiste en los siguientes pasos: 1- hallar los enlaces comunes de ambos padres, 2- crear sub-árboles a partir de los enlaces comunes y 3- generar un hijo a través de la reconexión de los diferentes sub-árboles. La reconexión se lleva a cabo según el siguiente criterio: en caso que al menos uno de los padres cumpla con la restricción de retardo, la reconexión se hace a través del camino de menor costo entre los sub-árboles. En caso contrario, los sub-árboles se conectan a través del camino de menor retardo. El operador de mutación consiste en eliminar del árbol un enlace elegido al azar, y reconectar los sub-árboles de la misma forma que lo hace el operador de cruzamiento. Este trabajo fue mejorado consecutivamente por Zhengying et al. [ZHE01] y por Araujo et al. [ARA02].

La imposición de un valor máximo *a priori* es la principal desventaja de los enfoques basados en restricciones de retardo máximo, el mismo puede descartar soluciones de muy bajo costo, con retardo máximo solo apenas superior a la restricción dada *a priori*.

El progresivo aumento de redes MPLS y los protocolos de señalización como RSVP-TE, los cuales en conjunto permiten el encaminamiento explícito y la reserva de recursos a través de los caminos, ha motivado el estudio de nuevos algoritmos con esquemas alternativos de balanceo de carga y optimización de recursos. Para llevar a cabo dicha tarea, el objetivo adoptado en recientes publicaciones [SEO02, DON04] es la minimización de la utilización máxima de los enlaces de la red ($\alpha$). Para resolver el problema dinámico de ingeniería de tráfico multicast, en el cual las solicitudes de tráfico

llegan una después de otra, Seok et al. [SEO02] propusieron un algoritmo heurístico basado en la extensión del algoritmo de Dijkstra [DIJ59] que minimiza $\alpha$. Aunque esta propuesta es útil para minimizar la utilización máxima de los enlaces de la red y reducir el congestionamiento sobre la misma, el ancho de banda consumido puede ser desperdiciado debido a posibles rutas largas en términos de saltos. Por lo tanto, los autores restringen la longitud de las rutas a un valor máximo dado *a priori*, de forma a hallar un árbol multicast en el cual el número de saltos a los nodos destinos esté acotado. De esta manera, los autores intentan optimizar, además de la utilización máxima de los enlaces, el consumo de ancho de banda. El algoritmo propuesto en este trabajo puede ser dividido en dos pasos: 1- modificación del grafo original a uno en el cual el número de saltos desde el nodo origen a cualquier otro nodo de la red esté acotado por un valor dado a priori; 2- encontrar el árbol multicast que minimice la utilización máxima de los enlaces, utilizando para ello el algoritmo del camino de menor utilización máxima de los enlaces. Este trabajo demostró que el tráfico multicast puede ser mejor balanceado y por lo tanto la probabilidad de rechazos de solicitudes por falta de recursos puede ser disminuida cuando la métrica de enrutamiento utilizada es la utilización de los enlaces. La formulación del problema claramente demuestra la característica multiobjetivo de los problemas de enrutamiento multicast, pues existe más de un objetivo a optimizar.

En [DON04], Donoso et al. propusieron un esquema de ingeniería de tráfico multicast multi-árbol. Con este enfoque, el flujo de datos desde el nodo fuente a los nodos destinos es enviado a través de distintos árboles, de forma a balancear la carga sobre la red. El esquema no solamente tiene en cuenta la utilización máxima de los enlaces, sino también el número total de saltos, el consumo total de ancho de banda y el retardo total de extremo a extremo. El método propuesto minimiza una función objetivo compuesto por la suma ponderada de las cuatro métricas citadas. Debido a la naturaleza NP-completa del esquema propuesto, los autores también propusieron un algoritmo heurístico para la resolución del problema. Este consiste de dos etapas: 1- obtener un grafo modificado. En esta etapa, todos los caminos posibles desde el nodo fuente a cada uno de los nodos destinos son hallados. Luego, para cada uno de los nodos destinos, para cada camino al nodo destino, para cada nodo en dicho camino, un valor de distancia basado en el número de saltos, ancho de banda consumido y retardo es calculado; 2- para cada destino multicast, hallar los caminos requeridos para la transmisión de flujo en el grafo modificado. Es importante notar que este trabajo ya considera el estudio de algoritmos

evolutivos como trabajo futuro, pues la característica multiobjetivo del problema y las relaciones de compromiso entre los distintos objetivos son resaltados en las pruebas de simulación hechas por los autores.

En [CRI04a] y [CRI04b], Crichigno et al. presentan una algoritmo de ruteamiento multicast multiobjetivo MMA1, donde optimizan simultáneamente el costo del árbol, el retardo máximo extremo a extremo, el retardo promedio y la utilización máxima de los enlaces. Mas tarde, en [CRI04c], se propone un algoritmo de ruteamiento multicast multiobjetivo MMA2 que a diferencia de la versión anterior, codifica a los algoritmos como grafos en lugar de utilizar tablas de ruteo.

Dada la naturaleza multiobjetivo del problema de enrutamiento multicast en redes de computadoras, la relación de compromiso entre las distintas métricas a optimizar (presentadas en la Sección 1.4) y la imperiosa necesidad de optimizar cada una de ellas en forma simultánea, este trabajo propone formular el problema como uno puramente multiobjetivo. Por tal motivo, la siguiente Sección presenta una introducción al enfoque multiobjetivo utilizado en adelante.

## 1.6 Enfoque Multiobjetivo

En redes de computadoras la optimización de múltiples objetivos simultáneamente, proporciona un conjunto de soluciones y la imposibilidad de decidir cuál de ellas es mejor si se consideran todos los objetivos al mismo tiempo. Se dice que las soluciones de un problema con objetivos múltiples son óptimas porque ninguna otra solución, en todo el espacio de búsqueda, es superior a ellas cuando se tienen en cuenta *todos* los objetivos al *mismo* tiempo, i.e. ningún objetivo puede mejorarse sin degradar a los demás. Esto ha sido mostrado en la Sección 1.4, donde, con ejemplos, se ha demostrado que $\alpha_T$, $C_T$, $D_A$ y $D_M$ pueden estar en conflicto entre si.

Al conjunto de estas soluciones óptimas se conoce como soluciones Pareto óptimas. Su nombre les fue dado en honor al ingeniero y economista Wilfredo Pareto, quien fue el primero en definir un nuevo criterio de optimalidad para los problemas en los que existen múltiples objetivos a optimizar, y existen conflictos al realizar la optimización simultánea

de los mismos [PAR96]. A partir de este concepto se establece, como requisito para afirmar que una situación es mejor que otra, el que en ella no se empeore ningún objetivo, pero se mejore a alguno. En caso contrario, según Pareto [PAR96], para decidir se requiere un juicio de valor aportado por el tomador de decisiones y la ciencia no puede guiarnos.

La optimización simultánea de múltiples objetivos en los problemas de ingeniería de tráfico multicast, como la utilización máxima de los enlaces, el costo del árbol multicast, el retardo máximo de extremo a extremo y el retardo medio lleva a soluciones en las que los objetivos presentan conflictos entre si; es decir, la mejora en uno conduce a un deterioro en el otro. Aunque la mayoría de los problemas de enrutamiento en redes de computadoras involucran este tipo de situaciones, las propuestas computacionales presentadas hasta la fecha se limitan a convertir el problema de objetivos múltiples en uno en que existe un solo objetivo. Esta reducción es debida a los modelos matemáticos empleados y puede realizarse de varias maneras. Por ejemplo, priorizando uno de los objetivos y utilizando los restantes como restricciones, o generando un objetivo compuesto, otorgando pesos a cada uno de ellos, de forma a optimizar la suma ponderada de los mismos. De todos modos, ninguna de estas reducciones refleja fielmente al problema y, por lo tanto, tampoco otorga soluciones completamente satisfactorias para este contexto multiobjetivo.

## 1.7 Organización del Presente Trabajo.

En el presente capítulo se ha introducido al problema de enrutamiento multicast en redes de computadoras. A continuación en el Capítulo 2 se da una definición formal de un Problema de Optimización Multiobjetivo (MOP) y se definen términos relevantes referentes a dichos problemas. Luego, en el Capítulo 3 se presenta la formulación matemática del problema de enrutamiento multicast en redes de computadoras como un MOP, y se definen las funciones objetivos. Posteriormente, en el Capítulo 4 se presentan los Algoritmos Evolutivos MultiObjetivos o MOEAs que serán considerados en el presente trabajo. Por su parte, el Capítulo 5, se realizan comparaciones estáticas de los MOEAs citados. Mientras que en el Capítulo 6 los escenario de prueba dinámicos y políticas de selección de individuos. Las conclusiones son dejadas para el Capítulo 7.

# 2 Optimización con Objetivos Múltiples

## 2.1 Introducción

El problema de enrutamiento multicast en redes de computadoras es considerado en este trabajo como un Problema de Optimización Multiobjetivo (MOP) con las siguientes funciones objetivos a optimizar: 1- utilización máxima de los enlaces del árbol ($\alpha_T$), 2- costo del árbol ($C_T$), 3- retardo máximo de extremo a extremo ($D_M$) y 4- retardo medio a los nodos destinos ($D_A$). De manera a abordar el Problema de Optimización Multiobjetivo (MOP) a continuación se presenta la definición general de un problema multiobjetivo. También se menciona una técnica tradicional de búsqueda utilizada en la actualidad para la resolución de problemas de enrutamiento multicast con más de un objetivo a optimizar, indicando brevemente sus desventajas potenciales. Posteriormente, se propone a los algoritmos evolutivos como herramientas que poseen cualidades deseables para la resolución de MOPs.

## 2.2 Problemas de Optimización Multiobjetivo

En el área de optimización multiobjetivo, debido a la naturaleza aún incipiente del ámbito de investigación, no existe una notación estándar, y no ha sido sino hasta hace muy poco tiempo atrás que los investigadores han empezado a preocuparse por definir con claridad estos aspectos. Sin embargo, en gran parte de los trabajos consultados se percibe aún bastante confusión al respecto. Por lo tanto, es esencial establecer una notación clara antes de iniciar la discusión. A continuación se da la definición formal de un problema de optimización multiobjetivo.

**Definición 1: Problema de Optimización Multiobjetivo (*Multiobjective Optimization Problem:* MOP).** Un MOP general incluye un conjunto de $h$ 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:

$$\begin{array}{ll} \textit{Optimizar} & \boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x})) \\ \textit{sujeto a} & \boldsymbol{e}(\boldsymbol{x}) = (e_1(\boldsymbol{x}), e_2(\boldsymbol{x}), \ldots, e_m(\boldsymbol{x})) \geq \boldsymbol{0} \\ \textit{donde} & \boldsymbol{x} = (x_1, x_2, \ldots, x_h) \in \boldsymbol{X} \\ & \boldsymbol{y} = (y_1, y_2, \ldots, y_k) \in \boldsymbol{Y} \end{array} \qquad \textbf{(2.1)}$$

siendo $\boldsymbol{x}$ el vector de decisión e $\boldsymbol{y}$ el vector objetivo. El espacio de decisión se denota por $\boldsymbol{X}$, y el 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 $\boldsymbol{X}_f$ y su correspondiente conjunto de vectores objetivo factibles $\boldsymbol{Y}_f$.

**Definición 2: Conjunto de soluciones factibles.** El conjunto de soluciones factibles $\boldsymbol{X}_f$ se define como el conjunto de vectores de decisión $\boldsymbol{x}$ que satisface $\boldsymbol{e}(\boldsymbol{x})$:

$$\boldsymbol{X}_f = \{\boldsymbol{x} \in \boldsymbol{X} \mid \boldsymbol{e}(\boldsymbol{x}) \geq \boldsymbol{0}\} \qquad \textbf{(2.2)}$$

La imagen de $\boldsymbol{X}_f$, es decir, la región factible del espacio objetivo, se denota por

$$\boldsymbol{Y}_f = \boldsymbol{f}(\boldsymbol{X}_f) = \mathrm{r}_{\boldsymbol{x} \in \boldsymbol{X}_f} \{\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})\} \qquad \textbf{(2.3)}$$

De estas definiciones se tiene que cada solución del MOP en cuestión consiste de una $h$-tupla $\boldsymbol{x}=(x_1, x_2, \ldots, x_h)$, que conduce a un vector objetivo $\boldsymbol{y} = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x}))$, donde cada $\boldsymbol{x}$ debe cumplir con el conjunto de restricciones $\boldsymbol{e}(\boldsymbol{x}) \geq \boldsymbol{0}$. El problema de optimización consiste en hallar la $\boldsymbol{x}$ que tenga el "mejor valor" de $\boldsymbol{f}(\boldsymbol{x})$. En general, no existe un único "mejor valor", sino un conjunto de soluciones. Entre éstas, ninguna se puede considerar mejor que las demás si se tienen en cuenta todos los objetivos simultáneamente. De este hecho se deriva que pueden existir –y generalmente existen– conflictos entre los diferentes objetivos que componen el problema. Por ende, al tratar con MOPs se precisa de un nuevo concepto de "óptimo".

En la optimización de un solo objetivo el conjunto de variables de decisión factibles está completamente ordenado mediante una función objetivo $f$. Es decir, dadas dos soluciones $\boldsymbol{u},\boldsymbol{v} \in \boldsymbol{X}_f$, se cumple una sola de las siguientes proposiciones: $f(\boldsymbol{u}) > f(\boldsymbol{v})$, $f(\boldsymbol{u}) = f(\boldsymbol{v})$

o $f(\boldsymbol{v}) \succ f(\boldsymbol{u})$. El objetivo consiste en hallar la solución (o soluciones) que tengan los valores óptimos (máximos o mínimos) de $f$. Cuando se trata de varios objetivos, sin embargo, la situación cambia. $\boldsymbol{X}_f$, en general, no está totalmente ordenada por los objetivos. El orden que se da generalmente es parcial (i.e. pueden existir dos vectores de decisión diferentes $\boldsymbol{u}$ y $\boldsymbol{v}$ tal que $\boldsymbol{f}(\boldsymbol{u})$ no puede considerarse mejor que $\boldsymbol{f}(\boldsymbol{v})$ ni $\boldsymbol{f}(\boldsymbol{v})$ puede considerarse mejor que $\boldsymbol{f}(\boldsymbol{u})$). Para expresar esta situación matemáticamente, las relaciones =, ≤ y ≥ se deben extender. Esto se puede realizar de la siguiente manera:

**Definición 3: Extensión de las relaciones =, ≤ y ≥ a** MOPs. Dados 2 vectores de decisión $\boldsymbol{u},\boldsymbol{v} \in \boldsymbol{X}$,

$$\begin{aligned}
&\boldsymbol{f}(\boldsymbol{u}) = \boldsymbol{f}(\boldsymbol{v}) && \text{si y solo si} && \forall i \in \{1, 2, \ldots, k\}: f_i(\boldsymbol{u}) = f_i(\boldsymbol{v}) \\
&\boldsymbol{f}(\boldsymbol{u}) \leq \boldsymbol{f}(\boldsymbol{v}) && \text{si y solo si} && \forall i \in \{1, 2, \ldots, k\}: f_i(\boldsymbol{u}) \leq f_i(\boldsymbol{v}) \\
&\boldsymbol{f}(\boldsymbol{u}) < \boldsymbol{f}(\boldsymbol{v}) && \text{si y solo si} && \boldsymbol{f}(\boldsymbol{u}) \leq \boldsymbol{f}(\boldsymbol{v}) \wedge \boldsymbol{f}(\boldsymbol{u}) \neq \boldsymbol{f}(\boldsymbol{v})
\end{aligned} \qquad \textbf{(2.4)}$$

Las relaciones ≥ y > se definen de manera similar. Es claro que dos vectores $\boldsymbol{u},\boldsymbol{v} \in \boldsymbol{X}$ solo pueden cumplir con una de las tres expresiones dadas por las Ecuaciones 2.4. Esta situación se expresa con los siguientes símbolos y términos:

**Definición 4: Dominancia Pareto en un contexto de minimización.** Para dos vectores de decisión $\boldsymbol{u}$ y $\boldsymbol{v}$,

$$\begin{aligned}
&\boldsymbol{u} \succ \boldsymbol{v} \ (\boldsymbol{u} \text{ domina a } \boldsymbol{v}) && \text{si y solo si} && \boldsymbol{f}(\boldsymbol{u}) < \boldsymbol{f}(\boldsymbol{v}) \\
&\boldsymbol{v} \succ \boldsymbol{u} \ (\boldsymbol{v} \text{ domina a } \boldsymbol{u}) && \text{si y solo si} && \boldsymbol{f}(\boldsymbol{v}) < \boldsymbol{f}(\boldsymbol{u}) \\
&\boldsymbol{u} \sim \boldsymbol{v} \ (\boldsymbol{u} \text{ y } \boldsymbol{v} \text{ no son comparables}) && \text{si y solo si} && \boldsymbol{f}(\boldsymbol{u}) \nleq \boldsymbol{f}(\boldsymbol{v}) \wedge \boldsymbol{f}(\boldsymbol{v}) \nleq \boldsymbol{f}(\boldsymbol{u})
\end{aligned} \qquad \textbf{(2.5)}$$

**Definición 5: Dominancia Pareto en un contexto de maximización.** Para dos vectores de decisión $\boldsymbol{u}$ y $\boldsymbol{v}$,

| | | | |
|---|---|---|---|
| $\boldsymbol{u} \succ \boldsymbol{v}$ ($\boldsymbol{u}$ domina a $\boldsymbol{v}$) | si y solo si | $\boldsymbol{f}(\boldsymbol{u}) > \boldsymbol{f}(\boldsymbol{v})$ | |
| $\boldsymbol{v} \succ \boldsymbol{u}$ ($\boldsymbol{v}$ domina a $\boldsymbol{u}$) | si y solo si | $\boldsymbol{f}(\boldsymbol{v}) > \boldsymbol{f}(\boldsymbol{u})$ | **(2.6)** |
| $\boldsymbol{u} \sim \boldsymbol{v}$ ($\boldsymbol{u}$ y $\boldsymbol{v}$ no son comparables) | si y solo si | $\boldsymbol{f}(\boldsymbol{u}) \not\geq \boldsymbol{f}(\boldsymbol{v}) \wedge \boldsymbol{f}(\boldsymbol{v}) \not\geq \boldsymbol{f}(\boldsymbol{u})$ | |

Alternativamente, $\boldsymbol{u} \rhd \boldsymbol{v}$ denota que $\boldsymbol{u}$ domina o es no comparable con $\boldsymbol{v}$.

Luego, de aquí en adelante, ya no será necesario diferenciar el tipo de optimización a realizar (minimización o maximización), al punto que un objetivo puede ser maximizado, mientras que otro puede ser minimizado.

Definido el concepto de dominancia Pareto, puede ser introducido el criterio de optimalidad Pareto de la siguiente manera:

**Definición 6: Optimalidad Pareto.** Dado un vector de decisión $\boldsymbol{x} \in \boldsymbol{X}_f$, se dice que $\boldsymbol{x}$ es no dominado respecto a un conjunto $\boldsymbol{V} \subseteq \boldsymbol{X}_f$ si y solo si

$$\forall\ \boldsymbol{v} \in \boldsymbol{V} : (\boldsymbol{x} \succ \boldsymbol{v} \vee \boldsymbol{x} \sim \boldsymbol{v}) \qquad \textbf{(2.7)}$$

En caso que $\boldsymbol{x}$ sea no dominado respecto a todo el conjunto $\boldsymbol{X}_f$, y solo en ese caso, se dice que $\boldsymbol{x}$ es una **solución Pareto óptima**. Por lo tanto, el conjunto Pareto óptimo $\boldsymbol{X}_{true}$ puede ser definido formalmente a continuación.

**Definición 7: Conjunto Pareto óptimo.** Dado el conjunto de vectores de decisión factibles $\boldsymbol{X}_f$, se denomina conjunto Pareto óptimo $\boldsymbol{X}_{true}$ al conjunto de vectores de decisión no dominados que pertenecen a $\boldsymbol{X}_f$, es decir:

$$\boldsymbol{X}_{true} = \{\ \boldsymbol{x} \in \boldsymbol{X}_f \mid \boldsymbol{x} \text{ es no dominado con respecto a } \boldsymbol{X}_f\ \} \qquad \textbf{(2.8)}$$

El correspondiente conjunto de vectores objetivos $\boldsymbol{Y}_{true} = \boldsymbol{f}(\boldsymbol{X}_{true})$ constituye el **frente Pareto óptimo.**

## 2.3 Búsqueda y Toma de Decisiones

Se pueden distinguir 2 niveles de dificultad al resolver un problema de optimización multiobjetivo: la búsqueda y la toma de decisiones [HOR97]. El primer aspecto se refiere al proceso de optimización, durante el cual se explora el espacio de soluciones factibles buscando las soluciones Pareto óptimas. En esta fase, como ocurre en la optimización de objetivo único, el espacio de búsqueda puede ser lo suficientemente grande y complejo como para impedir el uso de técnicas de optimización que den resultados exactos [STE86]. El segundo aspecto es equivalente a seleccionar un conjunto o una única solución de compromiso del conjunto Pareto óptimo ya encontrado. Este paso generalmente tiene que ver con cuestiones inherentes a las condiciones del problema y los recursos disponibles.

Los métodos de optimización multiobjetivo pueden clasificarse en tres categorías diferentes dependiendo de la manera en que se combinan ambas fases [COH78, HWA79]:

a) Toma de decisión previa a la búsqueda (decidir, luego buscar): los objetivos del MOP se combinan en un único objetivo que implícitamente incluye información de preferencia obtenida del responsable de la toma de decisiones. Estos son métodos de toma de decisión *a priori*. Gran parte de las técnicas tradicionales para resolución de MOPs utilizan esta estrategia [COH78, STE86].
b) Búsqueda previa a la toma de decisión (buscar, luego decidir): se realiza la optimización sin incluir información de preferencia. El responsable de la toma de decisiones escoge del conjunto de soluciones obtenidas (que idealmente son todas del conjunto Pareto óptimo). A éstos se conoce como métodos de toma de decisión *a posteriori*.
c) Toma de decisión durante la búsqueda (decidir mientras se busca): esto permite que quien toma las decisiones pueda establecer preferencias durante un proceso de optimización interactivo. Luego de llevarse a cabo cada paso del proceso de optimización, se presenta al responsable de decisiones los conflictos existentes y se le permite especificar sus preferencias o compromisos. De esta manera se desarrolla una búsqueda guiada. Estos métodos reciben el nombre de *progresivos* o *interactivos*.

Combinar múltiples objetivos en un único objetivo a optimizar, posee la ventaja de que a la función resultante se puede aplicar cualquiera de los métodos de optimización de un único objetivo, ya disponibles en la literatura [COH78, STE86], sin mayores modificaciones. No obstante, tal combinación requiere conocimiento del dominio del problema que no siempre está disponible. En los problemas de diseño de diversas áreas de ingeniería, se realiza la búsqueda justamente para tener un mejor conocimiento del espacio de búsqueda del problema y las soluciones alternativas. Al realizar la búsqueda antes de la toma de decisión no se requiere este conocimiento, pero se impide que quien toma la decisión exprese previamente sus preferencias, lo que probablemente conduce a una reducción de la complejidad del espacio de búsqueda. Un problema con las técnicas de toma de decisión interactivas o *a posteriori* es la dificultad de presentar al responsable de la toma de decisiones las diferencias entre las soluciones de un MOP de muchas dimensiones. Aún así, la integración de la búsqueda y la toma de decisiones que proponen las técnicas interactivas parece una alternativa promisoria para aprovechar las ventajas de los dos primeros tipos de técnicas.

El presente trabajo esta enfocado en técnicas que han demostrado ser capaces de explorar espacios de búsquedas prácticamente ilimitados y altamente complejos y generar aproximaciones al conjunto Pareto óptimo.

## 2.4 El Método de Suma con Pesos

En redes de computadoras, el método tradicional de enrutamiento ha sido basado en esquemas que optimizan una sola métrica. Recientemente se han propuestos nuevos esquemas y algoritmos que consideran más de una métrica a optimizar [SEO02, DON04]. El método utilizado es el de la suma ponderada utilizando pesos [COH78]. Este método combina las diferentes funciones objetivo en una sola función *F*, de la siguiente manera:

Optimizar $$F = \sum_{j=1}^{k} w_j f_j(\boldsymbol{x}) \qquad \textbf{(2.9)}$$

donde $\boldsymbol{x} \in \boldsymbol{X_f}$, y $w_j$ es el peso usado para ponderar la *j*-ésima función objetivo.

Usualmente, y sin pérdida de generalidad, se escogen pesos fraccionales y diferentes de cero, de manera a cumplir la condición de normalización: $\sum_{j=1}^{k} w_j = 1,\ w_j > 0$ [DON04].

El procedimiento es el siguiente: se escoge una combinación de pesos y se optimiza la función *F* para obtener una solución óptima. Otras soluciones surgen a partir de optimizaciones realizadas sobre una combinación diferente de pesos [DON04].

En el presente contexto, optimizar implica maximizar todas las funciones objetivo, o minimizarlas. Esto es, o se maximiza o se minimiza todas las funciones $f_j(\boldsymbol{x})$; no se admite la maximización de algunos objetivos y la minimización de otros.

En [DEB99] se demuestra que, si se utiliza un algoritmo de optimización que obtiene resultados exactos siempre y los pesos escogidos son siempre positivos, el método genera soluciones que siempre pertenecen al conjunto Pareto óptimo. La interpretación de este método es la siguiente. Alterar el vector de pesos y optimizar la ecuación implica encontrar un hiperplano (una línea para el caso en que se tengan dos objetivos) con una orientación fija en el espacio de la función. La solución óptima es el punto donde un hiperplano con esta orientación tiene una tangente común con el espacio de búsqueda factible. De aquí deducimos que este método no puede usarse para encontrar soluciones Pareto óptimas en problemas de optimización multicriterio que tienen un frente Pareto óptimo cóncavo. Para una discusión más detallada de esta característica se refiere al lector a [DEB99].

Si bien no se adecuan a la naturaleza del problema, la ventaja principal del método de suma con pesos es que permite la utilización de algoritmos desarrollados para la resolución de problemas de optimización mono-objetivos (*Single Objective Problems - SOPs*) bien conocidos y de probada eficacia, aún cuando se trate de problemas reales, de gran tamaño. Para problemas de gran escala, muy pocas técnicas de optimización multiobjetivo reales se han presentado [HOR97], a diferencia de técnicas de optimización de un solo objetivo que han existido desde hace bastante tiempo y han sido probadas en diferentes situaciones. Sin embargo, este método cuenta con las siguientes limitaciones [DEB99]:

a) El algoritmo de optimización se debe aplicar varias veces para encontrar las múltiples soluciones Pareto óptimas. Como cada corrida es independiente de las demás, generalmente no se obtiene un efecto sinergético. Por tanto, delinear el frente Pareto óptimo resulta computacionalmente muy caro.
b) Requiere conocimiento previo del problema a resolver y es sensible a los pesos utilizados.
c) Algunos algoritmos son sensibles a la forma del frente Pareto (problemas con curvas cóncavas).
d) La variación entre las diferentes soluciones encontradas depende de la eficiencia del optimizador de un solo objetivo. Podría darse el caso de encontrar siempre la misma solución o soluciones muy parecidas, en corridas múltiples.
e) Como los optimizadores de un solo objetivo no son eficientes en búsquedas de universos discretos [DEB99], tampoco serán eficientes para optimizaciones multiobjetivo en espacios discretos.

En [DEB99] se ha demostrado que todas las dificultades arriba mencionadas pueden ser superadas con la utilización de algoritmos evolutivos. Las características de estos algoritmos hacen posible que:

a) se manejen espacios de búsqueda de casi cualquier tamaño y características;
b) debido a su paralelismo inherente, se puedan generar múltiples soluciones en una sola corrida, permitiendo un efecto sinergético.

A continuación, se introduce el criterio que se utilizará en el presente trabajo para lidiar con el problema de enrutamiento multicast multiobjetivo en redes de computadoras. Es decir, se empleará algoritmos evolutivos. La siguiente Sección presentará una introducción a los algoritmos evolutivos y más adelante en el capítulo 4, se estudiarán los algoritmos utilizados en el presente trabajo.

## 2.5 El método del Orden Lexicográfico.

El método de Orden Lexicográfico se basa en la asignación de prioridades a priori a cada uno de los Objetivos a optimizar. El objetivo que posee la mayor prioridad es utilizado para realizar el orden, si ocurre un empate el siguiente objetivo de mayor

prioridad es comparado, y así sucesivamente. Todos los objetivos $f_1,\ldots,f_k$ son ordenados en orden creciente de prioridad.

## 2.6 Algoritmos Evolutivos en Optimización Multiobjetivo

El término algoritmo evolutivo (*Evolutionary Algorithm* - EA) se refiere a técnicas de búsqueda y optimización inspiradas en el modelo de la evolución propuesto por Charles Darwin [DAR85], luego de sus viajes exploratorios.

La naturaleza de los individuos se caracteriza mediante cadenas de material genético que se denominan cromosomas. En los cromosomas se halla codificada toda la información relativa a un individuo y a sus tendencias. El cromosoma es una cadena de símbolos llamados genes. Cada individuo posee un nivel de adaptación al medio que lo dota de mayor capacidad de sobrevivencia y generación de descendencia. Tal nivel de adaptabilidad está ligado a las características que están codificadas en sus cromosomas. Como el material genético puede transmitirse de padres a hijos al ocurrir el apareamiento, los hijos resultantes poseen cadenas de cromosomas parecidas a las de sus padres y combinan las características de los mismos. Por lo tanto, si padres con buenas características se cruzan, posiblemente generarán hijos igualmente buenos o incluso mejores [GOL89].

En la codificación de un individuo o cromosoma, debe estar presente toda la información relevante del mismo y que se considera influye en la optimización o búsqueda. Como se citó anteriormente, el cromosoma es una cadena de símbolos conocidos como genes. El cromosoma o individuo representa una solución al problema de optimización.

Para cada individuo es determinado su nivel de aptitud o adaptabilidad (*fitness*), dependiendo de la calidad de la solución que representa. Posteriormente, los individuos existentes generan a otros individuos mediante operadores genéticos como selección, cruzamiento y mutación. El operador de selección elige los padres que se cruzarán. La probabilidad de que un individuo sea escogido como padre y/o que sobreviva hasta la siguiente generación está ligada a su adaptabilidad: a mayor adaptabilidad, mayor probabilidad de sobrevivencia y de tener descendientes, de la misma forma que ocurre en

los procesos naturales. Luego de escogerse los padres, se procede a la recombinación o cruzamiento de los mismos para obtener a la nueva generación. De esta manera, en cada nueva generación se tienen buenas probabilidades de que la población se componga de mejores individuos, ya que los hijos heredarán las buenas características de sus padres, y al combinarlas podrán ser aún mejores. Por otro lado, durante la recombinación pueden ocurrir alteraciones (mutaciones) en la información genética de un individuo. Si tales alteraciones se producen para bien, originarán un individuo bueno con alta adaptabilidad y la alteración se transmitirá a los nuevos individuos; si el cambio no es benéfico, el individuo alterado tendrá una adaptabilidad baja y poca o ninguna descendencia, con lo que la alteración prácticamente morirá con él. De esta manera, luego del curso de varias generaciones, la población habrá evolucionado hacia individuos con nivel de adaptabilidad elevado, es decir, representarán buenas soluciones al problema propuesto [GOL89].

La reproducción enfoca la atención en los individuos con alta adaptabilidad, y de esta manera **explota** la información disponible sobre la adaptación del individuo al medio ambiente. La recombinación y la mutación perturban de alguna manera a los individuos y proveen así de heurísticas para la **exploración** del espacio de búsqueda. Por ello se dice que los EAs utilizan conceptos de explotación y exploración. A pesar de ser simplistas desde el punto de vista de la biología, estos algoritmos son suficientemente complejos como para proveer mecanismos de búsqueda robustos y que se adaptan a gran variedad de problemas [GOL89].

### 2.6.1 Algoritmos Genéticos

En la práctica se implementa el algoritmo genético escogiendo una codificación para las posibles soluciones del problema. La codificación se realiza mediante cadenas de bits, números o caracteres para representar a los individuos. Luego, las operaciones de cruzamiento y mutación se aplican de manera muy sencilla mediante funciones de manipulación de vectores. A pesar de que se han realizado numerosas investigaciones [HEl00] sobre codificación usando cadenas de longitud variable y aún otras estructuras, gran parte del trabajo con algoritmos genéticos se enfoca en cadenas de bits de longitud fija. Justamente, el hecho de utilizar cadenas de longitud fija y la necesidad que existe de codificar las posibles soluciones, son las dos características cruciales que diferencian a

los algoritmos genéticos de la programación genética, que no posee representación de longitud fija y que normalmente no utiliza una codificación del problema y sus soluciones.

El ciclo de trabajo de un GA (también conocido como ciclo generacional) es generalmente el siguiente: calcular la adaptabilidad de los individuos en la población, crear una nueva población mediante selección, cruzamiento y mutación, y finalmente descartar la población vieja y seguir iterando utilizando la población recién creada. A cada iteración de este ciclo se la conoce como una generación. Cabe observar que no hay una razón teórica para que este sea el modelo de implementación. De hecho este comportamiento puntual no se observa como tal en la naturaleza. Sin embargo el modelo sigue siendo válido y conveniente [GOL89].

La primera generación (generación 0) de este proceso, opera sobre una población de individuos generados al azar. A partir de allí, los operadores genéticos se aplican para mejorar a la población. La Figura 2.1 presenta el pseudocódigo del algoritmo genético simple.

```
Inicio
   t = 0                                  // generación 0
   InicializarPoblacion P(t)              // inicializar la población de individuos al azar
   Evaluar P (t)                          // Hallar el fitness de todos los individuos
   Hacer{
        P(t+1) = SeleccionarPadres P(t)   // seleccionar población para generar descendientes
        Cruzar P(t+1)                     // recombinar los "genes" de padres seleccionados
        Mutar P(t+1)                      // perturbar la población generada estocásticamente
        Adaptabilidad P(t+1)              // calcular adaptabilidad de la población recién creada
        t = t + 1                         // incrementar el contador de generaciones
   }Mientras no se cumpla el criterio de parada
Fin
```

**Figura 2.1.** Pseudocódigo del algoritmo genético simple

El proceso de selección natural donde el más fuerte tiene mayor capacidad de supervivencia se realiza en *SeleccionarPadres* de la figura 2.1. En el GA la capacidad de supervivencia de un individuo está ligada a su valor de *adaptabilidad*. Este operador se aplica a cada iteración sobre una población de individuos de tamaño constante, con el objetivo de seleccionar individuos prometedores para generar la nueva población. Entre los individuos seleccionados pueden hallarse dos o más individuos idénticos. Esto se debe a que los individuos con baja adaptabilidad tienen poca probabilidad de ser elegidos,

mientras que los de buena adaptabilidad son seleccionados con mayor frecuencia. El operador de selección puede aplicarse de diversas maneras. A continuación se describen dos implementaciones muy conocidas: selección por torneo y selección por ruleta. En la selección por torneos [GOL89] se escoge al azar un grupo de individuos y gana el torneo aquel con mejor adaptabilidad. La cantidad de individuos que se escogen para la competencia se fija de antemano y permanece constante en la implementación tradicional. Esta forma de selección refleja de manera más adecuada el proceso natural de selección. Por otra parte, la implementación de selección por ruleta es fácilmente comprensible imaginando una ruleta en la que el número de partes en que se divide la misma es igual a la cantidad de individuos de la población, siendo el tamaño de cada parte proporcional a la adaptabilidad de cada individuo. Es de esperar que al hacer girar la ruleta varias veces, se obtendrá mayor cantidad de individuos con alta adaptabilidad. Aunque pueden existir otras formas de realizar la selección, las dos anteriormente mencionadas son las más comunes en las implementaciones publicadas.

A cada par de individuos seleccionados se aplica el operador de cruzamiento. La operación de cruzamiento también puede ser realizada de maneras diferentes y aquí solo se discutirá una de ellas: el cruzamiento de dos puntos. En el mismo, se escogen aleatoriamente dos puntos de cruce. El primer punto de cruce define el punto (gen) a partir del cual se hará el intercambio genético, y el segundo define la longitud del segmento (número de genes) a intercambiar. La Figura 2.2 ilustra esta operación, en la cual los cromosomas consisten de cadenas de enteros de longitud 6.

**Figura 2.2.** Operación de cruzamiento de dos puntos. El primer punto de cruce es 2 mientras que la longitud del segmento a intercambiar es 3

El operado de cruzamiento representa una forma de búsqueda local, en las inmediaciones del espacio de búsqueda que rodea a los padres. Por su parte, el proceso de mutación es básicamente una búsqueda aleatoria. Se selecciona aleatoriamente una posición específica dentro de cromosoma del individuo a mutar, para luego cambiar el valor contenido en dicha posición. Dado que en la naturaleza la probabilidad de que ocurra una mutación es pequeña, los GAs tratan de representar lo mismo asignando un valor de ocurrencia muy bajo al operador de mutación [GOL89].

### 2.6.2 Algoritmos Evolutivos y Optimización Multiobjetivo

Los EAs resultan interesantes pues están especialmente dotados para lidiar con las dificultades propuestas por los MOPs. Esto se debe a que pueden devolver un conjunto entero de soluciones luego de una corrida simple y no presentan otras limitaciones propias de las técnicas tradicionales. Incluso, algunos investigadores han sugerido que los EAs se comportarían mejor que otras técnicas de búsqueda ciega [FON95a, FON95b]. Esta afirmación todavía requiere de una demostración fehaciente y debería considerarse a la luz de los teoremas de "no hay almuerzo gratis" ("*no free lunch*") relacionados con la optimización, y que indican que si un método se comporta "mejor" que otros en un conjunto de problemas, tendrá un desempeño "peor" en otro conjunto de problemas [WOL97]. De todos modos, la realidad es que hoy día existen pocas alternativas válidas en cuanto a otros posibles métodos de solución [HOR97]. De hecho, las numerosas publicaciones recientes sobre resolución de MOPs usando EAs (que pueden hallarse compiladas en [COE00]), parecen considerar este hecho, y han dado lugar al nacimiento de todo un nuevo campo de investigación: los MOEAs, algoritmos evolutivos aplicados a optimización multiobjetivo.

Ya en 1967, Rosemberg sugirió, sin hacer simulaciones, un método de búsqueda genética para aplicaciones químicas con diversos objetivos [ROS67]. Sin embargo, las primeras implementaciones prácticas fueron sugeridas recién en 1984 [SHA84]. Posterior a esto no se realizaron estudios significativos por casi una década, a excepción de un procedimiento de ordenación basado en no-dominancia, expuesto por Goldberg [GOL89]. Nuevas implementaciones de MOEAs surgieron entre 1991 y 1994 [KUR91, HAJ92, FON93, HOR94, SRI94]. Posteriormente, estas sugerencias –y variaciones de las mismas– se implementaron para la resolución de diferentes MOPs [PAR98]. Recientemente, algunos investigadores se han puesto la tarea de tratar tópicos específicos de la búsqueda y optimización multiobjetivo con algoritmos evolutivos, como son: convergencia al frente Pareto [VAN98a, VAN98b, RUD98] y aplicación de elitismo [PAR98, OBA98]. También se han publicado resúmenes analíticos generales, comparaciones y compendios [FON95b, HOR97, VAN98a, DEB99, COE99].

## 2.7 Resumen del Capítulo

En el presente Capítulo se ha dado la definición formal de un problema de optimización multiobjetivo y se han definido términos y conceptos que serán utilizados en los posteriores Capítulos. Luego, se ha propuesto a los MOEAs como herramientas que poseen cualidades deseables para la resolución de MOPs. Habiendo introducido estos conceptos, en el Capítulo 3 se formula el problema de enrutamiento multicast como un MOP. Posteriormente, en el Capítulo 4 se proponen los MOEAs que se utilizarán a lo largo del presente trabajo.

# 3 Formulación Matemática

## 3.1 Introducción

En este capítulo se considera el Problema de Enrutamiento Multicast como uno MOP, en el cual debe ser construido un árbol multicast para un grupo dado minimizando 1- la utilización máxima de los enlaces del árbol, 2- el costo del árbol, 3- el retardo máximo de extremo a extremo, y 4- el retardo medio.

El presente Capítulo está organizado de la siguiente manera: en la Sección 3.2 se da el modelo matemático del problema de enrutamiento multicast en redes de computadoras. Posteriormente, de forma a aclarar el significado de las notaciones matemáticas, la Sección 3.3 presenta un ejemplo de un problema de enrutamiento multicast multiobjetivo.

## 3.2 Modelo Matemático

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 $j$; $i, j \in V$.
- $z_{ij} \in \Re^+$ : capacidad del enlace $(i,j)$.
- $c_{ij} \in \Re^+$ : costo del enlace $(i,j)$.
- $d_{ij} \in \Re^+$ : retardo (delay) del enlace $(i,j)$.
- $t_{ij} \in \Re^+$ : tráfico actual fluyendo a través del enlace $(i,j)$.
- $s \in V$: nodo origen del grupo multicast.
- $N \subseteq V\text{-}\{s\}$: conjunto de nodos destinos del grupo multicast.
- $\phi \in \Re^+$ : demanda de tráfico del grupo multicast, en bps.
- $T(s,N)$: árbol multicast con origen en $s$ y conjunto de nodos destinos $N$.
- $p_T(s, n) \subseteq T(s,N)$: camino (*path*) que conecta el nodo fuente $s$ y el nodo destino $n \in N$.

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

$$d\left(p_T(s,n)\right) = \sum_{(i,j)\in p_T(s,n)} d_{ij} \qquad \textbf{(3.1)}$$

Usando la notación arriba definida, el problema de enrutamiento multicast puede ser definido como un MOP que trata de hallar el árbol multicast $T(s,N)$, que minimiza las siguientes funciones objetivos:

1- Utilización máxima de los enlaces del árbol:

$$\alpha_T = \underset{(i,j)\in T}{Max} \left\{ \frac{\phi + t_{ij}}{z_{ij}} \right\} \qquad \textbf{(3.2)}$$

2- Costo del árbol:

$$C_T = \phi \sum_{(i,j)\in T} c_{ij} \qquad \textbf{(3.3)}$$

3- Retardo máximo de extremo a extremo:

$$D_M = \underset{n\in N}{Max} \left\{ d\left(p_T(s,n)\right) \right\} \qquad \textbf{(3.4)}$$

4- Retardo medio:

$$D_A = \frac{1}{|N|} \sum_{n\in N} d\left(p_T(s,n)\right) \qquad \textbf{(3.5)}$$

donde |N| representa la cardinalidad del conjunto N, sujeto a restricciones de capacidad en los enlaces:

$$\phi + t_{ij} \le z_{ij} \;;\; \forall (i,j) \in T(s,N) \qquad \textbf{(3.6)}$$

esto es: $\alpha_T \le 1$.

La Ecuación 3.2 da la utilización máxima de los enlaces del árbol. La minimización de esta métrica implica que son preferidos aquellos árboles con la menor utilización máxima de sus enlaces, de forma a balancear la carga sobre la red [SEO02, DON04].

La Ecuación 3.3 da el costo total del árbol multicast. Formulado de esta manera, esta métrica puede tener más de un significado. Si el costo representa alguna cantidad monetaria para el proveedor de servicio, entonces sería deseable proveer un servicio multicast a un costo mínimo. Por ejemplo, el costo de cada enlace puede tener un significado monetario a pagar por bit de información por unidad de tiempo. En este caso, la función objetivo dada por la Ecuación 3.3 da el costo total a pagar por bit de información por unidad de tiempo para llevar a cabo la transmisión multicast. Por otra parte, si el costo de cada enlace tiene un valor unitario, el costo total es el consumo total de ancho de banda del grupo. En cualquiera de los casos, la minimización de esta métrica es siempre deseable. Es claro que formular el problema con enlaces cuyos costos pueden tener valores arbitrarios positivos provee generalidad al problema.

La Ecuación 3.4 define el retardo máximo de extremo a extremo. Debido al creciente número de aplicaciones multicast sensibles a esta métrica, es de suma importancia que los árboles multicast sean construidos minimizando esta función objetivo. A diferencia de los trabajos publicados hasta la fecha [KOM93, RAV98, XIA99, ZHE01, ARA02], en los cuales el retardo máximo es considerado como una restricción, en el presente trabajo se formula explícitamente como una función objetivo a minimizar. Las ventajas de este enfoque serán resaltadas en el Ejemplo 3.1.

La Ecuación 3.5 define el retardo medio a los nodos destinos. Es importante notar que, si bien las Ecuaciones 3.3 y 3.4 definen funciones objetivos relacionadas, tienen distintos significados.

Por último, la Ecuación 3.6 restringe el tráfico máximo en cada enlace a la capacidad del canal.

## 3.3 Problema Ejemplo

Se propone el siguiente ejemplo de manera a comprender mejor la formulación propuesta. En el mismo se ha utilizando tres de las cuatro métricas mencionadas anteriormente. Considere la red de computadoras mostrada en la Figura 3.1 [BAR00], en donde los números sobre cada enlace $(i, j)$ corresponden a valores de retardo $d_{ij}$ y costo $c_{ij}$. El retardo está dado en milisegundos. Se genera una solicitud multicast en la que el nodo $s$=8 desea transmitir información a los nodos $N$ = {0, 3, 11, 12}.

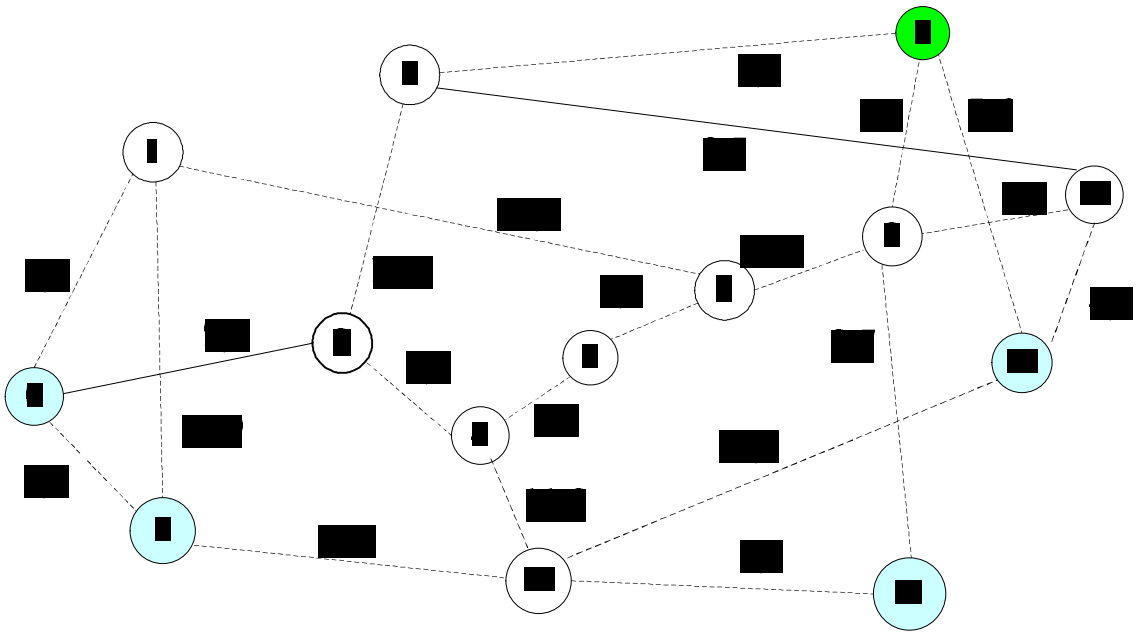


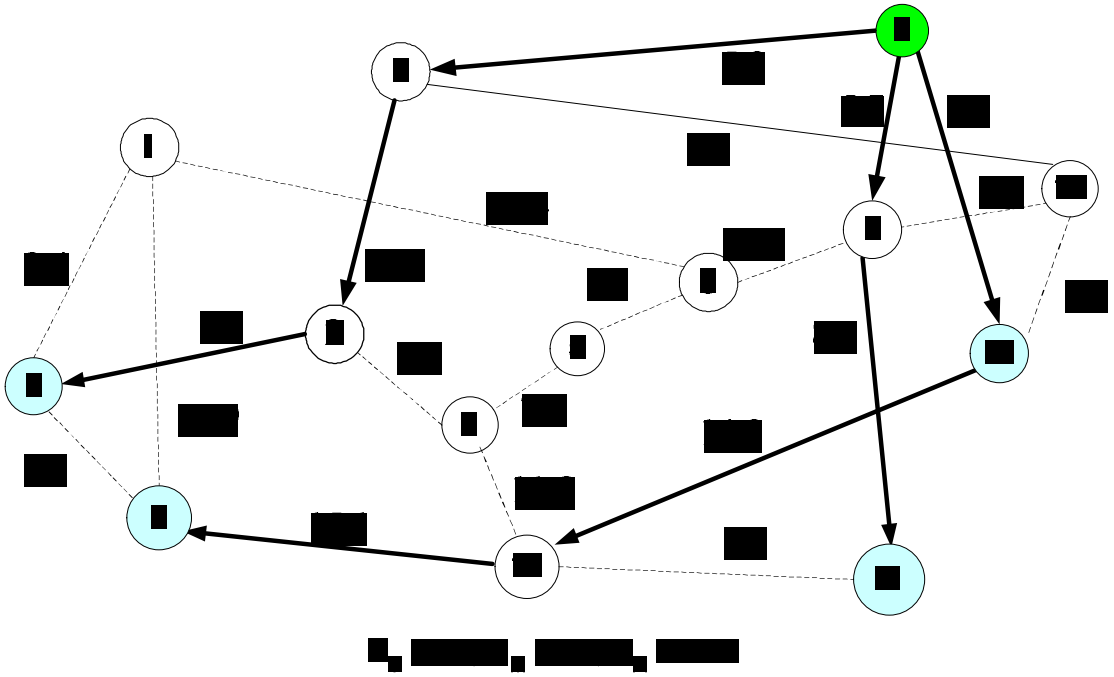
**Figura 3.1.** Red de la NSF. Cada enlace $(i,j)$ tiene asignado sus correspondientes valores de retardo $d_{ij}$ y costo $c_{ij}$. Se genera una solicitud multicast dada por $s$ = 8 y $N$ = {0, 3, 11, 12}.

Una forma tradicional de enrutar la demanda solicitada es a través del árbol del camino más corto, sobre todo para el caso que la aplicación sea sensible al retardo máximo. Una alternativa es entonces la construcción de un árbol multicast que esté compuesto por los caminos más cortos (SPT), en términos de retardo, a los nodos destinos. El SPT podría encontrar una de las soluciones mostradas en las figuras 3.2 (a) y (b), que poseen el menor retardo máximo. Nótese que la solución (b) posee un valor

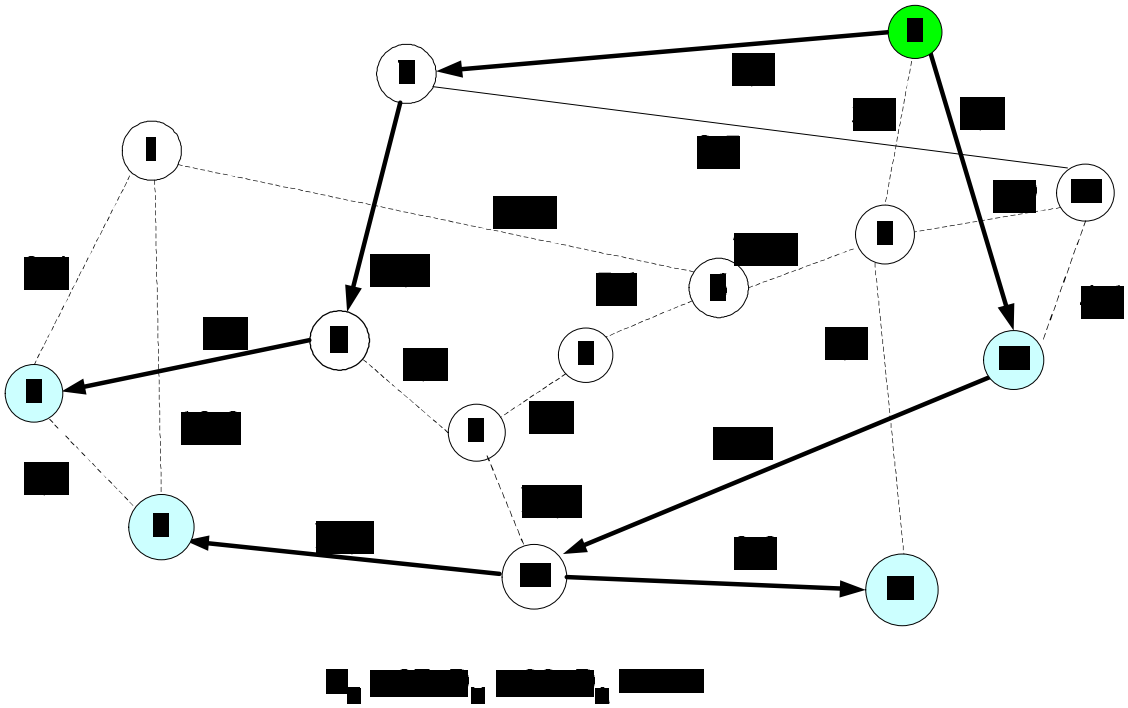
mucho menor de costo que la solución (a), el SPT podría encontrar cualquiera de las soluciones, no nos asegura que encontrará las solución con menor costo.

Debido a que el SPT solo considera una métrica, el árbol multicast puede tener un costo muy elevado, o como en el caso anterior cuando tenemos dos o mas soluciones con el mismo retardo máximo, el SPT no garantiza encontrar la solución con menor costo o retardo medio. Por lo tanto, se podrían hallar otras soluciones que, sacrificando el retardo máximo tengan un menor costo o que garanticen el menor costo para un determinado límite de retardo máximo. Para esto podría usar un algoritmo basado en restricciones como KPP [KOM93a, Kom93b], de forma a optimizar el costo del árbol. KPP y otros algoritmos basados en restricciones [RAV98, XIA99, ZHE01, ARA02] requieren un valor de retardo máximo definido a priori. Entonces, suponga que el grupo multicast desea un árbol con un valor $D_M$ menor a 38 ms. En dicho caso, KPP hallaría la solución mostrada en la Figura 3.2 (b). Note que el KPP en este caso garantiza encontrar la solución con el menor costo.

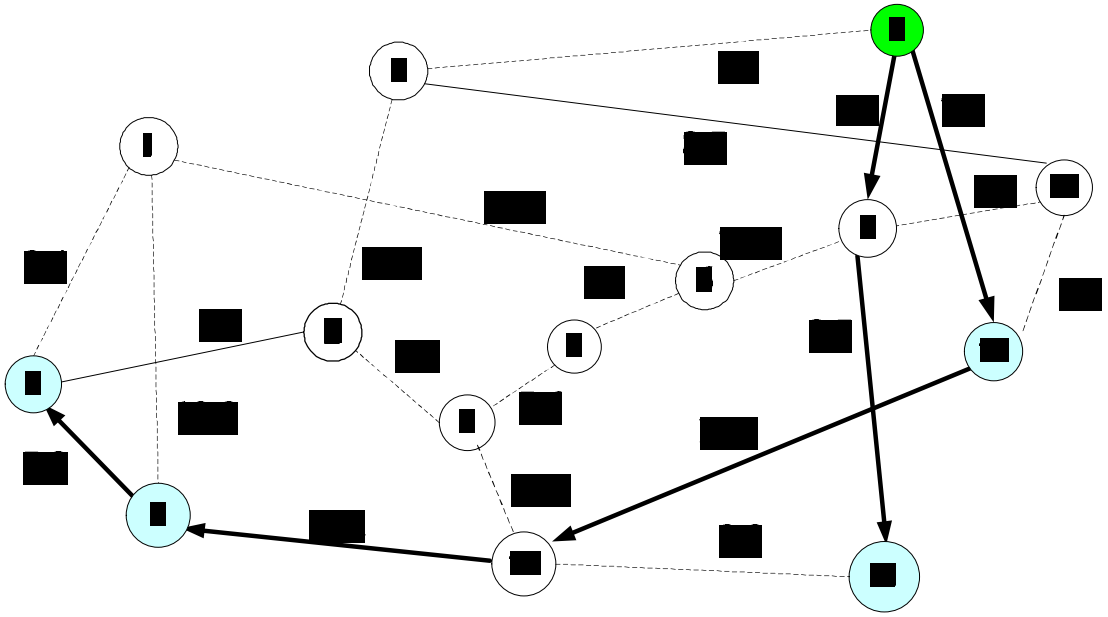
El problema de los algoritmos basados en restricciones es la necesidad del valor de retardo máximo dado a priori, pues se pueden descartar soluciones de muy bajo costo con valores $D_M$ apenas superiores al máximo permitido. Si comparamos las soluciones mostradas en las figuras 3.2 (a) y (e), se puede notar que sacrificando un poco mas de retardo máximo en la solución (a), podemos conseguir una solución con un menor costo y un valor similar de retardo medio. Esta podría ser una solución muy interesante que el KPP no encontraría en el caso de considerar la restricción de retardo máximo como inferior a 36.



(a)

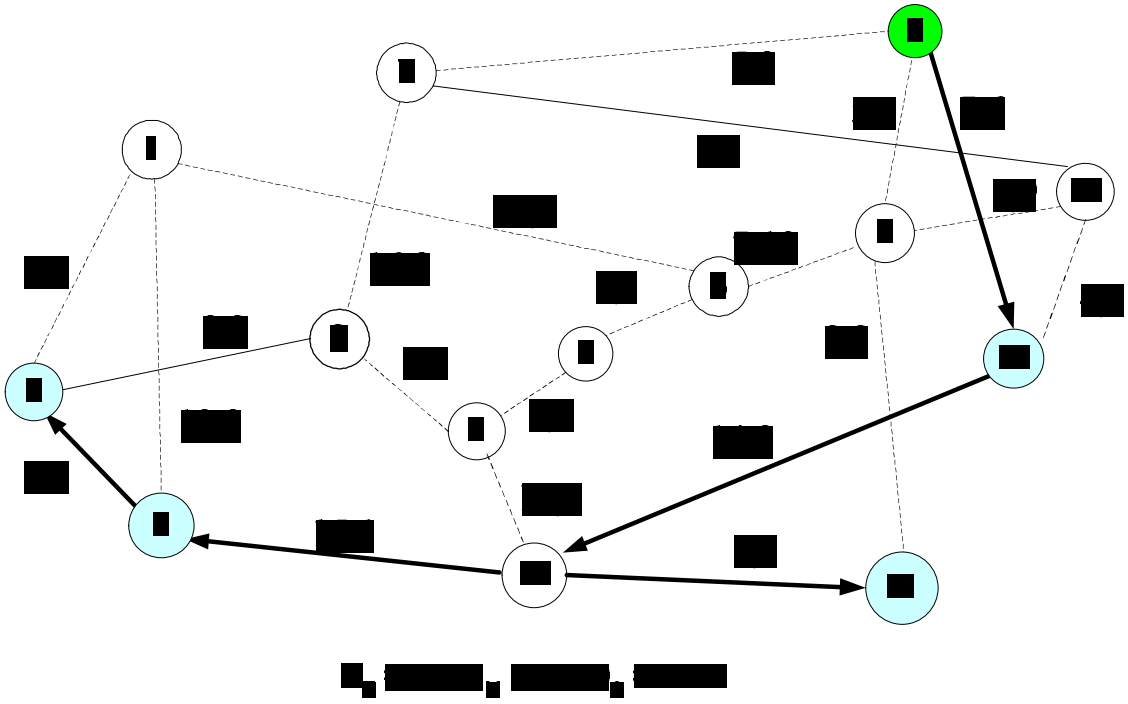


(b)



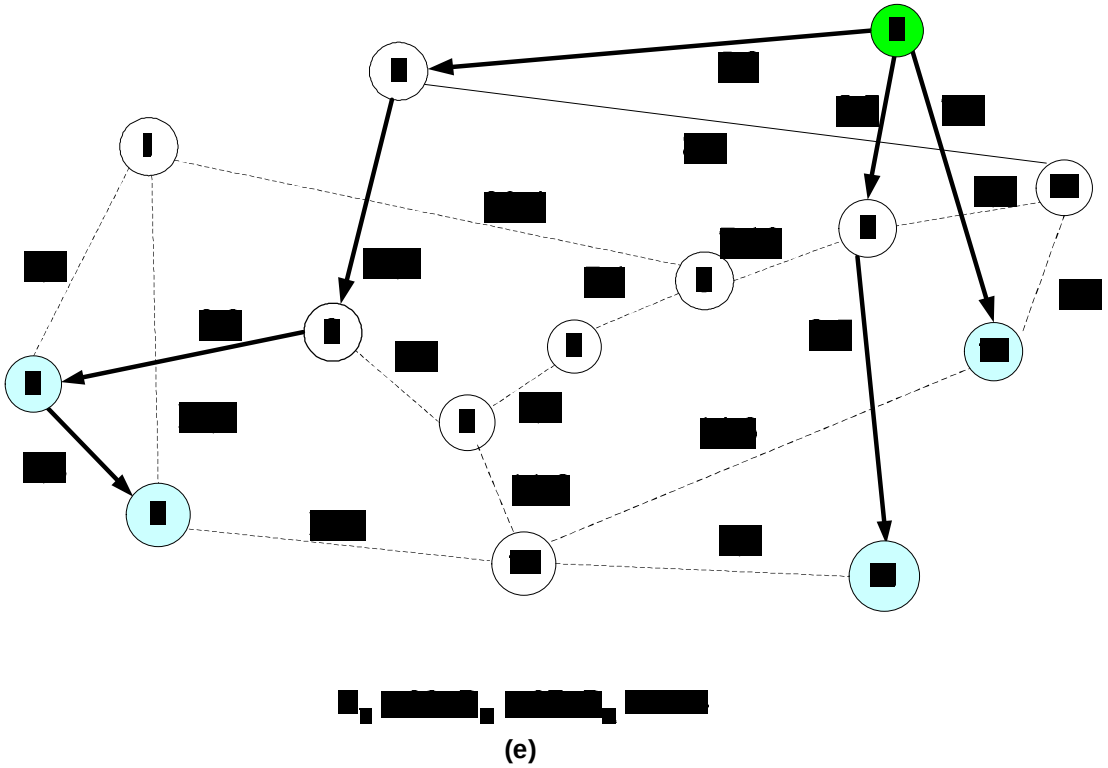
[redacted]

(c)



[redacted]

(d)



**(e)**

**Figura 3.2.** Cinco soluciones alternativas para el grupo multicast de la Figura 3.1

Note que las cinco alternativas son soluciones no dominadas. De hecho, por búsqueda exhaustiva, puede comprobarse que estas 5 soluciones forman parte del conjunto Pareto del problema. Por lo tanto, son soluciones óptimas en un sentido multiobjetivo.

Un conjunto de soluciones óptimas como este puede ser de gran utilidad en la práctica, pues la elección de la mejor opción puede ser tomada a posteriori y según las necesidades de cada caso. De la formulación matemática se puede concluir que esta gran variedad de soluciones óptimas solo puede ser obtenida si las funciones objetivos son tratadas en forma independiente. Es decir, no forman una función objetivo escalar combinada a través de una combinación lineal, ni tampoco son tratadas como simples restricciones. De esta forma, el enfoque multiobjetivo elude el problema de definir los pesos relativos para cada objetivo o hacer restricciones a priori, hallando todo un conjunto de soluciones óptimas.

Dado que el problema del árbol multicast de costo mínimo en una red de computadoras es un problema NP-completo [KOM93a], el problema formulado en esta

sección también lo es. Esencialmente, la intratabilidad del problema se debe a la existencia de un número de árboles que crece exponencialmente con el número de nodos de la red, y las soluciones óptimas solo pueden hallarse por búsqueda exhaustiva [KOM93a]. Evidentemente, dicha técnica no tiene practicidad alguna, por lo que se necesitan métodos alternativos que no exijan la exploración completa del espacio de búsqueda. En general, estos métodos no tienen por objetivo hallar todas las soluciones del conjunto Pareto óptimo, sino simplemente encontrar una aproximación lo suficientemente buena.

La complejidad del espacio de búsqueda, su tamaño y las últimas propuestas presentadas de algoritmos genéticos multiobjetivo [ZIT99], hacen de estos últimos candidatos ideales para ser aplicados en este ámbito. Debido a que estos algoritmos no han sido ampliamente estudiados en aplicaciones de redes de computadoras, el presente trabajo es un aporte original y útil en el estudio de MOEAs.

La ventaja principal de los MOEAs es su paralelismo inherente, pues son capaces de mantener todo un conjunto de soluciones (al mantener una población) y por tanto pueden obtener varios miembros del conjunto Pareto óptimo en una única corrida. Además, no requieren mayor información sobre el espacio de búsqueda; basta con especificar los objetivos y la manera de calcularlos. Tampoco presentan dificultades relacionadas con la complejidad del espacio de búsqueda, es decir, son bastante generales y robustos [ZIT99].

## 3.4 Resumen del Capítulo

En el presente Capítulo se ha formulado el problema de enrutamiento multicast como un Problema de Optimización Multiobjetivo, donde las funciones objetivos son formuladas explícitamente y deben ser optimizadas simultánea e independientemente. El siguiente Capítulo presenta una breve introducción a los cinco MOEAs estudiados en le presente trabajo, que son:

1. NSGA (Non dominated Sort Genetic Algorithm);
2. NSGA2 (Non dominated Sort Genetic Algorithm 2);
3. cNSGA2 (Controlled Non dominated Sort Genetic Algorithm 2);
4. SPEA (Strength Pareto Evolutionary Algorithm);
5. SPEA2 (Strength Pareto Evolutionary Algorithm 2).

# 4 Algoritmos Evolutivos MultiObjetivos

## 4.1 Introducción

Los algoritmos considerados para este trabajo son los Algoritmos Evolutivos MultiObjetivos o MOEAs 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 continúa 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, como se verá en la Figura 4.2.

La inicialización 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 [HOR86]. 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 4.1.

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

**Figura 4.1** Procedimiento utilizado para la construcción de un árbol $T_p$.

El operador genético utilizado en este trabajo, 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 4.2.

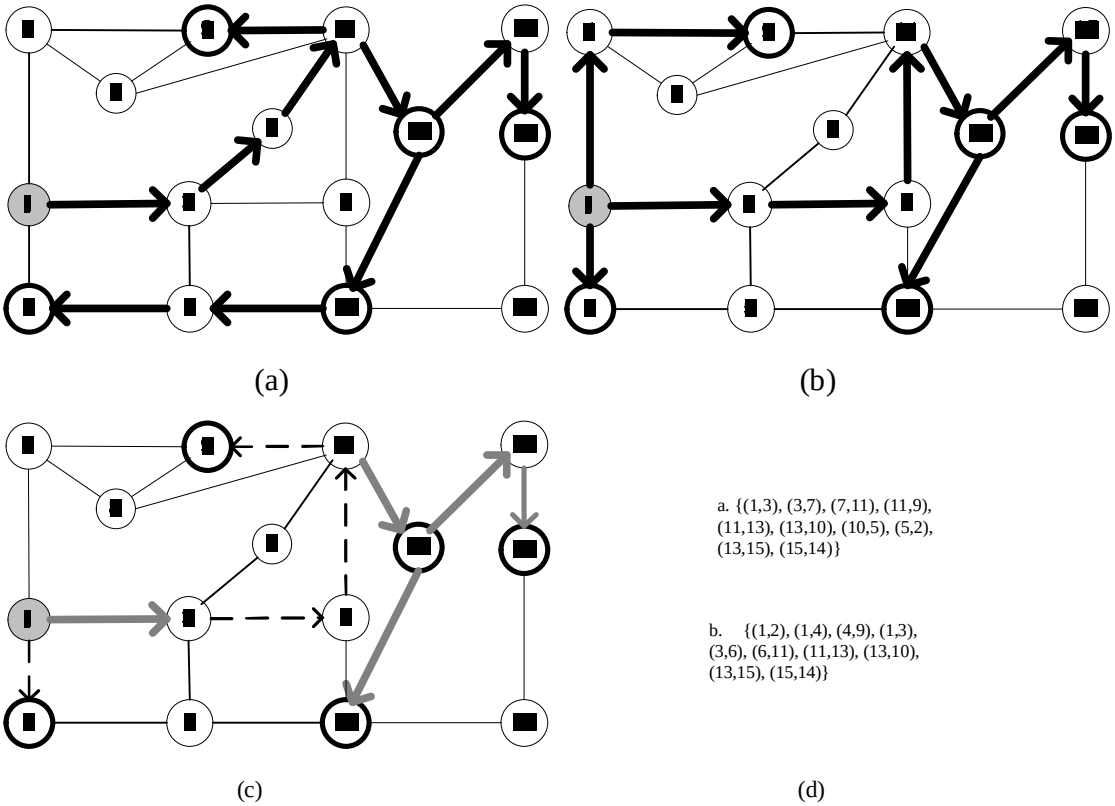



**Figura 4.2.** (a), (b) Individuos seleccionados, sobre los cuales se aplica el operador genético propuesto. (c) Resultado del operador de cruzamiento. Los enlaces discontinuos son seleccionados aleatoriamente para unir los sub-árboles, mientras que los enlaces continuos pertenecen a ambos padres y son heredados por el nuevo individuo. (d) Representación de los árboles de las figuras (a) y (b).

## 4.2 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 estos, se sugiere leer las publicaciones referenciadas.

### 4.2.1 **NSGA** (Non dominated Sort Genetic Algorithm)

Este algoritmo fue propuesto por Srinivas y Deb [SRI94]. 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. El pseudocódigo de este algoritmo se presenta en la Figura 4.3.

```
- 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 4.3.** Pseudocódigo del NSGA

### 4.2.2 **SPEA** (Strength Pareto Evolutionary Algorithm)

Este algoritmo fue introducido por el Zitzler y Thiele [ZIT99]. El SPEA utiliza una población externa de individuos no dominados $P_{nd}$. En cada generación, se copian los individuos no dominados de P a $P_{nd}$ y se borra de este las soluciones dominadas. Para cada individuo de $P_{nd}$, se computa un valor de fuerza (*strength*) proporcional al número de las soluciones a las cuales cada individuo domina.

En SPEA, el *fitness* de cada miembro de la población actual P se computa según las fuerzas de todas las soluciones no dominadas de $P_{nd}$ que la dominen. El pseudocódigo de este algoritmo se presenta en la figura 4.4.

- 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 $P_{nd}$
- Calcular la adaptabilidad de los individuos de P y $P_{nd}$
- Seleccionar individuos del conjunto $P \vee P_{nd}$
- Aplicar los operadores de cruzamiento y mutación.

} **mientras** el criterio de parada no sea alcanzado.

**Figura 4.4.** Pseudocódigo del SPEA

### 4.2.3 **SPEA2** (Strength Pareto Evolutionary Algorithm 2)

SPEA2 tiene las siguientes diferencias principales con respecto a su precursor [ZIT01]: (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 de $P_{nd}$ 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 4.5.

- 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 $P_{nd}$
- Calcular la adaptabilidad de los individuos de P y $P_{nd}$
- Seleccionar individuos del conjunto $P_{nd}$
- Aplicar los operadores de cruzamiento y mutación.

} **mientras** el criterio de parada no sea alcanzado.

**Figura 4.5.** Pseudocódigo del SPEA2

### 4.2.4. NSGA2 (Non dominated Sort Genetic Algorithm 2)

Deb et al. [DEB00] propusieron una versión revisada del NSGA [SRI94], 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 4.6.

- 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 Fi
- I=1

**Mientras** $|P_{t+1}| < N$ { // N número de individuos deseados en P

- Calcular adaptabilidad de cada individuo en Fi
- $P_{t+1} = P_{t+1}$ U Fi
- I = I +1

}

- Ordenar Pt+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 4.6.** Pseudocódigo del NSGA2

### 4.2.5. cNSGA2 (Controlled Non dominated Sort Genetic Algorithm 2)

Deb y Goel propusieron una variación del NSGA 2, llamado *Controlled Non-dominated Sorting Genetic Algorithms* [DEB01]. 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 $n_i = r \cdot n_{i-1}$, donde r es la razón geométrica. El pseudocódigo de este algoritmo se presenta en la figura 4.7.

```
- 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.
        - ordenar Rt, considerando dominancia y obtener frentes Fi
        - 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 4.7.** Pseudocódigo del cNSGA2

## 4.3 Resumen del Capítulo

En el presente capítulo se ha presentado una breve introducción a los MOEAs más estudiados en los últimos tiempos. Estos MOEAs han sido utilizados para resolver problemas de los más variados. Es importante destacar que el algoritmo NSGA corresponde a los algoritmos de primera generación, por no ser elitista. El *elitismo* consiste en seleccionar los individuos no dominados (buenos padres) de una generación y conservarlos hasta la siguiente generación. Esto hace que los buenos padres o las buenas soluciones no se pierdan en el proceso de cruzamiento y selección.

Los algoritmos NSGA2, cNSGA2, SPEA y SPEA2, son algoritmos de segunda generación porque utilizan el concepto de *elitismo,* por ende es de esperar que estos algoritmos de segunda generación obtengan los mejores resultados. La principal diferencia entre los SPEAs (SPEA y SPEA2) y los NSGA2s (NSGA2 y cNSGA2) es la

manera en que implementan el *elitismo*. En el caso de los SPEAs, utilizan una población externa donde se guardan los individuos no dominados, por la otra parte, los NSGA2s utilizan una clasificación por frentes, donde el primer frente correspondes a los individuos no dominados que pasarán directamente a la siguiente generación.

# 5 Comparaciones Estáticas

## 5.1 Introducción

Las pruebas experimentales presentadas en este capítulo, se realizaron con la 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 6Mbps representan el retardo del mismo (ver Figura 5.1).

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 de 15. En la Tabla 5.1 se presentan los grupos multicast utilizados para cada prueba.

| | **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} |

**Tabla 5.1**: Grupos multicast utilizados durante las pruebas

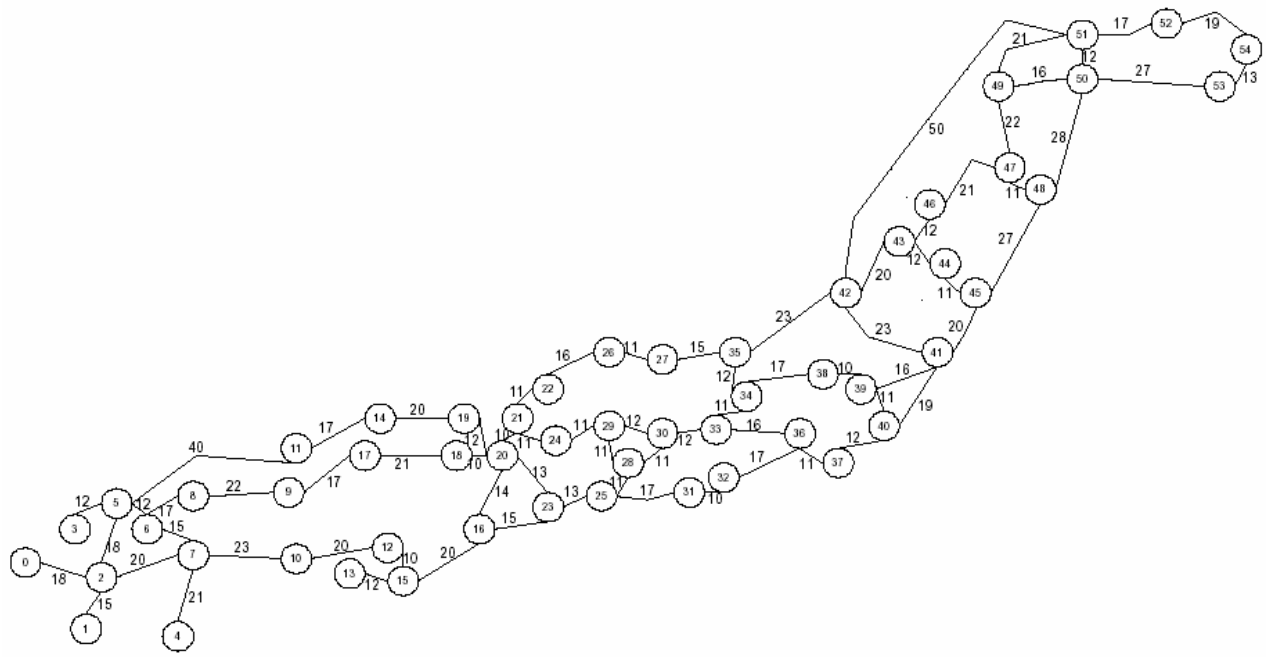


**Figura 5.1.** Red de la Nipon Telegraph and Telephone, Co utilizada en las simulaciones presentadas en el presente trabajo.

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.

La ejecución de los algoritmos se realizó en una NoteBook HP-Compaq NX9010 Pentium4 2.8 GHz, 512MB de memoria RAM, utilizando el compilador Borland C++ 5.02.

## 5.2 Procedimiento de comparación

El procedimiento de comparación utilizado fue el siguiente:

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 con las 15 corridas del algoritmo NSGA)
   - $\boldsymbol{Y}_{\text{NSGA2}}$ (frente Pareto con las 15 corridas del algoritmo NSGA2)
   - $\boldsymbol{Y}_{\text{cNSGA2}}$ (frente Pareto con las 15 corridas del algoritmo CNSGA2)
   - $\boldsymbol{Y}_{\text{SPEA}}$ (frente Pareto con las 15 corridas del algoritmo SPEA)
   - $\boldsymbol{Y}_{\text{SPEA2}}$ (frente Pareto con 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 nodominadas, y así se forma una aproximación del Frente Pareto Optimo $\hat{\boldsymbol{Y}}_{true}$.

## 5.3 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 $\boldsymbol{p} \succ \boldsymbol{q}$, $\forall\ \boldsymbol{p} \in \boldsymbol{Y}_{\text{ALGORITMO}_i}$, 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. Por ejemplo, la Tabla 5.3 muestra que las soluciones del SPEA dominan a una solución del NSGA (ver última fila, 4ta. columna).

**Prueba 1** (9 destinos multicast).

| | $\in Y_{true}$ | $Y_{true} \succ$ | $\lvert Y_{algoritmo}\rvert$ | $\%(\in Y_{true})$ |
|---|---|---|---|---|
| $\boldsymbol{Y}_{\text{cNSGA2}}$ | 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 5.2.** Comparación de las soluciones de los algoritmos con $Y_{true}$

| $\boldsymbol{Y}_i$ \ $\boldsymbol{Y}_j$ | $\boldsymbol{Y}_{\text{cNSGA2}}$ | $\boldsymbol{Y}_{\text{SPEA2}}$ | $\boldsymbol{Y}_{\text{NSGA}}$ | $\boldsymbol{Y}_{\text{NSGA2}}$ | $\boldsymbol{Y}_{\text{SPEA}}$ |
|---|---|---|---|---|---|
| $\boldsymbol{Y}_{\text{cNSGA2}}$ | | 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 5.3.** 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 $Y_{true}$. Además, se observa en la Tabla 5.3 que una solución del NSGA no está en $Y_{true}$, dada la presencia de un "1" en la columna que corresponde al NSGA.

**Prueba 2** (14 destinos multicast).

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

**Tabla 5.4.** 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 | 0 | 0 |
| $Y_{SPEA2}$ | 0 | | 3 | 0 | 0 |
| $Y_{NSGA}$ | 0 | 0 | | 0 | 0 |
| $Y_{NSGA2}$ | 0 | 0 | 3 | | 0 |
| $Y_{SPEA}$ | 0 | 0 | 3 | 0 | |

**Tabla 5.5.** Cantidad de soluciones en las cuales un algoritmo domina a otro

Nuevamente se puede observar que el SPEA2, NSGA2, SPEA y cNSGA2 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 rendimiento que tienen los cuatro algoritmos de segunda generación, cuando comparamos con el NSGA de primera generación. Sin embargo, hasta aquí todos los algoritmos de segunda generación tienen igual desempeño por lo que todavía no puede concluirse cual es el más recomendable. Para ello, se deberán considerar problemas más difíciles, como se muestra a continuación.

**Prueba 3** (19 destinos multicast).

| | $\in Y_{true}$ | $Y_{true} \prec$ | $\lvert Y_{algoritmo} \rvert$ | $\%(\in Y_{true})$ |
|---|---|---|---|---|
| $Y_{cNSGA2}$ | 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 5.6.** 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 5.7.** 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 $Y_{true}$. Con esto el NSGA2 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** (24 destinos multicast).

| | $\in Y_{true}$ | $Y_{true} \succ$ | $\lvert Y_{algoritmo} \rvert$ | $\%(\in Y_{true})$ |
|---|---|---|---|---|
| $Y_{cNSGA2}$ | 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 5.8.** 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 5.9.** 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 $Y_{true}$. Nuevamente, sobresalen los algoritmos SPEA y SPEA2.

## 5.4 Conclusiones.

Se puede observar que los algoritmos NSGA2, cNSGA2, SPEA, SPEA2 poseen el mismo rendimiento cuando el grupo musticast es pequeño (Prueba 1 y 2), dado que, el NSGA no llega a todas las soluciones en $Y_{true}$.

En la Prueba 3 y 4 que consideran un mayor número de nodos destinos, ya se puede apreciar una diferencia entre los algoritmos de segunda generación, notándose un mejor rendimiento de los algoritmos tipo SPEA en cualquiera de sus dos versiones (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 conclusión, se puede recomendar la utilización de algoritmos tipo SPEA en cualquiera de sus versiones (SPEA y SPEA2) para la resolución del problema de ingeniería de tráfico multicast, en un ambiente estático.

# 6 Escenarios de Prueba Dinámicos y Políticas de Selección de Individuos

## 6.1 Escenarios Dinámicos

Los escenarios dinámicos consisten en la simulación de entradas y salidas de grupos multicast a lo largo del tiempo. Los algoritmos son sometidos a estos escenarios de prueba, así se tienen apreciaciones más realistas, a diferencia de las pruebas estáticas, donde considerábamos una red totalmente descargada. Como actualmente en la literatura no se tienen escenarios de prueba dinámicos típicos o *test bed's*, aquí se proponen tres escenarios dinámicos de prueba con distintas distribuciones de carga. La ventaja de tener escenarios de prueba típicos es que facilita la comparación entre los distintos algoritmos sobre Ingeniería de Trafico Multicast.

### 6.1.1 Algoritmo de creación de los escenarios dinámicos

Para las pruebas que se muestran en el siguiente capítulo se utilizaron escenarios dinámicos, que consisten en simular el funcionamiento de una red en diferentes condiciones de carga. Los escenarios se forman a partir del siguiente algoritmo.

**Desde** i = 1: $\Psi$ ,

grupo(i) = ***groupGenerator***($|N|_{min}$, $|N|_{max}$ ); //retorna un grupo multicast
Tinicio(i) = tiempo de inicio aleatorio con distribución uniforme donde el grupo i es atendido.
Tfinal(i) = Tinicio(i) + tiempo de duración de la comunicación.
demand(i) = valor aleatorio con distribución uniforme entre $|\phi|_{min}$ y $|\phi|_{max}$
**Fin desde**

**Figura 6.1.** Pseudocódigo del generador de escenarios dinámicos.

La subrutina ***groupGenerator*** encargada de generar el grupo multicast es resumida en la Figura 6.2, donde podemos apreciar el algoritmo de generación de escenarios de prueba, consta de 5 parámetros iniciales, que son:

| | |
|---|---|
| $\Psi$ | Número de pedidos multicast |
| $\|N\|_{min}$ | Longitud mínima para los grupos. |
| $\|N\|_{max}$ | Longitud máxima para los grupos. |
| $\|\phi\|_{min}$ | Demanda mínima para cada grupo en Mbps |
| $\|\phi\|_{max}$ | Demanda máxima para cada grupo en Mbps. |

Con los valores de $|N|_{min}$ y $|N|_{max}$, se generan aleatoriamente con una distribución uniforme la cantidad de nodos involucrados en el grupo. También el Tiempo Inicial es un número aleatorio de distribución uniforme. No obstante, el tiempo que un grupo se mantiene en la red, es aleatorio con distribución exponencial.

***groupGenerator***( $|N|_{min}$ , $|N|_{max}$ ) // genera un grupo de longitud aleatoria entre $|N|_{min}$ y $|N|_{max}$.
long_grupo = longitud del grupo, aleatorio con valores entre $|N|_{min}$ y $|N|_{max}$.
**Desde** i=1:long_grupo,
    ptr = se selecciona un nodo aleatorio factible para el grupo multicast; nodos
        que no son parte del grupo multicast actual.
    grupo(i) = ptr  // se agrupa el nodo al grupo multicast.
**Fin desde**

**Figura 6.2.** Subrutina ***gourpGenerator.***

## 6.1.2 Escenarios de Baja Carga, Alta Carga y en Saturación.

Los parámetros utilizados para cada uno de los escenarios son mostrados en la Tabla 6.1. La demanda acumulada y promedio para cada uno de los escenarios mostrados en las figuras 6.3, 6.4 y 6.5, es calculada de la siguiente manera:

Demanda Acumulada:

$$D_t = \sum_{i=1}^{\pi} di \qquad \textbf{(6.1)}$$

Siendo $\pi$, el numero de grupos activos en el instante t y di la demanda de cada grupo.

La demanda acumulada de la red, nos permite tener una apreciación de la carga que posee la red en un determinado tiempo. Es importante destacar que la demanda acumulada de la red se mantiene constante entre dos eventos. Siendo un evento la entrada o salida de un grupo multicast de la red, esto es, en cada evento tenemos un número potencialmente diferente de grupos en la red y este número se mantiene hasta el siguiente evento. Por lo tanto, la demanda acumulada de la red $D_t$ es la suma de cada una de las demandas en tráfico por la red, en un determinado tiempo.

Demanda promedio:

$$\overline{D} = \frac{1}{2 \times \Psi} \sum_{t=0}^{2 \times \Psi} \phi_{\Delta t} \qquad \textbf{(6.2)}$$

donde $\phi_{\Delta t}$ es la demanda constante en el lapso de tiempo $\Delta$t.

La demanda promedio de la red, nos muestra la carga que posee la red a lo largo de todo el escenario. Esto permite distinguir claramente los escenarios de a cuerdo al nivel de la demanda promedio. Para realizar el cálculo de la demanda promedio y recordando que la demanda acumulada en la red es constante entre dos eventos, podemos tomar estos lapsos constantes de la demanda acumulada como un solo valor $\phi_{\Delta\tau}$, y así hacer el promedio de $\phi_{\Delta\tau}$ de acuerdo al número de lapsos constantes $\Delta$t, que es dos veces el número total de pedidos multicast $\Psi$, puesto que cada pedido entra y sale una vez. Entonces, la demanda acumulada promedio es calculada como el promedio, de la demanda acumulada Dt en los lapsos $\Delta$t en los que la demanda acumulada es constante con un valor que se denota $\phi_{\Delta\tau}$.

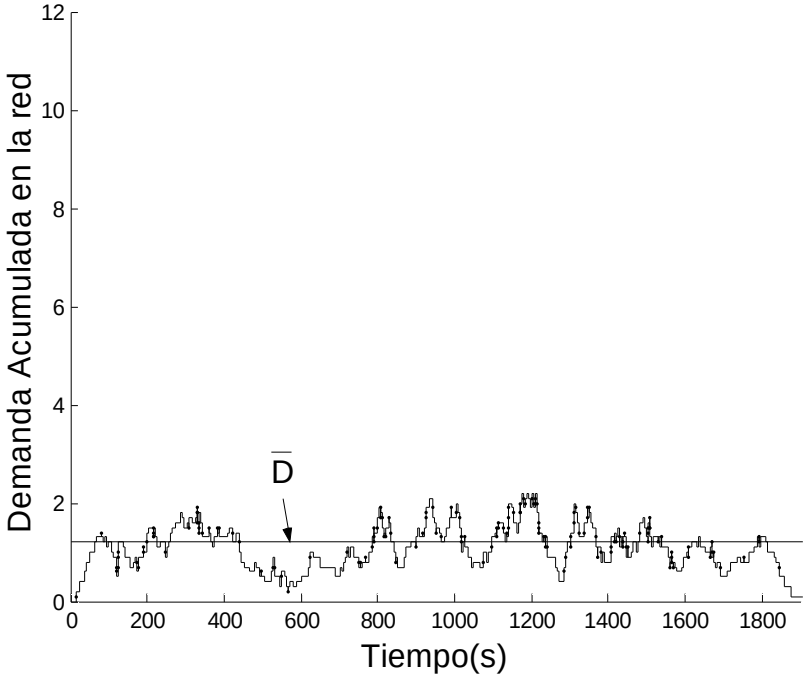



**Figura 6.3.** Escenario de baja carga, muestra la demanda acumulada en la red en un determinado tiempo.

En la figura 6.3, podemos apreciar que el Escenario de prueba de baja carga es bastante relajado desde el punto de vista del tráfico.

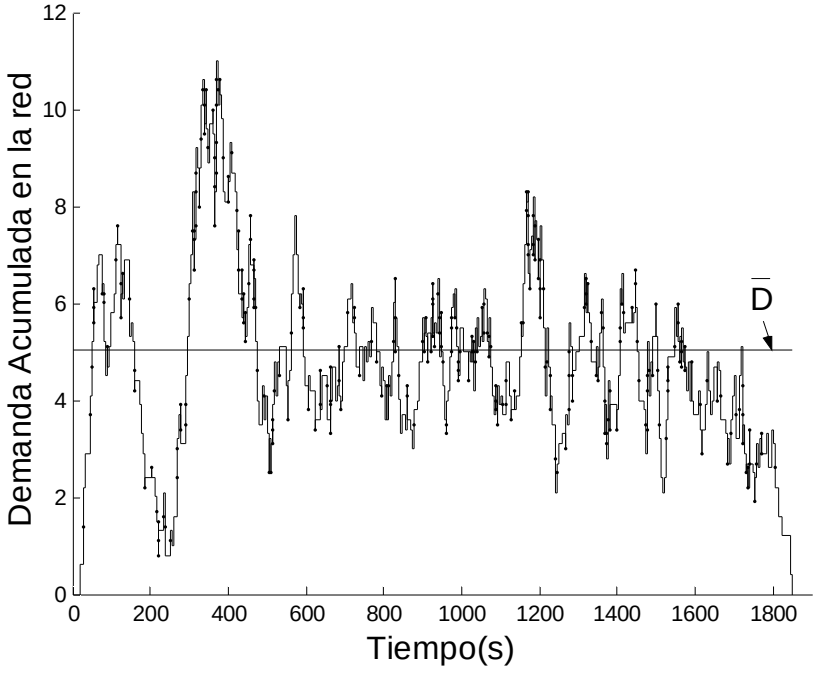



**Figura 6.4.** Escenario de alta carga.

Por su parte, se puede apreciar en la figura 6.4, un escenario de alta carga, bastante más exigente que el anterior desde el punto de vista de la demanda acumulada en la red. También cada grupo multicast consta de un número mayor de nodos destinos, lo que produce un mayor congestionamiento de la red.

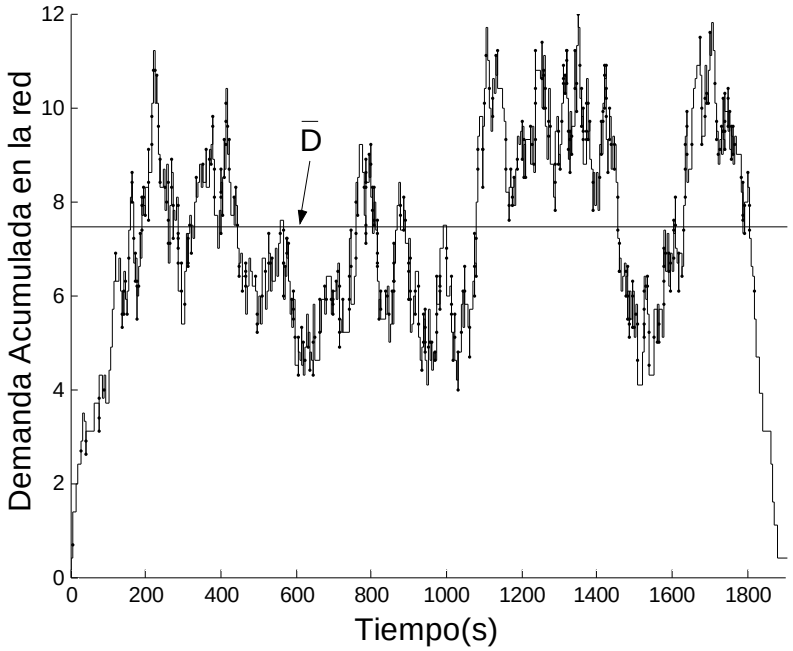



**Figura 6.5.** Escenario en saturación.

En la figura 6.5, tenemos la demanda acumulada vs tiempo, del escenario en saturación. Se puede apreciar que es bastante mas exigente que los anteriores pues se tienen 400 pedidos multicast y un mayor número de destinos, lo que genera una mayor carga en la red.

La Tabla 6.1 resume los parámetros utilizados en cada uno de los 3 escenarios a ser considerados en el presente trabajo.

| Escenarios | $\overline{D}$ | Parámetros | | | | |
|---|---|---|---|---|---|---|
| | | $\Psi$ | $\lvert N \rvert_{min}$ | $\lvert N \rvert_{max}$ | $\lvert \phi \rvert_{min}$ | $\lvert \phi \rvert_{max}$ |
| Baja carga | 1.208 | 200 | 4 | 10 | 0,1 | 0,2 |
| Alta carga | 5.050 | 300 | 10 | 25 | 0,2 | 0,8 |
| En saturación | 7.463 | 400 | 10 | 35 | 0,2 | 0,8 |

**Tabla 6.1**. Muestra los valor de $\overline{D}$ de cada escenario y los parámetros utilizados para la creación de los mismos.

## 6.2 Políticas de Selección de Individuos

Como los MOEAs proporcionan un conjunto de soluciones Pareto, se necesita elegir solo una de ellas como respuesta a cada entrada de un nuevo grupo multicast. Aquí surge una pregunta interesante. ¿Cuál es la solución óptima, siendo que todas las soluciones pertenecen al frente Pareto?
De manera a intentar encontrar una respuesta a esta problemática, se proponen a continuación diversas estrategias de selección del árbol de ruteo solución a una nueva demanda multicast.

### 6.2.1 Selecciones Estáticas

Las selecciones estáticas propuestas son del tipo “Orden lexicográfico” conforme fuera explicado en la sección 2.5. En las pruebas que siguen se utilizarán las siguientes combinaciones.

a. 1ero. Alfa, 2ndo. Costo, 3ro. Delay medio, la que denotaremos $\alpha C$;
b. 1ero. Costo, 2ndo. Alfa, 3ro. Delay medio, la que denotaremos $C\alpha$;
c. 1ero. Alfa, 2ndo. Delay medio, 3ro. Costo, la que denotaremos $\alpha D_A$;
d. 1ero. Delay medio, 2ndo. Alfa, 3ro. Costo, la que denotaremos $D_A\alpha$;

### 6.2.2 Selección Semi-Estática.

Las Selección semi-estática responden al siguiente algoritmo de selección propuesto. El administrador define un punto aceptable de trabajo ($\alpha_m, C_m, D_{am}$). Estos parámetros determinan la peor calidad de servicio que todavía resulta aceptable. Con estos parámetros el algoritmo selecciona dinámicamente la solución más conveniente entre el grupo de soluciones Pareto.

**Funcionamiento del Algoritmo.**

Si consideramos los parámetros ingresados por el administrador de la red, como límites superiores, entonces podríamos escribir la siguiente tabla “de verdad” que contiene todas las posibles situaciones, que podríamos encontrar.

| Caso | Alfa | Costo | Delay |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 |
| 3 | 0 | 1 | 0 |
| 4 | 0 | 1 | 1 |
| 5 | 1 | 0 | 0 |
| 6 | 1 | 0 | 1 |
| 7 | 1 | 1 | 0 |
| 8 | 1 | 1 | 1 |

**Tabla 6.2.** Casos posibles del algoritmo.

Representamos con un “0”, si el objetivo no a alcanzado su límite superior, y con un “1” si el objetivo a alcanzado el límite superior. Queda claro que el objetivo que ha alcanzado su límite superior, debe tener la mayor prioridad a la hora de elegir la solución del conjunto Pareto. También es claro que aquel objetivo que no haya alcanzado su límite, puede tener la menor prioridad a la hora de elegir el árbol de enrutamiento.

Un problema se presenta cuando tenemos más de dos soluciones con la misma prioridad. Sean dos o tres objetivos que no han alcanzado su límite superior (ya atienden valores máximos) o bien que han superado su límite superior. En estos Casos necesitamos alguna forma de desempatar entre estos objetivos en igual situación. La propuesta es la siguiente:

**Caso 1:** En este caso todos los objetivos son satisfechos, ningún objetivo ha alcanzado su límite superior, entonces vemos cual está más cerca de su límite, o sea el mayor entre los parámetros a, b, c, será el de mayor prioridad, el siguiente más grande será el segundo en prioridad, y así sucesivamente. Los parámetros a, b y c se definen como:

$$a = \frac{D_c * 100}{D_m} \quad \textbf{(6.3)}$$

donde $D_c$ es el promedio del retardo actual y $D_m$ es el límite de retardo definido por el administrador.

$$b = \frac{C_c * 100}{C_m} \quad \textbf{(6.4)}$$

donde $C_c$ es el promedio del costo actual y $C_m$ es el límite de costo definido por el administrador.

$$c = \frac{\alpha_c * 100}{\alpha_m} \quad \textbf{(6.5)}$$

donde $\alpha_c$ es el promedio de la utilización de los enlaces actual y $\alpha_m$ es el límite de la utilización de los enlaces definido por el administrador.

Si el empate persiste, se desempata en forma aleatoria.

*Ejemplo*: Sea a = 60, b = 30, c = 80 entonces

Al ordenar tenemos que c > a > b, entonces el criterio de selección utilizado será del tipo “orden lexicográfico” con:

Primero Alfa ($\alpha$)
Segundo Delay($D_A$)
Tercero Costo(C)

**Casos 2, 3 y 5:** En estos casos tenemos un objetivo que ha alcanzado su límite superior, y dos que no lo han hecho. Lo lógico es dar mayor prioridad al objetivo que ha alcanzado el límite superior. En caso de empates, desempatar aquellos dos objetivos restantes de la misma manera que en el caso 1.

**Caso 8:** Aquí tenemos que todos los objetivos han llegado a su límite superior, la idea es calcular los parámetros a, b y c como se muestra a continuación:

$$\alpha = 100 - \frac{D_m * 100}{D_c} \quad \textbf{(6.6)}$$

$$c = \frac{(\alpha_c - \alpha_m) * 100}{1 - \alpha_m} \quad \textbf{(6.7)}$$

$$b = 100 - \frac{C_m * 100}{C_c} \quad \textbf{(6.8)}$$

Así se obtiene el porcentaje del objetivo que se ha rebasado de su límite superior. Es importante notar que sabemos que el $\alpha$ máximo alcanzable siempre será 1, entonces utilizamos otra forma de evaluar el objetivo.

Una vez calculados estos parámetros se podrá saber cual de los objetivos ha superado en un mayor porcentaje su límite, dicho objetivo será el primero en optimizar.

**Casos 4, 6 y 7:** En estos casos tenemos dos objetivos que han llegado a su límite superior y un objetivo por debajo de su límite. Es lógico dejar con la menor prioridad al objetivo que no ha alcanzado su límite. Para decidir cual de los dos objetivos que han rebasado su límite tendrá la mayor prioridad, usamos el mismo criterio que el caso 8, arriba descrito.

*Observaciones:*

$$D_C = \frac{\sum_{i=1}^{\pi} d_i}{\sum_{i=1}^{\pi} |N|_i} \quad \textbf{(6.9)}$$

*siendo π, el número de grupos multicast en un determinado momento, $d_i$ el delay medio del grupo i que esta en la red, y $|N|_i$ la longitud del grupo i.*

$$C_C = \frac{\sum_{i=1}^{\pi} ci}{\pi} \quad \textbf{(6.10)}$$

*siendo $c_i$, el costo del grupo i que esta en la red.*

$$\alpha_c = \frac{\alpha(i, j)}{|V|} \quad \textbf{(6.11)}$$

*siendo α (i,j), la sumatoria de α para cada enlace de la red en un determinado momento, y |V| la cantidad de nodos en la red.*

Para las simulaciones se utilizaron 3 puntos: [06, 14, 140] (SE1); [07, 15, 120] (SE2); [07, 13, 115] (SE3).

### 6.2.3 Selección dinámica DP (definido Peor Caso aceptable).

La Selección dinámica DP, selecciona las soluciones que se encuentran a menor distancia de la recta que pasa por el origen y el punto aceptable de trabajo ($\alpha_m, C_m, D_{am}$). Esto en la Figura 6.6 está representado por $d_1$ y $d_2$, para dos puntos de ejemplo. Así se puede garantizar que se optimizarán los 3 objetivos simultáneamente. El punto ($\alpha_m, C_m, D_{am}$) define el peor caso aceptable. En la actualidad algo similar es utilizado la implementar *DiffServ* [TAN03]. Con *DiffServ* se hace un tratamiento diferenciado entre los paquetes IP. Para el caso aquí presentado, permite hacer un tratamiento diferenciando entre los grupos multicast, logrando así optimizar una métrica más que otras de acuerdo a la necesidad de cada comunicación. Por ejemplo, para una transmisión multicast de video bajo demanda o educación a distancia, el punto ($\alpha_m, C_m, D_{am}$) tendría un valor de $D_{am}$ de unos pocos milisegundos y un valor de $C_m$

relativamente alto conforme a cuanto se esta dispuesto a pagar por una comunicación con bajo retardo medio. Para las simulaciones que siguen se utilizó el punto (07, 13, 115).

Como se puede ver en la figura 6.6, las soluciones que se encuentren dentro de la región de factibilidad del punto ($\alpha_m, C_m, D_{am}$) enmarcada por líneas de puntos, serán las primeras en tenerse en cuenta. Si existen soluciones dentro de esta región, entonces será elegida la solución con la menor distancia a la recta que pasa por origen y el punto del peor caso aceptable. En caso que no exista ninguna solución en la región de factibilidad, entonces se seleccionará una solución entre las soluciones no factibles exactamente como se hicieron con las demás políticas de selección.

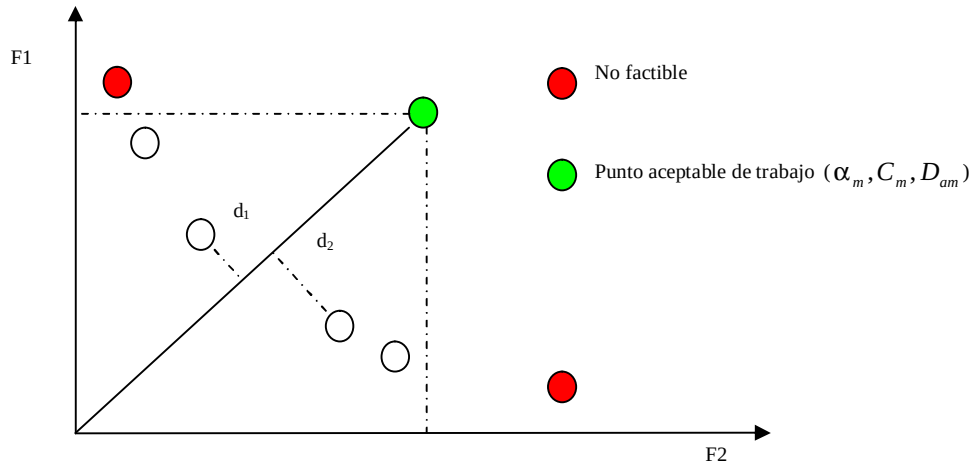



**Figura 6.6.** Selección dinámica DP.

## 6.2.4 **Selección dinámica DC** (respecto al origen de coordenadas).

La Selección dinámica DC, elige la solución que se encuentra más cerca del origen de coordenadas, tal como se indica en la figura 6.7. La ventaja principal de este tipo de selección es que no posee parámetros a priori, siendo *d* sencillamente la distancia de cada punto al origen de coordenadas. Es importante destacar que los ejes deben estar normalizados, para así no depender de los sistemas de medidas utilizados para cada eje coordenado.

La desventaja de la presente política radica en que no se puede discriminar entre varios tipos de comunicación, como si se puede hacer con el tipo de selección dinámica DP, arriba presentado.

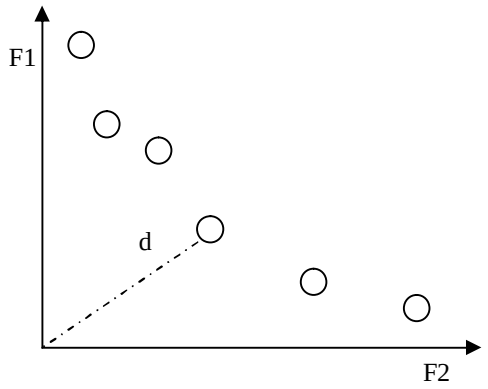



**Figura 6.7.** Selecciones dinámicas DC.

### 6.2.5 Comparación entre políticas.

En esta sección presentaremos una tabla comparativa entre las distintas políticas de selección de soluciones (ver tabla 6.3). En la misma se puede apreciar cual de las políticas posee parámetros a priori, así como el grado de complejidad de cada política de acuerdo al tiempo que tarda cada tipo de selección en elegir alguna solución para un conjunto Pareto de soluciones. La tercera columna nos muestra aquellos tipos de selección que tienen la posibilidad de implementar servicios diferenciados. Por último, tenemos la columna de portabilidad que nos dice si el tipo de selección es fácilmente transportable a otras topologías de redes. Por ejemplo la SemiEstática no es fácilmente transportable porque requiere de un conocimiento estadístico de la red, a diferencia de esta, en la política DP el parámetro a priori depende el grupo multicast en sí mismo y no de la red.

| Políticas | Parámetros a priori | Complejidad | DiffServ | Portabilidad |
|---|---|---|---|---|
| Estáticas | | Baja | | √ |
| SemiEstáticas | √ | Alta | | |
| DP | √ | Alta | √ | √ |
| DC | | Baja | | √ |

**Tabla 6.3**. Comparación de las políticas de selección.

### 6.2.6 Esquema de las políticas.

La siguiente tabla presenta un resumen de las políticas aquí presentadas.

| Políticas | Nombres | Siglas | Detalles particulares | | | |
|---|---|---|---|---|---|---|
| | | | Objetivo primario | Objetivo secundario | Objetivo de menor importancia | Observaciones: |
| Estáticas | E1 | $\alpha C$ | $\alpha_T$ | $C_T$ | $D_A$ | Las selecciones estáticas propuestas son del tipo "Orden lexicográfico" [4]. |
| | E2 | $C\alpha$ | $C_T$ | $\alpha_T$ | $D_A$ | |
| | E3 | $\alpha D_A$ | $\alpha_T$ | $D_A$ | $C_T$ | |
| | E4 | $D_A\alpha$ | $D_A$ | $\alpha_T$ | $C_T$ | |
| SemiEstáticas | | SE | En las selecciones semi-estáticas el administrador define el punto aceptable de trabajo ($\alpha_m, C_m, D_{am}$). Estos parámetros determinan la peor calidad de servicio todavía aceptable. Con estos parámetros el algoritmo selecciona dinámicamente la solución más conveniente entre el grupo de soluciones Pareto. | | | |
| Dinámicas | Peor Caso aceptable | DP | La política DP selecciona las soluciones que se encuentran a menor distancia de la recta que pasa por el origen y el punto aceptable de trabajo ($\alpha_m, C_m, D_{am}$). Esto nos garantiza poder optimizar los 3 objetivos simultáneamente (Figura 4). | | | |
| | Origen de Coordenadas | DC | La política DC elige la solución que se encuentra más cerca del origen. No posee parámetros a priori (ver Figura 5). | | | |

**Tabla 6.4.** Resumen de las políticas de selección.

## 6.3 Resumen del Capítulo

En el presente capítulo se expuso un algoritmo utilizado para la generación de escenarios dinámicos. Como en la literatura no se cuentan con escenarios de prueba dinámicos típicos, aquí se proponen tres, uno de baja carga, otro de alta carga y por último un escenario en saturación.

Como todavía no queda claro que política de selección de una solución del frente Pareto es la más apropiada en un contexto dinámico, aquí se proponen 7 políticas de selección, que serán comparadas experimentalmente en el siguiente capítulo.

# 7 Pruebas Dinámicas y Conclusiones

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 diseccionados, conforme se mostró en la Figura 5.1.

## 7.1 MOEA a Recomendar

Para cada una de las pruebas se realizaron 10 corridas. Los resultados de las comparaciones de los algoritmos se muestran a continuación. En las tablas que siguen las columnas de $\alpha_p$ contienen el promedio de los $\alpha_T$ para cada árbol durante toda la simulación del escenario dinámico, de todas las corridas, esto se puede ver en la ecuación 7.1, donde consistentemente $\Psi$ representa el número de pedidos multicast durante todo el escenario de prueba.

$$\alpha p = \frac{1}{10}\sum_{f=1}^{10}\left(\frac{1}{\psi}\sum_{w=1}^{\psi}\alpha_{T_{wf}}\right) \qquad \textbf{(7.1)}$$

donde $\alpha_{Twf}$ representa el promedio de la utilización máxima de los enlaces para cada árbol multicast durante toda la simulación en las 10 corridas.

Por su parte, las columnas $C_p$, contienen el promedio de los costos $C_T$ de cada árbol durante toda la simulación del escenario dinámico, de todas las corridas, conforme a la ecuación 7.2.

$$Cp = \frac{1}{10}\sum_{f=1}^{10}\left(\frac{1}{\psi}\sum_{w=1}^{\psi}C_{T_{wf}}\right) \qquad \textbf{(7.2)}$$

donde $C_{Twf}$ representa el costo promedio de cada árbol multicast durante toda la simulación en las 10 corridas.

Las columnas $D_p$ de las tablas que siguen, contienen el promedio del retardo de todos los árboles que se encuentran en un instante dado en la red durante toda la

simulación del escenario, siendo π el número de grupos multicast en la red en un instante dado.

$$D_p = \frac{1}{10}\sum_{f=1}^{10}\left(\frac{1}{\psi}\sum_{w=1}^{\psi} D_{A_{wf}}\right) \qquad \textbf{(7.3)}$$

Donde $D_{Awf}$ representa el promedio del retardo medio de cada árbol multicast durante toda la simulación en las 10 corridas.

La ultima columna, Na representa el número de grupos no aceptados en la red.

Para facilitar la lectura de los resultados que siguen, las casillas fueron coloreadas de la siguiente manera:

La mejor solución para una determinada columna (o función objetivo).

La segunda mejor solución para una determinada columna (o función objetivo).

| | Promedios | | | |
|---|---|---|---|---|
| | $\alpha_p$ | $C_p$ | $D_p$ | Na |
| SPEA2 | 0.11868 | 20.6331 | 113.2032 | 0 |
| SPEA | 0.11882 | 20.633 | 113.249 | 0 |
| CNSGA2 | 0.11855 | 20.633 | 113.1603 | 0 |
| NSGA2 | 0.11855 | 20.633 | 113.1424 | 0 |

**Tabla 7.1.** Comparación de MOEAs en baja carga.

| | Promedios | | | |
|---|---|---|---|---|
| | $\alpha_p$ | $C_p$ | $D_p$ | Na |
| SPEA2 | 0.57805 | 140.2427 | 115.2895 | 1 |
| SPEA | 0.57614 | 140.0172 | 115.7672 | 1 |
| CNSGA2 | 0.57841 | 140.0833 | 115.531 | 1 |
| NSGA2 | 0.57686 | 140.0906 | 115.5096 | 1 |

**Tabla 7.2.** Comparación de MOEAs en alta carga.

| | Promedios | | | |
|---|---|---|---|---|
| | $\alpha_p$ | $C_p$ | $D_p$ | Na |
| SPEA2 | 0.77147 | 222.7509 | 114.7587 | 12.7 |
| SPEA | 0.76702 | 223.1221 | 115.1808 | 12.6 |
| CNSGA2 | 0.76814 | 223.3169 | 114.8438 | 12.4 |
| NSGA2 | 0.7675 | 223.6509 | 114.8642 | 12.5 |

**Tabla 7.3.** Comparación de MOEAs en saturación.

En las tablas 7.1, 7.2 y 7.3 se observa claramente la diferencia de carga que existe entre los tres escenarios. La tabla 7.1, muestra los resultados obtenidos para el escenario de baja carga, donde ningún algoritmo rechazó algún pedido multicast. La tabla 7.2 exhibe el escenario de alta carga, donde todos los algoritmos rechazan un grupo multicast. En la tabla 7.3 se tienen los resultados del escenario en saturación, donde todos los algoritmos rechazan varios pedidos multicast, siendo que el algoritmo cNSGA2 obtiene en promedio menores valores de NA (grupos no admitidos) que el resto de los algoritmos, aunque la diferencia entre los mismo no es significativa.

Si bien los resultados de las pruebas estáticas demostraron (ver capítulo 5) un mejor desempeño de los algoritmos SPEA y SPEA2, en cantidad de soluciones en el frente Pareto calculado [FRA04], esto no implica un buen desempeño de los mismos en las simulaciones dinámicas por tres razones fundamentales:

1. Como es bien sabido en problemas de Optimización MultiObjetivo se tiene un conjunto de soluciones de compromiso. De este conjunto de soluciones se elige una de ellas, por lo que un algoritmo tenga 20% más soluciones en el frente Pareto, no tendrá mucha ventaja porque no necesariamente estas soluciones serán las escogidas. Lógicamente tienen más probabilidad de ser escogidas el 80% de soluciones coincidentes con las de otros algoritmos multiobjetivos. Por lo tanto, esto hace suponer que en las simulaciones dinámicas estos pequeños porcentajes no se podrán notar y tendremos un conjunto de algoritmos con desempeño muy similar.

2. Este pequeño porcentaje de cantidad de soluciones adicionales en el frente Pareto solamente se presenta cuando el número de grupo multicast es grande

[FRA04], y como nuestro número de grupo multicast es aleatorio entonces no siempre tendremos un grupo multicast grande.

3. El algoritmo ideal para las simulaciones dinámicas, será aquel que conozca el futuro de la dinámica de la simulación, esto es, puede que una solución no óptima sea una excelente solución para el futuro de la red. O que soluciones óptimas encontradas por aquellos algoritmos como SPEA y SPEA2, sean pésimas soluciones para el futuro de la red, y soluciones sub-optimas encontradas por los otros algoritmos sean excelentes soluciones.

Con estas tres razones expuestas, podemos concluir que mientras no exista una diferencia muy significativa en las pruebas estáticas, tendremos desempeños similares de los algoritmos en pruebas dinámicas. Por lo tanto, podemos entender que son las políticas de selección de soluciones el punto más importante a tratar si tenemos algoritmos multiobjetivos de segunda generación y de desempeño parecido, conforme se mostró en las Tabla 7.1, 7.2 y 7.3.

## 7.2 Política a Recomendar

Los resultados de las comparaciones de las políticas de selección se muestran a continuación en las Tablas 7.4 a 7.6. Para las columnas de promedios se utilizaron las ecuaciones 7.1, 7.2 y 7.3. En las columnas de desviación estándar se utilizaron las siguientes ecuaciones.

Para la comuna de $\alpha_d$ se hallo el promedio de las desviaciones estándares de las 10 corridas. Siendo $\alpha_p$ el promedio del $\alpha_T$ de un grupo multicast, conforme se indica en la ecuación 7.4.

$$\alpha_d = \frac{1}{10}\sum_{f=1}^{10}\sqrt{\frac{1}{\psi}\left(\sum_{w=1}^{\psi}(\alpha_{Twf}-\alpha_p)^2\right)} \qquad \textbf{(7.4)}$$

En la columna $C_d$ se tiene la desviación estándar del costo de un grupo multicast, donde Cp es el promedio del costo de los grupos multicast, conforme se muestra en la ecuación 7.5.

$$C_d = \frac{1}{10}\sum_{f=1}^{10}\sqrt{\frac{1}{\psi}\left(\sum_{w=1}^{\psi}(C_{Twf} - C_p)^2\right)} \qquad \textbf{(7.5)}$$

Finalmente la desviación estándar del retardo promedio de los grupos multicast esta dado por la ecuación 7.6.

$$D_d = \frac{1}{10}\sum_{f=1}^{10}\sqrt{\frac{1}{\psi}\left(\sum_{w=1}^{\psi}\left(D_{A_{wf}} - D_p\right)^2\right)} \qquad \textbf{(7.6)}$$

En las columnas Máximos, tenemos el promedio de los máximos valores alcanzados durante cada una de las simulaciones de $\alpha_p$, $C_p$, $D_p$ y NA.

Para facilitar la lectura de los resultados las casillas fueron coloreadas de la siguiente manera:

La mejor solución para una determinada columna
La segunda mejor solución para una determinada columna
La tercera mejor solución para una determinada columna

| | Promedios | | | | Desviación Estándar | | | | Máximos | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\alpha_p$ | $C_p$ | $D_p$ | Na | $\alpha_d$ | $C_d$ | $D_d$ | Na | $\alpha_p$ | $C_p$ | $D_p$ | Na |
| $\alpha D_A$ | 0.098772 | 25.77488 | 121.843 | 0 | 0.0288 | 9.9166 | 17.5498 | 0 | 0.196 | 54.55 | 203.828 | 0 |
| $C\alpha$ | 0.11865 | 20.63303 | 113.189 | 0 | 0.0378 | 7.8572 | 16.0793 | 0 | 0.233 | 41.8 | 170.098 | 0 |
| $\alpha C$ | 0.095282 | 23.62738 | 128.947 | 0 | 0.0282 | 9.0609 | 18.3066 | 0 | 0.175 | 50.325 | 231 | 0 |
| $D_A\alpha$ | 0.141083 | 25.25543 | 103.48 | 0 | 0.0493 | 9.7218 | 14.9986 | 0 | 0.267 | 52.65 | 146.425 | 0 |
| SE1 | 0.141135 | 25.24653 | 103.508 | 0 | 0.0493 | 9.7379 | 15.0087 | 0 | 0.267 | 52.725 | 146.306 | 0 |
| SE2 | 0.141063 | 25.2387 | 103.507 | 0 | 0.0493 | 9.7459 | 15.0053 | 0 | 0.267 | 52.75 | 146.298 | 0 |
| SE3 | 0.141148 | 25.25843 | 103.509 | 0 | 0.0493 | 9.7557 | 15.0062 | 0 | 0.267 | 52.65 | 146.369 | 0 |
| DP | 0.1347 | 21.0742 | 107.876 | 0 | 0.0419 | 7.9961 | 15.7801 | 0 | 0.25 | 42.7 | 153.902 | 0 |
| DC | 0.10278 | 22.2371 | 113.971 | 0 | 0.0332 | 8.4328 | 15.7857 | 0 | 0.2 | 46.9 | 165.568 | 0 |

**Tabla 7.4.** Comparación de políticas de selección, en baja carga.

| | Promedios | | | | Desviación Estándar | | | | Máximos | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\alpha_p$ | $C_p$ | $D_p$ | Na | $\alpha_d$ | $C_d$ | $D_d$ | Na | $\alpha_p$ | $C_p$ | $D_p$ | Na |
| $D_A\alpha$ | 0.664573 | 167.5921 | 102.712 | 1.9 | 0.186 | 61.889 | 11.9021 | 0.85 | 1 | 376.025 | 145.361 | 2 |
| $\alpha D_A$ | 0.513128 | 166.7889 | 115.043 | 1 | 0.1561 | 61.235 | 12.5058 | 0 | 0.933 | 378.475 | 183.912 | 1 |
| $\alpha C$ | 0.494735 | 151.1135 | 123.139 | 1 | 0.1521 | 55.269 | 13.4615 | 0 | 0.892 | 337.3 | 178.197 | 1 |
| $C\alpha$ | 0.577365 | 140.1085 | 115.524 | 1 | 0.1733 | 51.413 | 13.1125 | 0 | 0.992 | 308.875 | 157.69 | 1 |
| SE1 | 0.552703 | 145.4506 | 116.055 | 1 | 0.145 | 55.023 | 13.547 | 0 | 0.896 | 335.9 | 155.76 | 1 |
| SE2 | 0.590618 | 150.6912 | 112.13 | 1 | 0.1549 | 57.259 | 13.0159 | 0 | 0.925 | 359.175 | 147.827 | 1 |
| SE3 | 0.589808 | 146.609 | 112.872 | 1 | 0.1599 | 55.454 | 13.3639 | 0 | 0.921 | 342.15 | 154.885 | 1 |
| DP | 0.63564 | 141.7031 | 111.992 | 1 | 0.1732 | 52.251 | 12.6824 | 0 | 1 | 316.2 | 149.366 | 1 |
| DC | 0.534 | 147.6351 | 110.123 | 1 | 0.1649 | 54.052 | 12.0403 | 0 | 1 | 329.1 | 159.873 | 1 |

**Tabla 7.5.** Comparación de políticas de selección, en alta carga.

En la Tabla 7.4 en baja carga, se puede observar que ningún grupo multicast fue rechazado, no obstante en la Tabla 7.5 en alta carga ya se tienen grupos multicast rechazados a causa de la congestión de la red. En la Tabla 7.6 se tiene un mayor número de grupos multicast rechazados.

| | Promedios | | | | Desviación Estándar | | | | Máximos | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\alpha_p$ | $C_p$ | $D_p$ | Na | $\alpha_d$ | $C_d$ | $D_d$ | Na | $\alpha_p$ | $C_p$ | $D_p$ | Na |
| $D_A\alpha$ | 0.845833 | 257.4525 | 101.106 | 22.675 | 0.1508 | 61.767 | 9.71905 | 5.51 | 1 | 404.025 | 122.681 | 25 |
| $\alpha D_A$ | 0.711123 | 259.7413 | 110.709 | 12.725 | 0.1612 | 67.304 | 10.0022 | 2.25 | 1 | 415.55 | 137.892 | 14 |
| $\alpha C$ | 0.695123 | 236.8052 | 118.493 | 12.2 | 0.1616 | 61.501 | 10.5977 | 1.39 | 1 | 385.925 | 148.206 | 13.3 |
| $C\alpha$ | 0.768533 | 223.2102 | 114.912 | 12.55 | 0.1621 | 57.635 | 9.76095 | 1.68 | 1 | 356.85 | 141.647 | 13.3 |
| SE1 | 0.718393 | 232.9407 | 116.884 | 12.225 | 0.1438 | 62.038 | 10.1965 | 1.25 | 1 | 378.675 | 143.613 | 13 |
| SE2 | 0.7442 | 235.6539 | 114.547 | 12.325 | 0.1378 | 61.827 | 9.61568 | 1.43 | 1 | 386.025 | 138.208 | 13 |
| SE3 | 0.74743 | 231.6186 | 115.209 | 12.15 | 0.1376 | 60.774 | 9.7227 | 1.07 | 1 | 372 | 139.094 | 12.8 |
| DP | 0.78899 | 225.3467 | 113.284 | 12.5 | 0.1456 | 58.059 | 9.0189 | 1.58 | 1 | 357.6 | 135.188 | 13 |
| DC | 0.74046 | 232.9115 | 107.792 | 13.2 | 0.1722 | 59.533 | 9.4612 | 126 | 1 | 377.5 | 133.918 | 14 |

**Tabla 7.6.** Comparación de políticas de selección, en saturación.

Si tomamos los máximos valores de los 3 objetivos considerados, y con estos normalizamos los resultados. Podemos hallar la política que obtiene el mejor valor de compromiso, esto es sencillamente la distancia al origen. Los resultados son los expuestos en la tabla 7.7.

| | Baja carga | Alta Carga | En saturación | Promedios |
|---|---|---|---|---|
| DC | 0.82780482 | 0.86048284 | 0.89402689 | 0.86077152 |
| $C\alpha$ | 0.84021785 | 0.88201801 | 0.91371454 | 0.87865013 |
| $\alpha C$ | 0.87485886 | 0.88831112 | 0.91406095 | 0.89241031 |
| DP | 0.8716257 | 0.90493881 | 0.91954577 | 0.89870343 |
| $\alpha D_A$ | 0.89114014 | 0.90545843 | 0.92734 | 0.90797952 |
| SE1 | 0.93161239 | 0.88190254 | 0.91261757 | 0.90871083 |
| SE3 | 0.9317856 | 0.89316087 | 0.91677449 | 0.91390699 |
| SE2 | 0.93132372 | 0.89951172 | 0.91862201 | 0.91648582 |
| $D_A\alpha$ | 0.93149692 | 0.94795141 | 0.95054948 | 0.9433326 |

**Tabla 7.7.** Comparación de políticas de selección, promedios de los escenarios.

Como podemos apreciar en la tabla 7.7, la política dinámica DC posee los mejores valores de compromiso en los tres escenarios probados. También podemos ver que en promedio optimizar primero alfa obtiene buenas soluciones de compromiso. Una observación importante, en los tres escenarios la política $C\alpha$ obtiene resultados muy próximos al origen de coordenados lo que implica buenos valores de la relación de compromiso. Optimizar $D_A\alpha$ produce los peores valores de compromiso en los tres escenarios de prueba. Así como en baja carga las políticas de $D_A\alpha$ , SE1, SE2 y SE3, obtuvieron los peores valores de compromiso.

Al final, sí podemos descartar las políticas estáticas y la semi-estática, por seleccionar soluciones de los extremos del frente Pareto, por lo que funcionan básicamente como un monobjetivo. Podríamos decir que las dinámicas DP y DC, no seleccionan las soluciones de los extremos del frente Pareto. La ventaja principal de la DC, es que no posee un parámetro a priori, y como resultado genera las mejores soluciones de compromiso respecto a la distancia al origen, o sea un buen equilibrio entre los objetivos, tanto para las comunicaciones individuales como para el funcionamiento de la red.

La DP, posee una menor cantidad de comunicaciones rechazadas. Con la política dinámica DP se requiere más tiempo de procesamiento en el peor de los casos que con la política dinámica DC, por la complejidad de elección de la DP.

En resumen, si no se usará DiffServ se recomienda usar la política dinámica DC, mientras que si se usa DiffServ se recomienda usar la política dinámica DP.

## 7.3 Conclusiones

En el presente trabajo se codificaron los algoritmos NSGA, NSGA2, cNSGA2, SPEA y SPEA2. Se realizaron comparaciones estáticas con distintos grados de dificultad. Como conclusión, se puede recomendar la utilización de algoritmos tipo SPEA en cualquiera de sus versiones (SPEA y SPEA2) para la resolución del problema de ingeniería de tráfico multicast, en un ambiente estático.

Se propusieron tres escenarios de prueba dinámicos baja carga, alta carga y en saturación. Se realizaron comparaciones de los MOEAs en ambientes dinámicos, donde podemos concluir que mientras no exista una diferencia muy significativa en las pruebas estáticas, tendremos desempeños similares de los algoritmos en pruebas dinámicas. Por lo tanto, podemos entender que son las políticas de selección de soluciones el punto más importante a tratar si tenemos algoritmos multiobjetivos de segunda generación y de desempeño parecido.

Se sugirieron 4 políticas de selección de soluciones, se realizaron comparaciones entre las políticas para cada uno de los escenarios dinámicos, donde se puede concluir que: si no se usará DiffServ se recomienda usar la política dinámica DC, mientras que si se usa DiffServ se recomienda usar la política dinámica DP.

## 7.4 Trabajos Futuros

Como trabajos futuros, se propone simulaciones con varias clases de quality of service. El estudio e implementación de nuevos esquemas de ingeniería de tráfico multicast multi-árboles, donde el flujo de datos de un grupo multicast es transmitido a los nodos destinos a través de varios árboles.

# Bibliografía

[ARA02] P. T. de Araujo, y G. M. Barbosa, “Multicast Routing with Quality of Service and Traffic Engineering Requirements in The Internet, Based On Genetic Algorithm”, *Proceedings of the 7th Brazilian Symposium on Neural Networks* (SBRN´02), Brasil, 2002.

[BAR00] B. Barán, y R. Sosa, “A New Approach for Antnet Routing, ”*IEEE International Conference on Computer Communication and Networks* (ICCCN’2000), USA, 2000.

[COE00] C. Coello, “EMOO Web Page, A complete list in alphabetical order”, *Evolutionary Optimization: an International Journal on Internet*, 2000. Disponible desde http://www.jeo.org/emo.

[COE96] C. Coello, “A Empirical Study of Evolutionary Techniques for multi-objective optimization in Engineering Design”, *Ph.D Thesis,* Department of Computer Science, Tulane University, New Orleans, USA, 1996.

[COH78] J. Cohon, *Multiobjective programming and planning*, Academic Press, 1978.

[CRI04a] J. Crichigno, y B. Barán, “Multiobjective Multicast Routing Algorithm”, *11th International Conference on Telecommunications*”, Brasil, 2004.

[CRI04b] J. Crichigno, y B. Barán, “A Multicast Routing Algorithm Using Multiobjective Optimization”, *11th International Conference on Telecommunications*”, Brasil, 2004.

[CRI04c] J. Crichigno, F. Talavera, J. Prieto y B. Barán, “Enrutamiento multicast utilizando optimización multiobjetivo,” *CACIC’2004*, Buenos Aires, Argentina, 2004. pp. 147-158.

[DAR85] C. Darwin, *On the Origin of Species by Means of Natural Selection*, 6° Edición, Penguin Classics, 1985.

[DEB99] K. Deb, “Evolutionary algorithms for multi-criterion optimization in engineering design”, *Proceedings of the Evolutionary Algorithms in Engineering and Computer Science (EUROGEN’99)*, 1999.

[DEB00] 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.

[DEB01] K. Deb, and T. Goel. “Controlled elitist non-dominated sorting genetic algorithm for better convergence,” http://www.lania.mx/~ccoello/EMOO/.

[DIJ59] E. Dijkstra, “A note on two problems in connexion with graphs”, *Numerische Mathematik*, Vol. 1, pág. 269-271, 1959.

[DON04] Y. Donoso, R. Fabregat, y J. Marzo, “Multi-objective Optimization Algorithm for Multicast Routing with Traffic Engineering”, *IEEE 3rd International Conference on Networking* (ICN´2004), Guadalupe – Caribe Francés, 2004.

[DON02] Y. Donoso. “Extensión de los Métodos Hop-by-Hop, CR-LDP y RSVP-TE para Multicast IP sobre MPLS”, *XXVIII Conferencia Latinoamericana de Informática*, Uruguay, 2002.

[FRA04] F. Talavera, J. Prieto, J. Crichigno y B. Barán. “Comparación de algoritmos evolutivos multi-objetivos en un ambiente multicast,” *CACIC’2004*, Buenos Aires, Argentina, 2004. pp. 1611-1622.

[GOL89] D. Goldberg, *Genetic Algorithm is Search, Optimization & Machine Learning*, Addison Wesley, 1989.

[HEI00] J. Heitkoetter, y B. Heitkoetter, “The Hitch-Hiker’s Guide to Evolutionary Computation: A List of Frequently Asked Questions (FAQ)”, USENET: comp.ai.genetic, disponible vía FTP desde tfm.mit.edu/pub/usenet/news.answ-ers/ai-faq/genetic, 2000.

[HOR97] J. Horn, “The nature of niching: genetic algorithms and the evolution of optimal, cooperative population”, *Ph.D Thesis*, University of Illinois at Urbain Champaign, USA, 1997.

[HOR86] E. Horowitz, y S. Sahni, *Fundamentos de Estructura de Dados*, Campus LTDA, 1986.

[HWA00] R. Hwang, W. Do, y S. Yang, “Multicast Routing Based on Genetic Algorithms”, *Journal of Information Science and Engineering*, Vol. 16, pág. 885-901, 2000.

[KOM93a] V. Kompella, “Multicast Routing Algorithm for Multimedia Traffic”, *Ph.D. Thesis,* University of California*,* San Diego, USA, 1993.

[KOM93b] V. Kompella, J. Pasquale, y G. Polyzos, “Multicast routing in multimedia communication”, *IEEE/ACM Transactions on Networking*, Vol. 1 N° 3, pág. 286-291, 1993.

[KOU81] L. Kou, G. Markowsky, y L. Berman, “A Fast Algorithm for Steiner Trees”, *Acta Informatica*, Vol. 15, N° 2, pág. 141-145, 1981.

[OBA98] S. Obayashi, S. Takahashi, y Y. Takeguchi, “Niching and elitist models for mogas”, *5$^{th}$ International Conference on Parallel Problem Solving from Nature (PPSN-V)*, Berlín, Alemania, pág. 260-269, 1998.

[OOM02] D. Ooms, B. Sales, W. Livens, A. Acharya, F. Griffoul, y F. Ansari, “Overview of IP Multicast in a Multi-Protocol Label”, *RFC 3353,* 2002.

[PAR98] G. Parks, y I. Miller, “Selective breeding in a multiobjective genetic algorithm”, *5th International Conference on Parallel Problem Solving from Nature (PPSN-V)*, Berlín, Alemania, pág. 250-259, 1998.

[RAV98] C. P. Ravikumar, y R. Bajpai, “Source-based delay bounded multicasting in multimedia networks”, *Computer Communications*, Vol. 21, pág. 126-132, 1998.

[ROS67] R. Rosemberg, “Simulation on Genetic Populations with Biochemical Properties”, *Ph.D Thesis*, University of Michigan, Ann Harbor, Michigan, USA, 1967.

[SEO02] Y. Seok, Y. Lee, Y. Choi, y C. Kim, “Explicit Multicast Routing Algorithm for Constrained Traffic Engineering”, *IEEE Proceedings of 7th International Symposium on Computer and Communications* (ISCC´02), Italia, 2002.

[SHA84] J. Shaffer, “Multiple Objective Optimization with Vector Evaluated Genetic Algorithms”, *Ph.D Thesis*, Vanderbilt University, USA, 1984.

[SRI94] N. Srinivas, y K. Deb, “Multiobjective Optimization Using Non-dominated Sorting Genetic Algorithm”, *MIT Evolutionary Computation*, Vol. 2, N° 3, pág. 221-248, 1994.

[STA00] W. Stallings, *Comunicaciones y Redes de Computadores*, Prentice Hall, 2000.

[STA01] W. Stallings. “MPLS”, *The Internet Protocol Journal*, Vol. 4, N° 3, 2001.

[STE86] R. Steuer, *Multiple criteria optimization: theory, computation and application*, New York, 1986.

[TAN03] A. Tanenbaum, *Computer Networks*, Prentice Hall, 2003.

[VAN99] D. Veldhuizen, “Multiobjective Evolutionary Algorithms: Classifications, Analysis, and New Innovations”, *Ph.D Thesis*, Graduated School of Engineering of the Air Force Institute of Technology, Air University, 1999.

[WAN01] Z. Wang, Internet QoS, *Architectures and Mechanism for Quality of Service*, Morgan Kaufmann, 2001.

[WOL97] D. Wolpert, y W. Macready, “No Free Lunch Theorems for Optimizations”, *IEEE Transactions on Evolutionary Computation*, Vol. 1, Nº 1, pág. 67-82, 1997.

[ZHE01] W. Zhengying, S. Bingxin, y Z. Erdun, “Bandwidth-delay-constraint least-cost multicast routing based on heuristic genetic algorithm”, *Computer Communications*, Vol. 24, pág. 685-692, 2001.

[ZHU03] Y. Zhu, W. Shu, y M. Wu, “Comparison study and evaluation of overlay multicast network”, *IEEE International Conference on Multimedia and Expo (ICME2003)*, Baltimore, USA, 2003.

[ZIT99] E. Zitzler, y L. Thiele, “Multiobjective Evolutionary Algorithms: A comparative Case Study and the Strength Pareto Approach”, *IEEE Trans. Evolutionary Computation*, Vol. 3, N° 4, pág. 257-271, 1999.

[ZIT01] E. Zitzler, M. Laumanns, y L. Thiele, “SPEA2. Improving the strength Pareto evolutionary algorithm for multi-objetive optimization,” http://www.lania.mx/~ccoello/EMOO/