OPTIMIZACIÓN MULTIOBJETIVO DE REDES EMPLEANDO ALGORITMOS EVOLUTIVOS PARALELOS

Susana Duarte Flores

PROYECTO FINAL DE TESIS

Orientador:
Prof. Benjamín Barán, D.Sc.

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

Asunción – Paraguay
Octubre del 2001

Este trabajo está dedicado a mis padres y a mi hermana. 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, en las personas de Marta Almirón, José Vallejos, Rodrigo Ramos, Diana Benitez, Fabián Laufer, Víctor Bogarín, Aldo Sotelo, Christian Von Lucken, María Elena García, Virna Cuquejo y Carlos Brizuela. 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.

Al profesor Alberto Paccanaro, por haberme inspirado el amor a la investigación desde los inicios de mi carrera, por su persistente confianza y soporte, también por su paciencia y sus ideas siempre acertadas e iluminadoras.

Al profesor Luca Cernuzzi, por su apoyo y ayuda en diferentes etapas y por haber colaborado en mi formación integral.

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.

Indice General

Lista de Acrónimos

BXC	*Branch Exchange Method*
CS	*Sistemas clasificadores*
CSA	*Cut-Saturation Algorithm*
E	*Error*
EA	*Algoritmo Evolutivo*
EP	*programación evolutiva*
ES	*estrategia de evolución*
FFGA	*algoritmo genético multiobjetivo de Fonseca y Fleming*
FS	*fitness sharing*
G	*distancia generacional*
GA	*algoritmo genético*
GP	*programación genética*
GVND	*generación de vectores no dominados*
GRVND	*generación real de vectores no dominados*
ISP	*proveedor de servicios de Internet*
LAN	*red de área local*
MCS	*simulación Monte Carlo*
MFLOPS	*millones de instrucciones en punto flotante por segundo*
MOEA	*algoritmo evolutivo multiobjetivo*
MOP	*problema de optimización multiobjetivo*
NSGA	*nondominated sorting genetic algorithm de Srinivas y Deb*
PVM	*máquina paralela virtual*
RGVND	*razón de generación de vectores no dominados*
SA	*simulated annealing*
SPEA	*strength Pareto evolutionary algorithm de Zitsler y Thiele*
SOP	*problema de optimización de un solo objetivo*
TS	*tabú search*
VEGA	*vector evaluated genetic algorithm de Shaffer*

Lista de Símbolos Específicos Utilizados

σ_{share}	*radio de nichos para FS*
$cn_{\boldsymbol{i}}$	*contador de nicho para el individuo* ***i***
cost(t, l)	*costo de colocación, por unidad de distancia (i.e. km), de la tecnología de comunicación t, como enlace l*
d(i, j)	*distancia entre el individuo* ***i*** *y el individuo* ***j***
$dist\_unit_l$	*distancia en unidades de distancia (km) cubierta por el enlace l*
e	*vector de restricciones de un SOP o MOP*
E	*esquema*
ε_{niv}	*diferencia entre fitness real y virtual de niveles (NSGA)*
f	*función objetivo de un SOP*
f	*vector de funciones objetivo de un MOP*
F(i)	*fitness asignado al individuo* $\boldsymbol{i} \in \boldsymbol{I}$
F'(i)	*fitness compartido asignado al individuo* $\boldsymbol{i} \in \boldsymbol{I}$
g_{max}	*Cantidad máxima de generaciones para el GA*
H	cantidad de procesadores utilizados en la paralelización
h	*uno de los procesadores en el proceso de paralelización*
I	*espacio de individuos para el GA*
i	*individuo que pertenece al espacio* ***I***
$\boldsymbol{I_{niv}}$	*grupo de individuos en nivel de dominancia niv (NSGA)*
k	*cantidad de funciones objetivo de un MOP*
L	*conjunto de enlaces (entre pares de nodos) de la red a optimizar*
l	*un enlace de la red a optimizar*
m	*cantidad de restricciones de un SOP o MOP*
m%	*porcentaje de la población sometida a mutación en cada generación*
n	*cantidad de variables de decisión de un SOP o MOP*
N	*tamaño de población*
N'	*tamaño máximo del conjunto externo de individuos no dominados*
P	*población*

P'	*conjunto externo de individuos no dominados*
p_c	*probabilidad de cruzamiento*
p_m	*probabilidad de mutación*
$r(\mathbf{i})$	*rango del individuo* $\mathbf{i}$
r_m	*razón de mutación*
T	*cantidad de tecnologías de comunicación disponibles para interconectar dos nodos de la red a optimizar*
t	*tecnología de comunicación (fibras ópticas, radio enlaces, microondas, cables coaxiales, etc.)*
V	*conjunto de nodos (centros de comunicación a interconectar) en la red a optimizar*
$\boldsymbol{x}$	*vector de decisión de un MOP*
X	***espacio de decisión***
$\boldsymbol{X}_f$	*conjunto de vectores de decisión factibles*
$\boldsymbol{X}_{true}$	*conjunto de vectores de decisión Pareto-óptimos*
$\boldsymbol{y}$	*vector de funciones objetivo de un MOP*
$\boldsymbol{Y}$	*espacio objetivo*
$\boldsymbol{Y}_f$	*conjunto de vectores objetivo obtenidos por la aplicación de la función objetivo a* $\boldsymbol{X}_f$
$\boldsymbol{Y}_{true}$	*conjunto de vectores objetivo obtenidos por la aplicación de la función objetivo a* $\boldsymbol{X}_p$
$\wedge$	*Símbolo del operador "and" lógico*
$\vee$	*Símbolo del operador "or" lógico*
!	*Símbolo del operador "not" lógico*
$\succ$	*Símbolo para indicar dominancia entre soluciones*
$\sim$	*Símbolo para indicar que dos soluciones de un MOP son indiferentes o no comparables*

1 Optimización con objetivos múltiples

1.1 Introducción

Gran parte de los problemas del mundo real implican la optimización simultánea de varios objetivos que generalmente presentan conflictos entre ellos; es decir, la mejora en uno conduce a un deterioro en el otro. La presencia de tales tipos de problemas es tan significativa, que consume gran parte de nuestro tiempo cotidiano de decisión. Se trata, por ejemplo, de escoger el medio ideal para llegar al trabajo, establecer el orden de nuestras tareas, elegir el restaurante para el almuerzo, hacer las compras en el supermercado, preparar la cena y la distribución de actividades en el tiempo de ocio restante. También es el mismo tipo de problemas que enfrentan los ingenieros y técnicos a la hora de diseñar e implementar sistemas de todo tipo: existen múltiples objetivos a cumplir y se espera lograrlos todos en la medida de lo posible.

Aunque la mayoría de los problemas de decisión involucran este tipo de situaciones, las propuestas computacionales de automatización que se han presentado para resolverlos habitualmente 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 se prioriza uno de los objetivos y los demás se colocan como restricciones, o también se genera un objetivo compuesto otorgando pesos a los objetivos en juego y armando una suma ponderada de los mismos. De todos modos, ninguna de estas reducciones refleja fielmente al problema y, por tanto, tampoco otorga soluciones completamente satisfactorias.

Se pone como ejemplo el problema de la compra de un automóvil. El comprador desea un automóvil óptimo y por tanto no puede preocuparse solamente en minimizar el precio del auto, ya que también le interesan otros factores. Si se tratase de un problema de objetivo único, se conformaría con llamar a todos los distribuidores y hacer una lista de precios (sin considerar marcas, modelos, tamaño, confort, etc.) y escogería el automóvil de menor precio sin mayores complicaciones. Sin dudas, el modelo matemático que está empleando en su búsqueda ¡no es el más apropiado!. Por ende, los resultados tampoco son satisfactorios, y por ahorrarse unos pesos inicialmente, el comprador acaba gastando todos sus ahorros en combustible y costos de mantenimiento.

Sin embargo, el estado actual de la ciencia podría generar mejores resultados ya que existen modelos matemáticos que se ajustan mejor a la naturaleza de éstos problemas. Tales

modelos provienen de un área de la Investigación de Operaciones conocida como optimización con objetivos múltiples o multiobjetivo.

En los problemas de optimización de un solo objetivo (*SOP*s, del inglés *Single Objective Problem*) el resultado óptimo deseado está claramente definido. Partiendo del ejemplo anterior el objetivo sería minimizar precio del automóvil, y el resultado sería el automóvil con menor precio. Sin embargo, esta condición no se cumple para los problemas de optimización multiobjetivo (*MOPs,* por sus siglas en inglés: *Multiobjective Optimization Problem*) donde, en vez de un único óptimo, contamos con todo un conjunto de soluciones de compromiso.

Para evidenciar este hecho se vuelve al ejemplo del problema de la compra de un automóvil. El comprador tiene varios objetivos que desearía alcanzar, pero también múltiples restricciones. En cuanto a objetivos podríamos mencionar: minimizar el costo del automóvil, la cantidad de combustible consumida en una distancia dada, los gastos de mantenimiento implicados, etc. Además, deseará maximizar el confort, el espacio, la confiabilidad, la seguridad, tiempo transcurrido entre mantenimientos, costos de reventa, etc. Cuando se plantean las restricciones descubrimos que este comprador cuenta con un presupuesto limitado, desea un vehículo fabricado en la región y confía más en algunas marcas que en otras. Los conflictos que surgen entre los objetivos mencionados son obvios e inmediatamente –considerando la propia experiencia– surgen posibles soluciones. Así tenemos el automóvil que tiene el menor precio, pero está bastante alejado de los objetivos relacionados al confort, la seguridad y la confiabilidad. También tenemos el de costo superior a los demás pero con óptimas características, aunque amplio consumo de combustible. Así podemos citar numerosos ejemplos. Entre éstos se encuentran los automóviles promedio, que cumplen con las restricciones dadas y que implican una solución de compromiso entre todos los factores en juego. Entre ellos no se puede decir que alguno sea mejor, ya que al alterar un factor para mejorarlo, estamos empeorando otro. Así, en un problema de optimización multiobjetivo cotidiano, resulta evidente la existencia de múltiples soluciones y la imposibilidad de decidir cuál de ellas es mejor si se consideran todos los objetivos al mismo tiempo. En el caso del comprador, para realizar su elección deberá necesariamente contar con algún criterio, posiblemente de índole subjetiva, que le permita optar por una u otra alternativa.

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.

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 [PAR96] para los problemas en que existen múltiples objetivos a cumplir, y persisten conflictos al realizar la optimización simultánea de los

mismos. A partir de este concepto se establece, como requisito para afirmar que una situación es mejor que otra, el que en ella no se disminuya a nadie, pero se mejore a alguno; es decir que una situación será mejor que otra sólo si en la nueva es posible compensar las pérdidas de todos los perjudicados... y aún queda un sobrante. En todo otro caso, según Pareto [PAR96], para decidir se requiere un juicio de valor y la ciencia no puede guiarnos.

Introducido el concepto de optimalidad Pareto, a continuación, en la sección 1.2 se presenta formalmente las definiciones básicas de la optimización multiobjetivo. En la sección 1.3 se discuten las fases de resolución de un *MOP* y la forma en que ellas se entrelazan. La sección 1.4 está dedicada a las técnicas tradicionales para resolución de *MOPs*, discutiéndolas brevemente e indicando sus desventajas potenciales. Posteriormente, en la sección 1.5 se propone a los algoritmos evolutivos como herramientas que poseen cualidades deseables para la resolución de *MOPs*. Luego se agrega una cronología breve de los acontecimientos más importantes en el desarrollo de la optimización multiobjetivo utilizando algoritmos evolutivos, enfocando además las áreas donde todavía existen cuestiones abiertas. En la sección 1.6 damos una breve reseña del enfoque que seguirá el resto del libro.

1.2 Conceptos básicos y terminología

Previamente a la introducción del problema a tratar, se presenta una descripción formal de conceptos y terminología, de modo a facilitar las discusiones posteriores. Cabe mencionar que 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 de definir con claridad estos aspectos. Sin embargo, en gran parte de los trabajos consultados se percibe aún bastante confusión al respecto, por tanto es esencial establecer una notación clara antes de iniciar la discusión.

A partir de los conceptos introducidos en la sección anterior se define a un *MOP* de la siguiente manera:

Definición 1: Problema de Optimización Multiobjetivo (*Multiobjective Optimization Problem: MOP*). Un *MOP* general incluye un conjunto de *n* parámetros (variables de decisión), un conjunto de *k* funciones objetivo, y un conjunto de *m* restricciones. Las funciones objetivo y las restricciones son funciones de las variables de decisión. Luego, el *MOP* puede expresarse como:

$$
\begin{aligned}
&\textit{Optimizar} && \boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots , f_k(\boldsymbol{x})) \\
&\textit{sujeto a} && \boldsymbol{e}(\boldsymbol{x}) = (e_1(\boldsymbol{x}), e_2(\boldsymbol{x}), \ldots , e_m(\boldsymbol{x})) \geq \boldsymbol{0} \\
&\textit{donde} && \boldsymbol{x} = (x_1, x_2, \ldots , x_n) \in \boldsymbol{X} \\
& && \boldsymbol{y} = (y_1, y_2, \ldots , y_k) \in \boldsymbol{Y}
\end{aligned}
\qquad \textbf{(1.1)}
$$

siendo $\boldsymbol{x}$ el vector de decisión e $\boldsymbol{y}$ el vector objetivo. El espacio de decisión se denota por $\boldsymbol{X}$, y al espacio objetivo por $\boldsymbol{Y}$. Optimizar, dependiendo del problema, puede significar igualmente, minimizar o maximizar.

El conjunto de restricciones $\boldsymbol{e}(\boldsymbol{x}) \geq \boldsymbol{0}$ 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 los requerimientos $\boldsymbol{e}(\boldsymbol{x})$:

$$\boldsymbol{X}_f = \{\boldsymbol{x} \in \boldsymbol{X} \mid \boldsymbol{e}(\boldsymbol{x}) \geq \boldsymbol{0}\} \qquad \textbf{(1.2)}$$

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

$$\boldsymbol{Y}_f = \boldsymbol{f}(\boldsymbol{X}_f) = \bigcup_{x \in X_f} \{\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})\} \qquad \textbf{(1.3)}$$

De estas definiciones se tiene que cada solución del *MOP* en cuestión consiste de una *n*-tupla $\boldsymbol{x} = (x_1, x_2, \ldots , x_n)$, 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, y según ya se ha introducido, no existe un único "mejor valor", sino un conjunto de soluciones. Entre éstas, ninguna se puede considerar mejor a las demás si se tienen en cuenta todos los objetivos al mismo tiempo. Este hecho deriva de que puede existir –y generalmente existe– conflicto 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 $\boldsymbol{a}, \boldsymbol{b} \in \boldsymbol{X}_f$, se cumple una sola de las siguientes proposiciones: $f(\boldsymbol{a}) > f(\boldsymbol{b})$, $f(\boldsymbol{a}) = f(\boldsymbol{b})$ o $f(\boldsymbol{b}) > f(\boldsymbol{a})$. 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 suele ser parcial (i.e. existen vectores de decisión $\boldsymbol{a}$ y $\boldsymbol{b}$ con los que $\boldsymbol{f}(\boldsymbol{a})$ no puede considerarse mejor que $\boldsymbol{f}(\boldsymbol{b})$ y tampoco $\boldsymbol{f}(\boldsymbol{b})$ puede considerarse mejor que $\boldsymbol{f}(\boldsymbol{a})$).

Para ilustrar este concepto se presenta el siguiente gráfico que muestra la relación entre dos funciones f_1 y f_2. La función f_1 representa el costo de los dispositivos de seguridad de un sistema, mientras que f_2 representa el índice de ocurrencia de accidentes.

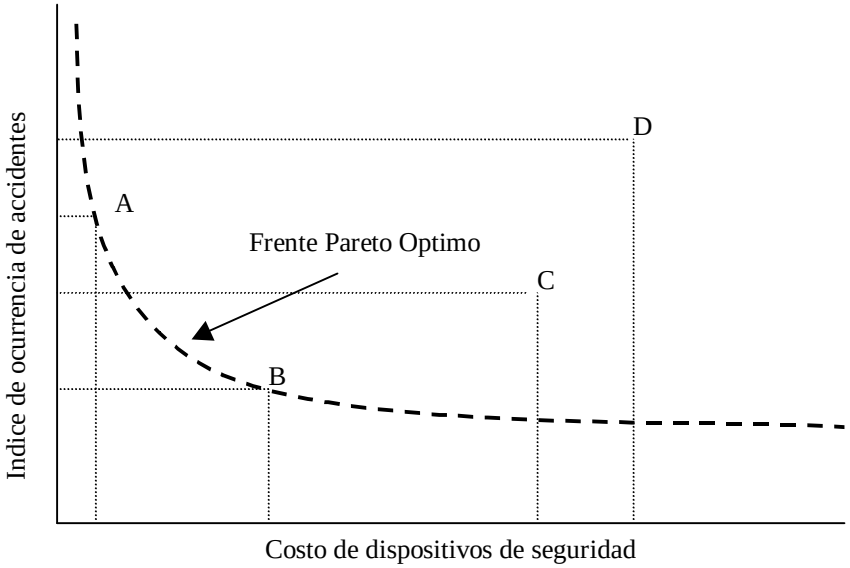


Fig. 1.1 Ejemplo de la optimalidad Pareto en el espacio objetivo.

La solución representada por el punto B es mejor que la representada por el punto C, debido a que provee de una mayor seguridad a un costo menor. La solución B sería también la escogida en el caso de estar realizando la optimización de un solo objetivo. Más aún, todas las soluciones que se encuentran en el rectángulo delimitado por el origen de coordenadas y la solución C y por encima de la curva, son mejores a C. En la comparación entre C y A, se obtiene que, disminuyendo mucho los costos, el nivel de seguridad provisto por A es solo ligeramente peor, aunque C y A son no comparables entre ellos porque no se podría argumentar que uno es mejor que otro al considerar todos los objetivos. Si comparamos a A con B tampoco podríamos establecer que alguna de las dos sea mejor, si se considera que ambos objetivos son igualmente importantes y no se introduce alguna consideración de índole subjetiva. Sin embargo, B es claramente superior a C en ambos objetivos. Para expresar esta situación matemáticamente, las relaciones =, $\leq$ y $\geq$ se deben extender. Esto se puede realizar de la siguiente manera:

Definición 3: Dados 2 vectores de decisión $\boldsymbol{u} \in \boldsymbol{X}$ y $\boldsymbol{v} \in \boldsymbol{X}$,

$$\begin{aligned} &\boldsymbol{f}(\boldsymbol{u}) = \boldsymbol{f}(\boldsymbol{v}) \text{ si y sólo si } \forall i \in \{1, 2, \ldots, k\}: f_i(\boldsymbol{u}) = f_i(\boldsymbol{v}) \\ &\boldsymbol{f}(\boldsymbol{u}) \geq \boldsymbol{f}(\boldsymbol{v}) \text{ si y sólo si } \forall i \in \{1, 2, \ldots, k\}: f_i(\boldsymbol{u}) \geq f_i(\boldsymbol{v}) \\ &\boldsymbol{f}(\boldsymbol{u}) > \boldsymbol{f}(\boldsymbol{v}) \text{ si y sólo si } \boldsymbol{f}(\boldsymbol{u}) \geq \boldsymbol{f}(\boldsymbol{v}) \wedge \boldsymbol{f}(\boldsymbol{u}) \neq \boldsymbol{f}(\boldsymbol{v}) \end{aligned} \quad \textbf{(1.4)}$$

Las relaciones $\leq$ y $<$ se definen de manera similar.

A partir de esta noción, se sigue que $\boldsymbol{f}(B) < \boldsymbol{f}(C)$, $\boldsymbol{f}(C) < \boldsymbol{f}(D)$, y como consecuencia $\boldsymbol{f}(B) < \boldsymbol{f}(D)$. Sin embargo, al comparar *A* y *C* o *A* y *B*, no se puede decir que alguna sea superior a la otra. Por ejemplo, a pesar de que la solución representada por *B* es más cara, provee menor índice de accidentes que la representada por *A*.

Por lo expuesto, se tiene que dos vectores de decisión $\boldsymbol{x_1}$ y $\boldsymbol{x_2}$ de un *MOP* pueden cumplir solo una de tres condiciones posibles: $\boldsymbol{f}(\boldsymbol{x_1}) > \boldsymbol{f}(\boldsymbol{x_2})$, $\boldsymbol{f}(\boldsymbol{x_2}) > \boldsymbol{f}(\boldsymbol{x_1})$ o $\boldsymbol{f}(\boldsymbol{x_1}) \not\succcurlyeq \boldsymbol{f}(\boldsymbol{x_2}) \wedge \boldsymbol{f}(\boldsymbol{x_2}) \not\succcurlyeq \boldsymbol{f}(\boldsymbol{x_1})$. Esta situación se expresa con los siguientes símbolos y términos:

Definición 4: Dominancia Pareto en un contexto de Maximización. Para dos vectores objetivo $\boldsymbol{a}$ y $\boldsymbol{b}$,

$$\begin{array}{lll} \boldsymbol{a} \succ \boldsymbol{b} \text{ (}\boldsymbol{a}\text{ domina a }\boldsymbol{b}\text{)} & \text{si y solo si} & \boldsymbol{a} > \boldsymbol{b} \\ \boldsymbol{b} \succ \boldsymbol{a} \text{ (}\boldsymbol{b}\text{ domina a }\boldsymbol{a}\text{)} & \text{si y solo si} & \boldsymbol{b} > \boldsymbol{a} \\ \boldsymbol{a} \sim \boldsymbol{b} \text{ (}\boldsymbol{a}\text{ y }\boldsymbol{b}\text{ no son comparables)} & \text{si y solo si} & \boldsymbol{a} \not\succcurlyeq \boldsymbol{b} \wedge \boldsymbol{b} \not\succcurlyeq \boldsymbol{a} \end{array} \quad \textbf{(1.5)}$$

Definición 5: Dominancia Pareto en un contexto de Minimización. Para dos vectores objetivo $\boldsymbol{a}$ y $\boldsymbol{b}$,

$$\begin{array}{lll} \boldsymbol{a} \succ \boldsymbol{b} \text{ (}\boldsymbol{a}\text{ domina a }\boldsymbol{b}\text{)} & \text{si y solo si} & \boldsymbol{a} < \boldsymbol{b} \\ \boldsymbol{b} \succ \boldsymbol{a} \text{ (}\boldsymbol{b}\text{ domina a }\boldsymbol{a}\text{)} & \text{si y solo si} & \boldsymbol{b} < \boldsymbol{a} \\ \boldsymbol{a} \sim \boldsymbol{b} \text{ (}\boldsymbol{a}\text{ y }\boldsymbol{b}\text{ no son comparables)} & \text{si y solo si} & \boldsymbol{a} \not\preccurlyeq \boldsymbol{b} \wedge \boldsymbol{b} \not\preccurlyeq \boldsymbol{a} \end{array} \quad \textbf{(1.6)}$$

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.

Se puede introducir el criterio de optimalidad Pareto, a partir del concepto de dominancia Pareto. Volviendo al gráfico, el punto A es único entre los demás: su vector de decisión $\boldsymbol{a}$ no está dominado por otro vector de decisión. Esto implica que $\boldsymbol{a}$ es óptimo en el sentido de que no puede mejorarse en un objetivo, sin degradar a otro. Tales soluciones se conocen como Pareto óptimas (también se utiliza el término “no inferior”).

Definición 6: Optimalidad Pareto. Dado un vector de decisión $\boldsymbol{x} \in \boldsymbol{X}_f$ y su correspondiente vector objetivo $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) \in \boldsymbol{Y}_f$, se dice que $\boldsymbol{x}$ es no dominado respecto a un conjunto $\boldsymbol{A} \subseteq \boldsymbol{X}_f$ si y solo si

$$\forall\ \boldsymbol{a} \in \boldsymbol{A} : (\boldsymbol{x} \succ \boldsymbol{a} \vee \boldsymbol{x} \sim \boldsymbol{a}) \quad \textbf{(1.7)}$$

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** ($\boldsymbol{x} \in \boldsymbol{X}_{true}$ –el conjunto Pareto óptimo real–). Mientras que la $\boldsymbol{y}$ correspondiente es parte del **frente Pareto óptimo real** $\boldsymbol{Y}_{true}$. Esto se define a continuación.

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

$$\boldsymbol{X}_{true} = \{ \boldsymbol{x} \in \boldsymbol{X}_f \mid \boldsymbol{x} \text{ es no dominado con respecto a } \boldsymbol{X}_f \} \quad \textbf{(1.8)}$$

El conjunto $\boldsymbol{X}_{true}$ también es conocido como el conjunto Pareto óptimo. Mientras que el conjunto correspondiente de vectores objetivo $\boldsymbol{Y}_{true} = \boldsymbol{f}(\boldsymbol{X}_{true})$ constituye el frente Pareto óptimo.

Retomando el ejemplo de la figura 1.1, los puntos de la curva (i.e. A, B) representan soluciones Pareto óptimas. Entre ellas son no comparables, o, en otras palabras, son indiferentes. Esto aclara aún más la diferencia de los *MOPs* con los *SOPs*, o problema de optimización de un único objetivo): en este caso no hay una única solución sino un conjunto de soluciones de compromiso. Ninguna de ellas se puede definir como "mejor" que las demás, a menos que se incluya alguna otra información que determine que un objetivo es más importante que los demás o que se fije un peso relativo entre los objetivos.

Como recapitulación y como una ilustración adicional de los conceptos presentados, se discutirá a continuación un *MOP* específico, usado anteriormente para idénticos propósitos en [SHA84], con una sola variable de decisión (n = 1), dos funciones objetivo (k = 2) y sin restricciones (m = 0). Este *MOP* se define como:

$$\begin{aligned} &\text{Minimizar:} && \boldsymbol{f}(x) = (f_1(x), f_2(x)), \\ &\text{donde} && f_1(x) = x^2, \\ & && f_2(x) = (x - 2)^2 \end{aligned} \quad \textbf{(1.9)}$$

Este par de funciones ha sido estudiado previamente no solo en [SHA84], sino también en [DEB89, COE96, BEN97, DEB98, FON95, HOR94] y en muchos otros trabajos, constituyéndose en un paradigma para la presentación de los principales conceptos de un *MOP*. En efecto, según los estudios bibliográficos realizados en [VAN99] la optimización de este par es la función objetivo clásica esogida por gran parte de los autores que proponen un algoritmo evolutivo multiobjetivo y que desean probar su eficiencia. Por este motivo ha sido seleccionado para su estudio preliminar.

La figura 1.2 presenta la gráfica de ambas funciones respecto a *x*. La misma sugiere que el conjunto Pareto óptimo es { $x \mid x \geq 0$ y $x \leq 2$ }. Esto es: $\boldsymbol{X}_{true} = \{x \in R \mid 0 \leq x \leq 2\}$. La solución $x = 0$ es óptima respecto a f_1 pero no respecto a f_2. Cualquier solución que no se encuentre en el rango [0, 2] no es miembro del conjunto Pareto óptimo, puesto en dicho rango ($\boldsymbol{X}_{true}$) existe siempre una solución que la domina.

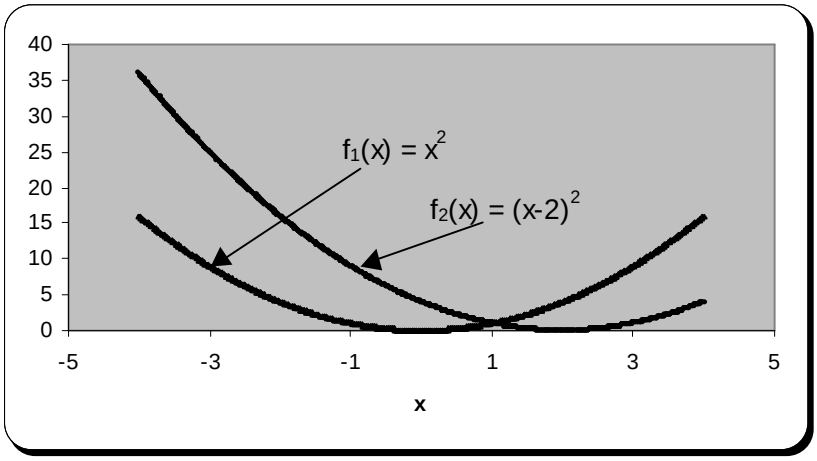


Fig. 1.2. Valores de f_1 y f_2 en función de $x \in X_f = (-\infty, \infty)$

La diferencia esencial entre la figura 1.2 y la figura 1.3 es que la primera representa los valores de las funciones f_1 y f_2 para diferentes valores de la variable independiente *x*. Mientras que la segunda representa los valores de la función f_1 graficados contra la función f_2, para el mismo valor de la variable independiente. Es decir, la figura 1.3 es una gráfica en el espacio objetivo ***Y***, que muestra los vectores del *MOP* como puntos. Los vectores no dominados (que se muestran como puntos), representan el frente Pareto del *MOP*, por encima de la curva, se encuentra el espacio objetivo factible $\boldsymbol{Y}_f$.

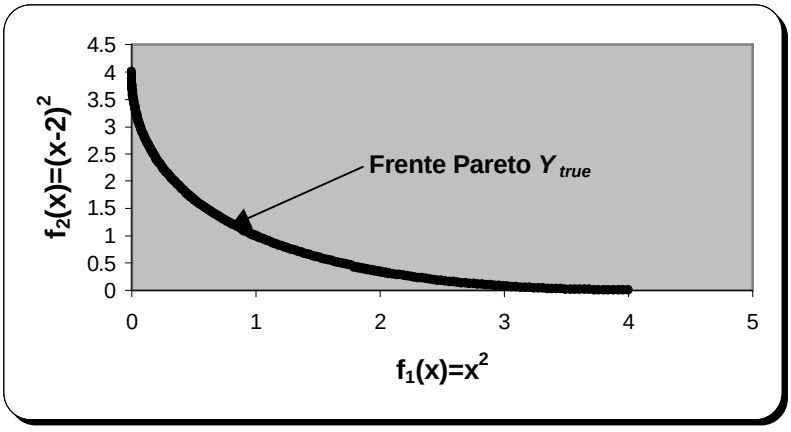


Fig. 1.3. Frente Pareto del *MOP*.

1.3 Búsqueda y Toma de Decisiones

Al resolver un *MOP*, se pueden distinguir 2 niveles de dificultad en el problema: 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 de soluciones de compromiso que se utilizarán, del conjunto Pareto óptimo ya definido. Este paso, generalmente, lo realiza un ser humano y tiene que ver con cuestiones inherentes a las condiciones del problema y los recursos disponibles.

Dependiendo de la manera en que se combinan ambas fases (búsqueda y toma de decisiones) los métodos de optimización multiobjetivo pueden clasificarse en tres categorías diferentes [COH78, HWA79]:

a) **Métodos de 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) **Métodos de 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) **Métodos de toma de decisión durante la búsqueda** (buscar mientras se decide): éstos permiten 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*.

La combinación de objetivos múltiples 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. Sin embargo, tal combinación requiere conocimiento del dominio del problema que no siempre está disponible. Por ejemplo, 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.

En particular, en este trabajo nos hemos enfocado en técnicas que han demostrado ser capaces de:

a) explorar espacios de búsquedas prácticamente ilimitados y altamente complejos, y
b) generar aproximaciones lo más fieles posibles al conjunto Pareto óptimo.

Además, hemos agregado una característica más: la posibilidad de obtener resultados en un periodo de tiempo razonable. Esto lo logramos mediante el paralelismo.

Todas estas características conforman el primer paso hacia la toma de decisiones durante el proceso de búsqueda (interactividad) y conforman la base de la investigación posterior en el área de resolución de *MOPs*.

1.4 Métodos tradicionales para la resolución de *MOPs*

Numerosos métodos clásicos existen para el tratamiento de problemas de optimización multiobjetivo. Todos ellos se caracterizan por operar en dos fases. Primeramente generan, a partir del problema de objetivos múltiples, un problema de optimización de un solo objetivo (*SOP*). En la segunda fase aplican un método de optimización tradicional al *SOP* generado en la fase anterior y obtienen sus resultados. Ambas fases son independientes entre sí, i.e. una vez obtenido el *SOP* se puede resolverlo con cualquier técnica típica (*hill climbing*, algoritmos de búsqueda aleatoria, métodos numéricos, etc.).

Varios tipos de métodos se emplean para la reducción realizada en la primera fase, cada uno de ellos tiene sus propias ventajas y desventajas. En gran parte de ellos, la técnica consiste en disponer de las funciones objetivo de modo tal que se puedan unir en una única función parametrizada, mientras el proceso de optimización se aplica sobre esta única función. Generalmente, los parámetros de la función se varían de modo sistemático y se requieren múltiples corridas del procedimiento de optimización (2da. fase) para conseguir un conjunto que se aproxime al conjunto Pareto óptimo real.

Algunas técnicas representativas son “Método de la suma con pesos” [COH78], “Método de las restricciones” [COH78], “Programación en base a objetivos [STE86], y la “Aproximación

Min Max" [STE86]. A continuación se presentarán, a modo de ejemplo, dos de los métodos más conocidos y de uso más extendido, exponiendo sus características más resaltantes. No discutiremos sobre los métodos empleados en la segunda fase, ya que son de conocimiento ampliamente difundido (búsqueda exhaustiva, programación lineal, optimización utilizando el concepto del gradiente, técnicas de inteligencia artificial, etc.) y al respecto existe ya una vasta bibliografía autorizada.

1.4.1 Método de la suma con pesos

En este método se combinan las diferentes funciones objetivo en una sola función *F*, de la siguiente manera:

$$\text{Optimizar} \qquad F = \sum_{j=1}^{k} w_j f_j(\boldsymbol{x}) \tag{1.10}$$

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 que se cumpla $\sum_{j=1}^{k} w_j = 1,\ w_j > 0$.

El procedimiento es sencillo: se escoge una combinación de pesos (posiblemente aleatoria) 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.

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 [DEB99b] 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 [DEB99b].

1.4.2 Método de la perturbación ε

Para intentar remediar la dificultad arriba mencionada, se utiliza otra técnica. La misma consiste en construir un problema de optimización en el que todos los objetivos, excepto uno, se usan como restricciones (*k*-1 restricciones), mientras el sobrante, que puede escogerse aleatoriamente, se usa como función objetivo del *SOP* resultante. Este método es el que se ha escogido, hasta ahora, para el tratamiento del problema de optimización de topologías de redes [LAU00, DEE97, DEN97, DEE98]. La forma general de expresión del problema es la siguiente:

$$\begin{aligned} &\text{Maximizar} && f_h(\boldsymbol{x}) \\ &\text{Sujeto a} && f_j(\boldsymbol{x}) \le \varepsilon_j ; \quad 1 \le j \le k \qquad j \ne h \\ & && \boldsymbol{x} \in \boldsymbol{X_f} \end{aligned} \tag{1.11}$$

Para encontrar una solución Pareto óptima, se escoge un valor adecuado para ε_j de la *j*-ésima función objetivo ($j \ne h$). Luego se resuelve el problema de optimización con objetivo único. El procedimiento se repite, con diferentes valores de ε_j para hallar nuevas soluciones pertenecientes al frente Pareto óptimo. La dificultad principal que presenta este método reside en la necesidad de un conocimiento del rango apropiado de valores a asignar a ε_j para las (*k*-1) funciones objetivo, lo que no resulta fácil en la práctica.

1.4.3 Discusión de los métodos clásicos

Aunque no se adecuan a la naturaleza del problema, la ventaja principal de los métodos tradicionales de primera fase que se mencionan en las secciones previas es que permite la utilización –en la segunda fase– de algoritmos de optimización para *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, las secciones precedentes han mostrado que éstas técnicas no están exentas de dificultades que limitan su aplicación a la optimización de objetivos múltiples.

Todos los métodos aludidos, y aún otros propuestos en un contexto tradicional, cuentan con al menos una de las siguientes dificultades [DEB99a, DEB99b]:

1) El algoritmo de optimización se debe aplicar varias veces para encontrar las soluciones Pareto óptimas múltiples. 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.

2) La mayoría de los algoritmos requieren conocimiento previo del problema a resolver y son sensibles a los pesos o niveles de demanda utilizados.

3) Algunos algoritmos son sensibles a la forma del frente Pareto (problemas con curvas cóncavas).

4) 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.

5) Relacionado al punto anterior, tenemos que en problemas que involucren al azar o incertezas, los métodos clásicos no son necesariamente confiables.

6) Como los optimizadores de un solo objetivo no son eficientes en búsquedas de universos discretos [DEB99b], tampoco serán eficientes para optimizaciones multiobjetivo en espacios discretos.

Investigaciones recientes [DEB99a, DEB99b] han demostrado que todas las dificultades arriba mencionadas pueden ser superadas con la utilización de un algoritmo evolutivo. Las características de éstos algoritmos hacen posible que:

a) se manejen espacios de búsqueda de casi cualquier tamaño y características; y

b) debido a su paralelismo inherente, se puedan generar múltiples soluciones en una sola corrida, permitiendo un efecto sinergético.

Con la afirmación previa, se introduce el criterio que se utilizará en el presente trabajo para lidiar con el problema multiobjetivo, es decir: se empleará algoritmos evolutivos. Es adecuado, entonces, dedicar la siguiente sección a la presentación de estos algoritmos.

1.5 Algoritmos Evolutivos en Optimización Multiobjetivo

El término algoritmo evolutivo (*EA*[1]. por sus siglas en inglés: *Evolutionary Algorithm*) 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.

En la naturaleza los individuos se caracterizan 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. Cada elemento que conforma el cromosoma recibe el nombre de allele. Cada individuo posee un nivel del adaptación al medio que lo dota de mayor capacidad de sobrevivencia y generación de descendencia. Tal nivel de adaptación está ligado a las características que están codificadas en sus cromosomas. Como el material genético puede

[1] Para una introducción más profunda y detallada a los *EAs*, referimos al lector a [GOL89].

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 tanto si padres con buenas características se cruzan, posiblemente generarán hijos igualmente buenos o incluso mejores.

Para resolver un problema de búsqueda u optimización utilizando algoritmos evolutivos y los conceptos sugeridos, primero se representa como individuos de una población finita a un número dado de soluciones posibles del problema, a este proceso se denomina codificación. En la codificación de un individuo debe estar presente toda la información relevante al mismo y que se considera influye en la optimización o búsqueda. Generalmente la codificación de un individuo o su cromosoma es una cadena de bits o de números enteros, dependiendo del problema que se desea resolver. En dicha cadena, el elemento que se encuentra en una posición dada recibe el nombre de allele.

Luego, se determina el nivel de aptitud o adaptación de cada individuo (*fitness*), dependiendo de la calidad de la solución que representa. Posteriormente los individuos existentes generan a otros individuos mediante los 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 *fitness* o aptitud: a mayor *fitness*, 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 características buenas 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 alto *fitness* y la alteración se transmitirá a los nuevos individuos; si el cambio no es benéfico, el individuo alterado tendrá un *fitness* bajo 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 genéticamente muy parecidos y que tienen un nivel de aptitud elevado, es decir, representan buenas soluciones al problema propuesto.

Los operadores descritos reciben el nombre de operadores de búsqueda u operadores genéticos. La reproducción enfoca la atención en los individuos con alto *fitness*, 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 *EA*s 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.

Los orígenes de estos algoritmos se remontan a la década de los 50, cuando se empezaron a estudiar los primeros conceptos, pero los mismos no se formalizan sino hasta el trabajo de Holland [HOL75] "*Adaptation in natural systems*".

Aproximadamente desde mediados de la década del 70, en el ámbito de los *EAs*, se han introducido numerosos algoritmos que se pueden clasificar principalmente como Algoritmos Genéticos (*GA*), Programación Evolutiva (*EP*) y Estrategias de Evolución (*ES*), Sistemas clasificadores (*CS*) y Programación Genética (*GP*) [HEI00]. A pesar de que el principio que los soporta es, en apariencia, muy simple, estos algoritmos han probado ser herramientas generales, robustas y potentes [HEI00]. En la siguiente sub-sección se brinda una pequeña introducción a los algoritmos genéticos. Para mayor información relativa a los demás algoritmos evolutivos se refiere al lector a [HEI00].

Para una introducción detallada de los conceptos que dan soporte a los algoritmos, se recomienda el libro introductorio de Goldberg [GOL89].

1.5.1 Algoritmos Genéticos

Los algoritmos genéticos se utilizan en varios ámbitos, principalmente para búsquedas y optimizaciones. En la práctica se implementa el algoritmo escogiendo una codificación para las posibles soluciones del problema. La codificación se realiza mediante cadenas de bits, números o caracteres para representar a los cromosomas. Luego, las operaciones de cruzamiento y mutación se aplican de manera muy sencilla mediante funciones de manipulación de valores de vectores. A pesar de que se han realizado numerosas investigaciones [HEI00] 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 el *fitness* 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.

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. El pseudocódigo 1.1, presentado a continuación, muestra el algoritmo principal.

```
Procedimiento GA( )
Inicio
        t = 0 /* empezar con un tiempo inicial */
        InicializarPoblacion P (t) /* inicializar una población de individuos generados al azar */
        Evaluate P (t) /* evaluar el fitness de todos los individuos de la población inicial */
        Mientras no se cumpla el criterio de parada
                t = t + 1 /* incrementar el contador de tiempo */
                P' = SeleccionarPadres P (t) /* seleccionar una poblacion para generar descendientes */
                Recombinar P' (t) /* recombinar los "genes" del grupo de padres seleccionados */
                Mutar P' (t) /* perturbar la población generada de manera estocástica */
                Evaluar P' (t) /* calcular fitness de la población recién creada */
                P = Sobrevivientes P, P' (t)/* Seleccionar sobrevivientes para la siguiente generación */
        Fin Mientras
Fin
```

Pseudocódigo 1.1. Algoritmo genético básico

El operador de selección, según se ha apuntado, simula el proceso de selección natural en que el más fuerte tiene mayor capacidad de supervivencia. En el *GA* la capacidad de supervivencia de un individuo está ligada al valor numérico de la función objetivo o *fitness*. 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 bajo *fitness* tienen poca probabilidad de ser elegidos, mientras que los de buen fitness son seleccionados con mayor frecuencia.

El operador de selección puede aplicarse de maneras diversas. Unas veces se realiza la selección mediante torneos en que se escoge al azar un grupo de individuos y gana el torneo aquel con mejor *fitness*. 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 recibe el nombre de selección por torneos y refleja de manera más adecuada el proceso natural de selección. Otra forma, conocida como selección de 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 al *fitness* de cada individuo. Es de esperar que al hacer girar la ruleta varias veces, se obtendrá mayor cantidad de individuos con alto *fitness*. Aunque pueden existir otras formas de realizar la selección, las dos anteriormente mencionadas son las más comunes en las implementaciones publicadas. A continuación se presenta el pseudocódigo 1.2, que contiene la descripción del operador de selección de ruleta solamente, ya que el de torneos es muy sencillo y su implementación muy directa.

```
Procedimiento Seleccion( )
Inicio
        ValorAleatorio = numero aleatorio entre 0 y 1
        SumaParcial = 0
        Rand = ValorAleatorio * ∑_{i=1}^{N} f_i
        i = 1
        Mientras ( i ≤ N y Rand ≥ SumaParcial )
                SumaParcial = SumaParcial + f_i
                i = i + 1
        Fin Mientras
        Retornar i
Fin
```

Pseudocódigo 1.2. Algoritmo genético. Operador de Selección

Una vez seleccionados los individuos se aplica a cada par de ellos el operador de cruzamiento. También el cruzamiento se puede realizar de maneras diferentes y aquí solo se discutirá una de ellas: el cruzamiento de 1 punto. En el mismo, se escoge aleatoriamente un punto de corte que se aplicará al par de cromosomas o codificación de los individuos seleccionados. Luego, los caracteres más significativos, a partir del punto de corte se conservan en sus posiciones relativas en el nuevo par de cromosomas, y los restantes son intercambiados de los cromosomas progenitores a los nuevos dos cromosomas obtenidos. Como ejemplo, considérese dos cromosomas A1 y A2 de un par de individuos, sobre los que se aplicará el operador de cruzamiento para obtener dos nuevos cromosomas. Sean los individuos:

A1 = 0 1 1 | 0 1

A2 = 1 1 0 | 0 0

donde el símbolo "|" indica el punto de corte. Los caracteres a la izquierda del punto de corte son los más significativos, por tanto permanecen en sus posiciones correspondientes en los nuevos cromosomas. Mientras que los caracteres a la derecha del punto de corte son cruzados de los cromosomas progenitores, como muestra el ejemplo:

A'1 = 0 1 1 | 0 0

A'2 = 1 1 0 | 0 1

De esta manera, A'1 y A'2 son los cromosomas resultantes de aplicar el operador de cruzamiento al par de cromosomas progenitores.

En el pseudocódigo 1.3 se describe al operador de cruzamiento. En dicho pseudocódigo se tiene que l representa a la longitud (en bits) del cromosoma, *Jcross* constituye el punto de corte, y los vectores *Hijo1* e *Hijo2* equivalen a la representación binaria de una pareja de herederos, mientras que *Padre1* y *Padre2* son los progenitores. La probabilidad de cruzamiento dada indica la probabilidad con que los progenitores se cruzarán y la probabilidad de generar clones; de esta forma, no todos los individuos seleccionados son destruidos por el cruzamiento.

```
Procedimiento Cruzamiento(Padre1, Padre2)
Inicio
        Rand = numero aleatorio entre 0 y 1
        Si Rand es menor a la probabilidad de cruzamiento dada
                JCross = numero aleatorio entre 1 y l
        Sino
                JCross = l
        Fin Si
        Desde i = 1 hasta i = JCross, i = i + 1
                Hijo1[i] = Padre1[i]
                Hijo2[i] = Padre2[i]
        Fin Desde
        Desde i = Jcross + 1 hasta i = l, i = i + 1
                Hijo1[i] = Padre2[i]
                Hijo2[i] = Padre1[i]
        Fin Desde
Fin
```

Pseudocódigo 1.3. Algoritmo genético. Operador de Cruzamiento

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.

Como en la naturaleza, la probabilidad de que ocurra una mutación es pequeña, en las condiciones de vida normales. En el *GA* se trata de representar lo mismo, con un valor de ocurrencia muy bajo para el operador de mutación.

1.5.2 Algoritmos Evolutivos y optimización multiobjetivo

Los *EAs* resultan interesantes ya que, a primera vista, parecen estar 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. 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 [COE00b]), parecen considerar este hecho, y han dado lugar al nacimiento de todo un nuevo campo de investigación: los algoritmos evolutivos

aplicados a optimización multiobjetivo (*MOEA* por sus siglas en inglés*: Multiobjective Optimization Evolutionary Algorithm*).

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* [ISH95, ISH96, FON98b, PARK98]. Más 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], técnicas de niching [OBA98], aplicación de elitismo [PARK98, OBA98], etc. Otros se han dedicado a sugerir nuevos algoritmos y técnicas evolutivas [LAUM98]. También se han publicado resúmenes analíticos generales, comparaciones y compendios [FON95b, HOR97, VAN98a, DEB99a, DEB99b, COE99a].

A pesar de esta gran variedad, aún existen numerosas cuestiones que todavía permanecen abiertas. Entre ellas podemos citar:

a) ¿Qué algoritmos propuestos se desempeñan mejor en qué tipo de problemas?
b) ¿Cuáles son las ventajas y desventajas de un algoritmo particular, respecto a los demás?
c) ¿Qué medida de calidad se utilizará para afirmar que un algoritmo es efectivo en un caso dado?
d) ¿Las técnicas como niching, elitismo y restricciones de cruzamiento pueden mejorar el desempeño de los *MOEAs*?
e) ¿Cuáles son las técnicas más apropiadas para la paralelización de *MOEAs*?
f) ¿Cómo afecta el proceso de paralelización al desempeño de los algoritmos?

Gran parte de las dudas surgen del mismo hecho que los *MOEAs* se aplican a una cantidad enorme de problemas de diversos ámbitos, por lo que una generalización completa parece ser imposible.

Además, es importante evidenciar que algunas aplicaciones del mundo real presentan mayor dificultad para encontrar soluciones no inferiores (Pareto óptimas), con lo que la determinación del frente Pareto completo no es factible. Para estos casos, es necesario redefinir los objetivos de la resolución de *MOPs* considerando lo siguiente:

a) Se debe minimizar la distancia entre el frente Pareto verdadero y el frente conformado por las soluciones (no dominadas) halladas.
b) Se debe tener una buena distribución (generalmente esto equivale a decir una distribución uniforme) de las soluciones identificadas.

c) Es deseable encontrar soluciones bien diferenciadas unas de otras.

1.6 Conclusiones finales y delineamiento de capítulos posteriores

Las cuestiones planteadas ayudan a delinear un marco de desarrollo. En el presente trabajo se pretende dar respuesta a las preguntas enunciadas, para el ámbito específico de aplicación del diseño de espinas dorsales (*backbones*) de redes de computadoras.

Para ello, en primer término, en el siguiente capítulo se introduce el problema a tratar en su formulación natural como *MOP*. También se discuten las funciones objetivo, su elección y la forma de calcularlas.

En el tercer capítulo se discute sobre los *MOEAs* que según la literatura presentan los mejores desempeños. Luego se consideran las técnicas que cada algoritmo implementa para llegar a una comprensión teórica adecuada de los mismos. También se introducen las propuestas de paralelización con las que posteriormente se obtienen los resultados. Esto, junto con el nuevo enfoque que plantea la optimización multiobjetivo, ayudará a contestar las cuatro primeras preguntas planteadas en la sección 1.5.2, siempre dentro de la aplicación propuesta.

En el capítulo cuarto se detalla la metodología utilizada para el diseño de los experimentos con *MOEA*s. Se incluye un apartado con las métricas que se utilizarán para medir el desempeño de los *MOEAs*. Ellas son muy importantes ya que proporcionan un marco dentro del cual se puede comparar los resultados obtenidos y tienen vinculación directa con la forma en que se puede responder a la primera y segunda pregunta de la sección anterior. Además es una síntesis de los esfuerzos realizados hasta el momento para responder a la tercera pregunta.

El capítulo quinto contiene los resultados de las corridas de los *MOEAs*, tanto en un ambiente secuencial (en un solo computador), como en un ambiente paralelo asincrónico (conformado por varios computadores) para un problema de diseño escogido. Con esto se explorará el medio paralelo propuesto en las últimas preguntas.

Por último, se agrega un capítulo que incluye un sumario de los resultados fundamentales, y además, ofrece perspectivas para trabajos futuros.

El apéndice A contiene unas publicación en conferencia internacional arbitrada que ha resultado del presente trabajo. Las implementaciones de *MOEAs* están disponibles para quienes deseen obtenerlo enviando un correo electrónico de solicitud a sduarte@cnc.una.py.

2 Diseño de Espinas Dorsales para Redes de Computadoras

2.1 Introducción

Según se introdujo en el capítulo anterior, la optimización multiobjetivo puede aplicarse a un número muy grande de problemas. En particular, para este trabajo, se considerará el problema del diseño de la espina dorsal (*backbone*) de una red de computadoras.

Uno de los objetivos esenciales del diseñador es minimizar el costo total, mientras otro es el de maximizar la confiabilidad. Dependiendo del tipo de datos que conformará el tráfico promedio de la red surgen otros objetivos como minimizar los retardos o maximizar la cantidad de paquetes entregados por la red (*throughput*). Cada uno de estos objetivos puede formularse explícitamente como una función objetivo para el proceso de optimización, o incluirse como una restricción. Claramente estamos en presencia de un *MOP*.

En este capítulo damos el modelo matemático del problema del diseño de *backbones* de redes al que aplicaremos los *MOEAs*. Luego introducimos los objetivos escogidos para optimizar y también las restricciones, conjuntamente con los algoritmos empleados para los cálculos correspondientes. En la sección 2.5 presentamos un resumen históricos de las técnicas que se han propuesto anteriormente para tratar el problema del diseño de *backbones* de redes, junto con una breve discusión de sus desventajas y limitaciones. Finalmente, añadimos una breve discusión para ayudar al lector a comprender mejor la complejidad del problema de búsqueda planteado.

2.2 Formulación matemática del problema

Una red de computadoras puede modelarse mediante un grafo no dirigido $\boldsymbol{G} = \{V, L\}$, donde:

- V es el conjunto de nodos, y
- L es el conjunto de enlaces. La cardinalidad de L ($|L|$) es, entonces, el número de posibles enlaces en la red. Si a éste llamamos n, el mismo puede expresarse como:

$$n = |L| = \frac{|V|(|V|-1)}{2} \quad \textbf{(2.1)}$$

Entre cada par de nodos $i, j \in V$ pueden existir distintos tipos de conexiones dependiendo de las tecnologías de comunicación disponibles. De esta forma $l_{ijt} \in L$ representa un enlace que interconecta a los nodos i y j, utilizando la tecnología $t \in \{1, 2, 3, \ldots, T\}$ con un costo c(l_{ijt}) y un confiabilidad $r(l_{ijt})$. Ejemplos de tecnologías de interconexión podrían ser: fibras ópticas, radio enlaces, micro ondas, cables con aislamientos, etc.

El problema del diseño de redes consiste en escoger los enlaces que se colocarán en la red, con su tecnología de comunicación correspondiente. Para realizar la elección, se conoce previamente la ubicación geográfica de los nodos, y los costos y confiabilidades de los diferentes enlaces potenciales. La red resultante debe cumplir con un cierto conjunto de objetivos y adecuarse a un conjunto de restricciones, según lo establece el diseñador.

Debido a que el problema, así enunciado, puede ser tan grande como el diseñador lo desee (es decir, él puede escoger tantos objetivos como desee, puede contar con tantas tecnologías de comunicación como le permita su presupuesto, etc.) surge la necesidad de limitarlo. En el presente trabajo lo expresamos como la optimización de $k = 2$ objetivos: costo y confiabilidad. Se cuenta con un requerimiento que indica que cada topología de red debe representar un grafo conectado. Esto se traduce a restringir la confiabilidad resultante de cada configuración a valores positivos, i.e. mayores a cero; luego, las soluciones propuestas deben cumplir solo con un requerimiento de confiabilidad mínima ($m = 1$). Además, se asume la existencia de un único enlace bidireccional entre cada par de nodos, i.e. la redundancia no está permitida. De esta manera, los enlaces potenciales entre cada par de nodos son las variables de decisión. Luego, cada variable de decisión $\boldsymbol{x}$ se compone de una n-tupla ($x_1, x_2, \ldots, x_n$), donde cada x_i representa un enlace entre un par de nodos de la red, con su tipo, costo y su confiabilidad correspondiente.

Las restricciones sobre redundancia y cantidad de objetivos son solo aparentes y no hacen a la formulación del problema menos general ya que la adición de nuevos objetivos es un problema trivial, aunque pueda requerir mayores recursos computacionales. Además, los enlaces redundantes se pueden tratar como otro tipo de enlaces, con su propio costo y confiabilidad [LAU00].

Luego, el problema queda formulado como:

$$\begin{aligned} &\textit{Optimizar} && \boldsymbol{y = f(x) = (f_1(x), f_2(x))} \\ &\textit{sujeto a} && \boldsymbol{e_1(x) > 0} \end{aligned} \quad \textbf{(2.2)}$$

donde:

- $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \boldsymbol{X}$, es el vector de decisión y representa una topología; cada $x_i \in \{0, 1, 2, \ldots, T\}$ representa un (tipo de) enlace entre cada par de nodos.

- T es la cantidad de tipos de enlaces diferentes (fibras ópticas, enlaces de radio, cables coaxiales, etc.; el valor 0 se utiliza para indicar la ausencia del enlace).
- $\boldsymbol{y} = (y_1 = f_1(\boldsymbol{x}), y_2, = f_2(\boldsymbol{x})) \in \boldsymbol{Y}$, es el vector objetivo.
- $f_1(\boldsymbol{x})$ es la confiabilidad correspondiente a la configuración $\boldsymbol{x}$
- $f_2(\boldsymbol{x})$ es la función de costo de la misma configuración $\boldsymbol{x}$
- $e_1(\boldsymbol{x})$ se refiere a la confiabilidad mínima aceptable (restricción de grafo conectado)

Según lo expuesto, las soluciones factibles del problema, i.e. las soluciones que pertenecen a $\boldsymbol{X}_f$, serán los vectores de decisión que cumplan con el requisito de confiabilidad mínima.

Además, asumimos que los nodos son perfectamente confiables, es decir, no fallan, mientras que los enlaces pueden estar solamente en uno de dos estados posibles: operacionales o en falla. Si un enlace falla no se considera su reparación, es decir la confiabilidad se calcula considerando el tiempo hasta una falla y sin considerar las reparaciones de estas fallas. Además, se asume que las fallas en los enlaces ocurren independientemente unas de otras, i.e. la falla de un enlace no conduce a fallas de otros enlaces.

En este punto cabe destacar que, si bien parámetros como velocidad, *througput* y retardos son importantes para las aplicaciones más modernas, los principales objetivos en el diseño de redes han sido tradicionalmente el costo y la confiabilidad [COL98, LAU00]. Ambas funciones han sido estudiadas en casi todos los artículos encontrados referentes a problemas de optimización de diseños. El caso específico de diseño de redes se ha presentado en dos variantes. En la primera se expresaba el problema como minimizar el costo manteniendo una restricción sobre la confiabilidad; mientras que en la segunda se presentaba el problema dual correspondiente: maximizar la confiabilidad manteniendo una restricción sobre el costo. Una tercera variante ha sido sugerida pero no implementada en [DEN97] y propone aplicar al problema el método de la suma con pesos descrito en el capítulo anterior.

Nuestra formulación del problema se basa en un planteamiento propuesto por Dengiz y Smith en [DEN97] y posteriormente adoptado en [DEE97, DEE98, LAU00]. Todas estas aproximaciones utilizan algoritmos genéticos para resolver versiones simplificadas del problema de un solo objetivo, obtenido a partir de las técnicas tradicionales. Sin embargo, la naturaleza multiobjetivo del problema no ha sido previamente evidenciada.

Para ilustrar la formulación propuesta, se presenta un pequeño ejemplo de diseño de una red a partir de la información relativa a la ubicación geográfica de los nodos. Considérese la grilla de la figura 2.1; en la misma se han colocado 5 nodos a interconectar (representados como puntos) de manera aleatoria, y se cuenta con 3 tipos de tecnología para los 10 enlaces posibles. A cada nodo se ha asignado un número para identificarlo. El desafío consiste en determinar qué enlaces se colocarán entre qué pares de nodos, indicando, además, el tipo del mismo. Se pretende que la red constituida tenga el menor costo y la mayor confiabilidad posible.

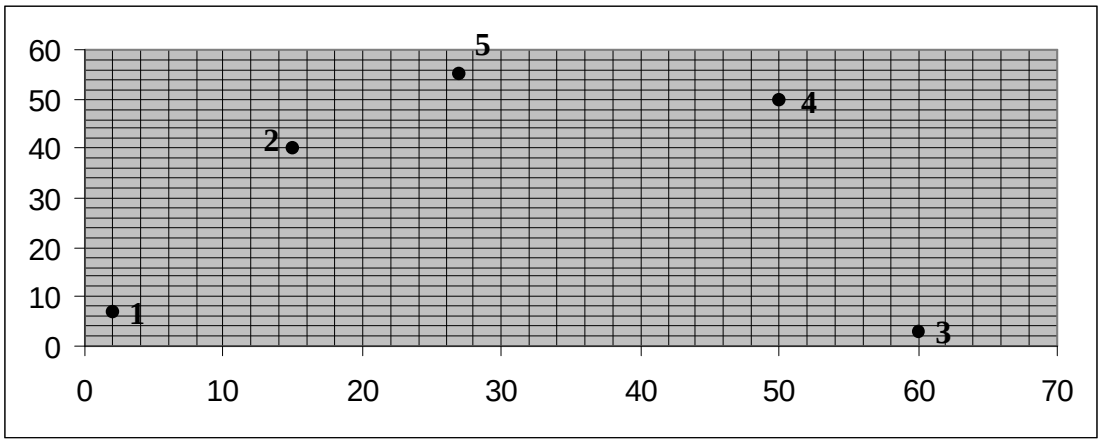


Figura 2.1. Conjunto de nodos a interconectar.

Este conjunto de nodos se utilizará en las siguientes secciones, como ejemplo de diseño de una red de computadoras.

2.3 Confiabilidad de redes de computadoras

Intuitivamente, el concepto de confiabilidad indica la probabilidad que tiene un sistema de desempeñarse adecuadamente en un intervalo de tiempo establecido. Directamente surge la estrecha relación existente entre la confiabilidad del sistema considerado como un conjunto, la configuración del mismo y la confiabilidad de sus componentes.

En [EDR80] se encuentra desglosada una larga lista de métodos de cálculo de confiabilidad para diversos tipos de sistemas y aplicaciones. Sin embargo, en algunos casos –debido a los altos costos computacionales involucrados– no existe forma práctica de calcular la confiabilidad exacta de un sistema en un tiempo razonable. En tales casos no queda más remedio que sustituir el cálculo exacto por métodos de estimación que otorgan buenas aproximaciones.

Al tratar el caso particular de redes de computadoras se pueden definir diversas formas de expresar la confiabilidad, dependiendo de la situación que queremos reflejar [COL98]. Generalizando, podemos decir que una red es confiable si permite el intercambio de datos entre un conjunto de terminales (nodos). Para el ejemplo de la figura 2.1, afirmaríamos que la red lograda es confiable si permite intercambio de información entre todos los nodos durante un periodo de tiempo establecido. Luego, podemos afirmar que la confiabilidad está basada en la *conectividad*.

Por otro lado, se podría argumentar que la sola posibilidad de conexión es aún insuficiente y deseamos añadir a ésta el requerimiento de que la red se desempeñe de manera adecuada en presencia de fallas, es decir, que además sea robusta. Por ejemplo, si cae uno de los enlaces presentes, la red debería ser capaz de enviar los paquetes por caminos alternativos, evitando además la congestión general. Evidentemente, este requerimiento es muy útil para indicar el

desempeño de la red considerada en conjunto; sin embargo, las métricas que lo consideran no son muy utilizadas en el campo académico [COL98]. Se cree que esto puede deberse a tres motivos [COL98]:

1- Para redes pequeñas que utilizan equipamientos poco confiables la desconexión de la red es la mayor preocupación.
2- Las métricas que consideran la robustez son más complicadas de calcular que las basadas en conectividad, y en general los académicos prefieren buscar propiedades en sistemas simples y luego extenderlos a los complejos.
3- A pesar de las simplificaciones asumidas por las métricas de conectividad, el problema sigue planteando desafíos que no se han resuelto de manera satisfactoria.

Por los motivos mencionados, en el presente trabajo hemos decidido utilizar métricas basadas en conectividad, aunque manteniendo siempre la posibilidad de extensiones; ya que una vez propuesto un método para calcular otras métricas, la inclusión de las mismas en el algoritmo se hace de manera sencilla, aunque el programa resultante sea computacionalmente más complejo. Además, mantener la utilización de éste tipo de métricas nos permite realizar comparaciones más directas con trabajos anteriores de la misma índole, como [LAU00, DEE97, DEE98].

La métrica de confiabilidad escogida para nuestra implementación se conoce como confiabilidad uniforme. Se trata de que en un momento dado, todos los nodos de la red puedan comunicarse con cualquiera de los demás, i.e. debe existir al menos un camino para enviar paquetes de un nodo de la red a cualquier otro. Por tanto esta métrica se define utilizando el concepto de conectividad de una red de computadoras definida según el modelo de redes presentado en la sección 2.2.

Definición 2.1. Conectividad de una red de computadoras. Dados dos conjuntos de nodos de la red $S \subseteq V$ y $S' \subseteq V$, se dice que la red posee conectividad si para cada nodo $s \in S$ existe al menos un camino (entendido del modo tal en que se definen los caminos en la teoría de grafos, i.e. ver [HORO83]) que comunique a s con cada nodo $s' \in S'$.

Dada la definición de conectividad, existen dos casos especiales de singular importancia. El primero es el caso en que los dos conjuntos mencionados *S* y *S'* contienen, cada uno de ellos, un solo nodo, y el nodo contenido en S es diferente al de S'; en tal caso tenemos reflejada una conexión punto a punto. El segundo caso, refleja la situación en la que ambos conjuntos son iguales al conjunto de nodos *V*, i.e. para cada nodo de la red existe siempre al menos un camino que lo comunica con el resto. A este se conoce como conectividad uniforme.

La métrica de confiabilidad uniforme es la que exige la conectividad uniforme y expresa la probabilidad de su existencia en un periodo dado de tiempo. A partir de aquí utilizaremos indistintamente los términos *confiabilidad* y *confiabilidad uniforme* para hacer referencia a la probabilidad de que la red se encuentre uniformemente conectada en un tiempo dado. Nótese

que la métrica de confiabilidad uniforme exige que la red forme –al menos- un árbol sumidero (*spanning tree*).

2.3.1 Cálculo de la confiabilidad de redes de computadoras

Sin lugar a dudas, el método de la enumeración exhaustiva es el más sencillo para calcular la confiabilidad de una red, ya que basta con enumerar todos los estados posibles y ver en cuáles de ellos se cumple con la restricción de confiabilidad uniforme. Sin embargo, el número de estados posibles depende de la cantidad de nodos que posee la red, i.e. el número de nodos indica la cantidad de enlaces posibles, y de la cantidad de tecnologías de comunicación disponibles. Es más, el número de estados crece exponencialmente con el crecimiento de los parámetros mencionados, convirtiendo al problema en un problema *NP-Hard* [DEN97]. Esto hace que, aún para redes de tamaño pequeño, el método carezca de toda utilidad práctica. La misma observación es aplicable a métodos que calculan la probabilidad de manera exacta, como por ejemplo el *backtracking* [DEE98]. Por tanto, es necesario recurrir a métodos de cómputo que se limiten a estimar el valor buscado.

Los métodos que estiman la confiabilidad en un sistema pueden ser de dos tipos diferentes:

a) los que calculan cotas, y
b) los que estiman aproximaciones.

Los primeros proporcionan un valor exacto que puede ser comparado con otro dado para servir como filtro, i.e. si la confiabilidad mínima requerida es superior a la cota superior de confiabilidad de un sistema, tal sistema puede ser rechazado. Los segundos no involucran cálculos exactos, sino estiman el valor mediante simulaciones y estadísticas sobre las mismas, por lo que siempre tienen un margen de error.

El problema con los métodos de tipo (a) es que son poco generales y propios del sistema a evaluar; más aún, distintos métodos, basados en diversas propiedades de un mismo sistema, pueden arrojar diferentes valores para las cotas.

Entre los métodos de tipo (b) los más populares son las simulaciones Monte Carlo, por ser extremadamente simples. Para comprender la simulación Monte Carlo recurrimos al siguiente ejemplo. Supongamos que deseamos estimar la probabilidad de que en el lanzamiento de una moneda al aire se obtenga cara. La propuesta consiste sencillamente en lanzar la moneda un número *n_l* de veces; y apuntar, luego de cada lanzamiento, la cantidad de veces que se ha obtenido cara hasta el momento. Si a este numero llamamos *n_c*, se puede estimar la probabilidad buscada mediante el cociente $\frac{n\_c}{n\_l}$. A cada lanzamiento de la moneda se conoce como una replicación, y al conjunto completo de lanzamientos como una simulación. Resulta

intuitivo derivar que al aumentar el número de replicaciones *n_l*, mejor será la aproximación obtenida.

El pseudocódigo 2.1 calcula la confiabilidad de una red dada, empleando simulaciones Monte Carlo. El procedimiento es sencillo, ya que mediante simulaciones sistemáticas sobre todos los enlaces de la red dada se genera una nueva red, que cuenta con los enlaces que tienen probabilidad de estar funcionando en ese momento. Si la red así generada cumple con el requisito de conectividad uniforme, i.e. conforma por lo menos un *spanning tree*, entonces es un caso positivo que se sumará al numerador de nuestra fórmula, para obtener luego el cociente que indicará la confiabilidad. El procedimiento es como sigue:

```
Procedimiento de Cálculo de Confiabilidad(x)
Inicio
                                        i = 0
        cont = 0
        Mientras i < Numero_replicaciones_MC
                Generar una red NET, a partir de la probabilidad de funcionamiento de los enlaces ya
                        presentes en x
                Si la red NET forma un spanning tree
                        cont = cont + 1
                Fin Si
                i = i + 1
        Fin Mientras
        Confiabilidad_estimada = cont/Numero_replicaciones_MC
End
```

Pseudocódigo 2.1. Cálculo de confiabilidad

Para el ejemplo de diseño de la red de la figura 2.1, primeramente generamos una red cuya confiabilidad queremos evaluar. Por ejemplo, la red mostrada en la figura 2.2.

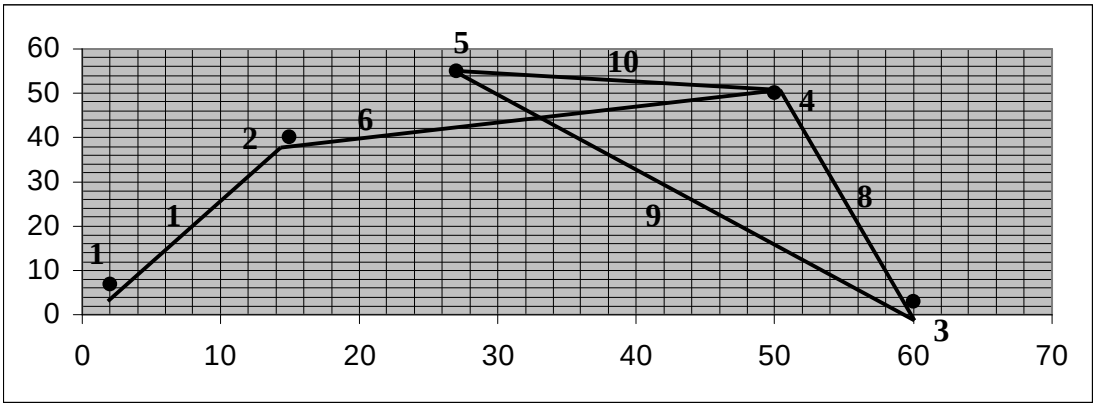


Figura 2.2. Red cuya confiabilidad se desea calcular.

Luego, cada simulación Monte Carlo consistirá en lo siguiente: a) tomar el enlace 1 y verificar su tipo, por ejemplo el enlace con la etiqueta 1, que se supone tiene confiabilidad 0.975; b) obtener aleatoriamente un número entre 0 y 1; c) si el número obtenido es mayor a la

confiabilidad del enlace, se asume que el enlace no está operativo y no se añade el enlace no operativo a la nueva red; d) si el número obtenido es menor o igual a la confiabilidad del enlace, se asume que el enlace está operando y para indicarlo se añade ese enlace a la red. Siguiendo entonces con el enlace 1, si el número aleatorio es menor o igual a 0.975, el enlace estará presente en la red resultante. Este mismo proceso se repetirá por cada enlace presente en la red original. De ese modo se obtendrá probabilísticamente una nueva red. Si la red obtenida está conectada, se sumará una replicación exitosa. El procedimiento completo vuelve a realizarse sobre la misma red una cantidad predefinida de veces (*Numero_replicaciones_MC* según el pseudocódigo 2.1). Finalmente la confiabilidad de la red se estima mediante el cociente entre la cantidad de replicaciones exitosas y la cantidad total de replicaciones.

Dos cuestiones adicionales son importantes antes de dejar la discusión sobre cálculo de confiabilidad de redes. La primera es recalcar que el método de simulación Monte Carlo propuesto es un estimador sin tendencias; mientras que la segunda es establecer la varianza del método descrito como [DEE98]:

$$Var(\mathrm{Re}\,liability(x)) = \frac{\mathrm{Re}\,liability(x)(1-\mathrm{Re}\,liability(x))}{Number\_MC\_replications} \quad \textbf{(2.3)}$$

Luego, la aproximación al resultado real mejora al aumentar el número de replicaciones *Numero_replicaciones_MC*.

2.4 Costo de una red de computadoras

El costo asociado a cada configuración de la red *Cost*(***x***) está dado por la suma de los costos de sus enlaces. Ahora bien, el costo de cada enlace depende, a su vez, del tipo de tecnología empleada en el enlace. Una manera de establecer el costo de un enlace es conociendo la distancia que cubre y el precio por unidad de distancia que implica la colocación de una tecnología específica. Esa es la aproximación utilizada en este trabajo. Luego, el costo de una red dada ***x*** es:

$$f_2(\mathbf{x}) = Cost(\mathbf{x}) = \sum_{i=1}^{n} dist\_unit_i * \mathrm{cost}(x_i) \quad \textbf{(2.4)}$$

donde

- $cost(c_i)$ es una función que retorna el costo de colocar el enlace *i* del tipo escogido x_i, por unidad de distancia, y

- $dist\_unit_i$ es la distancia, –en unidades de distancia– cubierta por el enlace *i*.

La implementación y el cálculo de la función de costo, luego, se realiza de manera directa y sencilla.

Para ilustrar este cálculo, el costo de la red de la figura 2.2 estará dado por la suma de los costos de los enlaces 1, 6, 8, 9 y 10. En las tablas 2.1 y 2.2 se incluyen los datos necesarios.

Tipo de Enlace	Costo Unitario ($/Km)	Confiabilidad
1	333	0.96
2	433	0.975
3	583	0.99

Tabla 2.1. Costos y confiabilidades de los diferentes tipos de enlace

Nº Enlace	Origen	Destino	Distancia	Costo T1	Costo T2	Costo T3
1	1	2	35.47	11810.94	15357.77	20678.01
2	1	3	58.14	19359.88	25173.65	33894.31
3	1	4	64.44	21459.78	27904.15	37570.72
4	1	5	54.12	18022.04	23434.06	31552.09
5	2	3	58.26	19399.93	25225.73	33964.44
6	2	4	36.40	12121.38	15761.44	21221.52
7	2	5	19.21	6396.721	8317.658	11199.06
8	3	4	48.05	16001.33	20806.54	28014.35
9	3	5	61.59	20508.58	26667.32	35905.42
10	4	5	23.54	7837.889	10191.61	13722.19

Tabla 2.2. Distancias y costos de los diferentes enlaces posibles

Para conocer el costo del enlace 1 se debe primero verificar su tipo. Como es de tipo 2 (confiabilidad de 0.975), su costo unitario está dado por 483$/Km. Como la distancia entre los nodos 1 y 2 es la medida de la longitud del enlace 1, el costo del mismo enlace se obtiene de multiplicar esta distancia por el costo unitario. Dado que la distancia es de 35,47 Km., el costo de la presencia de ese enlace en la red es de 11810.94$. De idéntica forma se obtiene el costo de los demás enlaces, y la suma indica el costo total de la red.

2.5 Técnicas aplicadas previamente al problema

Para ayudar al lector a ubicarse mejor en el estado actual de los trabajos en esta área, se presenta a continuación un pequeño resumen de técnicas y métodos que se han aplicado previamente al problema en cuestión, con sus limitaciones y desventajas; las cuales se pretende superar con el presente trabajo. Además, tal resumen ayudará a tener una idea más clara de los motivos por los que se ha escogido la utilización de algoritmos evolutivos. Los primeros métodos propuestos estaban basados en heurísticas para perturbar grafos. Posteriormente con las técnicas de inteligencia artificial y aquellas fundadas en meta-heurísticas, emergen numerosas aplicaciones de las mismas al problema de diseño de *backbones* de redes. En consecuencia, se

dividirá la presente sección en dos partes: a) los métodos más antiguos, que no tienen más que una importancia histórica; y, b) los métodos basados en inteligencia artificial, que se presentan como una opción que está dando resultados promisorios en el presente.

2.5.1 Aproximaciones fundadas en perturbación de grafos

Generalmente, toman una solución inicial (que no necesariamente debe ser buena) y luego la hacen evolucionar hacia un óptimo –que casi siempre resulta local–, mediante alteraciones en la disposición de los enlaces y otras heurísticas. Un ejemplo de esta metodología fue el *BXC* (*Branch Exchange Method*), propuesto por Tanembaum en 1981 [TAN81]. Se trata de un ciclo iterativo en el que, a partir de una solución inicial, se reemplazan dos enlaces relativamente cercanos, por otro par, usando diferentes combinaciones de los cuatro nodos más afectados por la eliminación de los dos primeros enlaces.

Otro algoritmo es el conocido como *CSA* (*Cut-Saturation Algorithm*), que data del año 1977 [BOO77]. Posee cuatro operaciones básicas: a) agregar, b) borrar, c) perturbaciones y d) colapso de cadenas. En la primera operación se halla una cortadura saturada (conjunto de enlaces que al borrarse separan a la red en dos sub-redes *G1* y *G2*) eliminando aquellos enlaces más utilizados en la red. Luego, se agrega un enlace $\{x, y\}$ a la red, de modo que $x \in V1$ e $y \in V2$, y los nodos x e y estén al menos a cinco enlaces de distancia uno de otro. En la operación de borrado, se elimina de la red al enlace menos utilizado y de mayor precio. La operación de perturbación es una combinación de las dos anteriores. Finalmente, en el colapso de cadenas, se busca un camino relativamente largo entre los nodos x e y, y se lo reemplaza por el enlace $\{x, y\}$. Se ha reportado que el *CSA* obtiene mejores resultados que el *BXC* [GER77]. En 1991 [NEW91] se presenta una modificación del *CSA* original. En esta versión no se considera solo el desempeño de la red, sino que además se agrega una métrica de capacidad de supervivencia de la misma.

MENTOR, presentado también en 1991 [KER91], se basa en la idea de que debe existir un enlace directo entre aquellos nodos con alto volumen de tráfico. Así, aquellos nodos que no intercambian muchos paquetes, pueden conectarse pasando por nodos intermedios. Todas las topologías generadas por *MENTOR*, tratan de ajustarse a esta propiedad. De esta manera va generando diferentes topologías que cumplan con esta propiedad.

Gavish [GAV92] formula un problema de diseño de redes unificado como un problema de optimización no lineal, e incluye tanto a *LAN*s como a *backbones*. La metodología de solución propuesta consiste en procedimientos de optimización basados en el gradiente. Como último ejemplo, en 1993, Jan [JAN93] presentó un método para optimizar costos considerando un requerimiento de confiabilidad mínima. En este trabajo se estima la confiabilidad de la red computando cotas superiores.

Por otro lado, con la creciente utilización de los métodos de inteligencia artificial y las técnicas meta-heurísticas, han surgido muchas aplicaciones de éstos métodos de optimización modernos al diseño de *backbones* de redes. Las mismas se discuten en la siguiente sección.

2.5.2 Métodos basados en inteligencia artificial

Existen varios métodos basados en inteligencia artificial y con diferentes herramientas. Las técnicas de búsqueda más usadas son: enfriamiento simulado (*simulated annealing*), búsqueda tabú y algoritmos evolutivos. Estos últimos han demostrado ser muy eficientes y su uso se ha extendido notoriamente.

Se ha utilizado, por ejemplo, en [PIE95], el templado simulado (*SA*, del inglés *simulated annealing*) para hallar redes de mínimo costo y que estén conectadas (con conectividad *K*), satisfaciendo a la vez un requerimiento sobre el retardo promedio máximo de los paquetes. El uso de búsqueda tabú (*TS*, por sus siglas en inglés: *tabú search*) [PIE97] se ha propuesto para resolver el mismo tipo de problema. También Costamagna en [COS98] utiliza *TS* para el diseño de *LAN*s, y Lee lo utiliza para el diseño de redes con topología de anillo [LEE97].

Los algoritmos evolutivos también se han aplicado previamente, pero nunca aquellos especializados en optimización de objetivos múltiples. En [DEN97, DEE97, DEE98] se combinan algoritmos genéticos con búsqueda local, algoritmos reparadores, métodos de cómputo de cotas y simulaciones Monte Carlo, para hallar la topología de red de menor costo manteniendo una restricción sobre la confiabilidad. En [LAU00] se relajan algunas restricciones del trabajo de Dengiz y de Deeter y se generan resultados para situaciones más realistas, aunque sin obtener más que un solo miembro del conjunto Pareto óptimo. Por otro lado, se han utilizado también algoritmos genéticos para tratar otros problemas relacionados a redes de computadoras según se ve en [DEE98] donde figuran publicaciones dedicadas en particular al diseño de redes ópticas; también en [DEE98] se mencionan trabajos donde se trata el problema de hallar el mínimo árbol sumidero; y la expansión de redes ya existentes.

Así también, en [DEN97] se han aplicado algoritmos genéticos, operadores de búsqueda local y algoritmos de reparación para hallar la topología de red con costo mínimo y que esté sujeta a un requerimiento de confiabilidad mínima. En este estudio se asumía que los enlaces eran idénticos y se estimaba la confiabilidad mediante el cómputo de cotas y simulaciones Monte Carlo. En estudios similares [DEE98, LAU00], se levanta la restricción de que todos los enlaces tienen la misma confiabilidad y se permite la existencia de enlaces redundantes. Además, en [LAU00] se propone la implementación del algoritmo genético en ambientes de computación distribuida y asincrónica.

De las aproximaciones mencionadas, algunas pueden calificarse como de búsqueda local, otras como técnicas meta-heurísticas. Las primeras sufren la amenaza de quedar varadas en

óptimos locales; mientras que *SA* y *TS* dependen del concepto de búsqueda en la vecindad con lo que no exploran satisfactoriamente todo el espacio de búsqueda. Los algoritmos genéticos, sin embargo, no sufren de estas limitaciones y además permiten la extensión natural para el tratamiento del problema como un *MOP*; así que es este el enfoque escogido.

La utilización de algoritmos genéticos ha permitido encontrar soluciones a redes de hasta decenas de nodos, en tiempos razonables; sin embargo, hasta ahora la búsqueda se ha limitado al tratamiento del problema como si tuviese un objetivo único. En verdad, como ya se ha expuesto en el capítulo anterior, el problema es de naturaleza multiobjetivo, aunque hasta ahora no se haya propuesto una metodología para tratarlo adecuadamente.

Hasta el presente, el problema de optimización se ha construido considerando a todos los objetivos, excepto uno, como restricciones; mientras que el sobrante ha sido la función objetivo del problema de un objetivo resultante [LAU00]. Para encontrar una solución Pareto óptima, se ha escogido un valor adecuado como cota para las restricciones; y luego se ha procedido a resolver el problema de búsqueda con un único objetivo, cuidando de que la solución propuesta cumpla con las restricciones.

Esta aproximación tiene las siguientes desventajas [SRI94]:

- El algoritmo se debe aplicar varias veces para encontrar las soluciones Pareto óptimas múltiples. 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.
- Se requiere un conocimiento previo del problema a resolver, así como del rango de valores adecuado que se asignará a cada restricción. Además, el algoritmo puede ser sensible a éstos valores.
- 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.

Luego, atendiendo a la naturaleza multiobjetivo del problema en cuestión, según la definición expresada en [CHA83], así como al crecimiento exponencial del espacio de búsqueda, es necesario contar con herramientas que cumplan con los siguientes requisitos:

- Que permitan añadir tantas funciones objetivos como se desee.
- Que no impongan restricciones sobre los tipos de enlaces o la naturaleza de las conexiones.
- Que manejen espacios de búsqueda de cualquier tamaño y características en un tiempo razonable.
- Que puedan generar múltiples soluciones en una sola corrida, permitiendo al diseñador tener toda una gama de opciones válidas.

En el capítulo siguiente se presentarán dos algoritmos que cumplen con estas condiciones.

2.6 Complejidad del problema planteado

Para tener una idea de la complejidad del problema a tratar, se presenta un ejemplo. Supongamos que en Paraguay se desea establecer un *backbone* nacional que comunique a todos los proveedores de servicio de Internet (*ISP* por sus siglas en inglés), las empresas de estado, las entidades públicas y las empresas privadas de importancia conformando una gran red a nivel nacional. Una red como tal debe implica enormes exigencias a nivel económico y además, para resultar útil, tiene parámetros de desempeño mínimos que debe cumplirse. Un proyecto como este, en un país como el Paraguay, demanda que las inversiones sean aprovechadas al máximo, de manera tal que la estructura de la red provea la confiabilidad máxima al menor costo posible.

Asumiendo que pretendemos resolver este problema utilizando búsqueda exhaustiva, debemos escoger un diseño e ir comparándolo sucesivamente con todos los demás. Si alguno resultare mejor ingresará al conjunto de soluciones y desplazará al anterior y posiblemente también a otros del conjunto de soluciones posibles. Si alguno resultare no comparable, se lo agregará al conjunto de soluciones. El procedimiento seguirá de esta manera hasta hallar todos los miembros del conjunto Pareto óptimo. Como el delineado del frente Pareto óptimo es sencillo en dos dimensiones, se puede presentar el mismo a un responsable de toma de decisiones, quien escogerá un diseño de compromiso, según sus prioridades y deseos.

Aunque tal metodología es sencilla y eficaz (necesariamente conduce a obtener las mejores redes posibles), tropieza con el problema del aumento exponencial de la dimensión del espacio de búsqueda. Supóngase que en el ejemplo contemos con un conjunto de V nodos a intercomunicar, y tenemos T tipos de tecnologías de comunicación disponibles. Si no consideramos redundancias de enlaces y si suponemos que los enlaces son bidireccionales e idénticos, el número total de distintos diseños a evaluar sería de $(T+1)^{\frac{|V|*(|V|-1)}{2}}$, donde $|V|$ es la cantidad de nodos que componen el conjunto *V*. Si hacemos $|V|$ = 20, y T + 1 = 4, tenemos 2.46 * 10^{114} diseños de redes a ser analizados. Asumiendo que cada red es revisada en una sola instrucción de máquina (lo cual es una gran simplificación, ya que el procedimiento para analizarla es también complejo), si se utilizara una IBM SP2 de 266 MFLOPS se tardaría aproximadamente 2.94 * 10^{98} años, lo que equivale a más de 10^{88} veces la edad del universo. Eso aún sin considerar que una red de apenas 20 nodos es una red aún muy pequeña, considerando las exigencias de conectividad actuales.

Evidentemente, para este problema, el uso de búsqueda exhaustiva no tiene practicidad alguna, por lo que se necesitan métodos alternativos que no exijan el examen completo 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 propuestas últimamente presentadas de algoritmos genéticos multiobjetivo [ZIT99, SRI94, etc.], hacen de estos últimos candidatos ideales para ser aplicados en este ámbito. Debido a que estos algoritmos no han sido ampliamente estudiados en aplicaciones reales, el presente trabajo es un aporte original y útil en el estudio de *MOEAs*.

La ventaja principal de los *MOEAs* es su paralelismo inherente, ya que 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. El siguiente capítulo se dedicará al estudio de los algoritmos evolutivos específicamente diseñados para lidiar con problemas de optimización multiobjetivo.

3 Algoritmos Evolutivos para la optimización multiobjetivo

3.1 Introducción

Los *MOEAs* (algoritmos evolutivos para la optimización multiobjetivo) son herramientas algorítmicas desarrolladas recientemente para la resolución de *MOPs*.

Como hemos apuntado en capítulos previos, su popularidad se debe principalmente a que:

a) resultan promisorios para la integración de los aspectos de búsqueda y de toma de decisiones que plantean los *MOPs*;
b) pueden realizar búsquedas aún en espacios ilimitados y altamente complejos;
c) tienen la habilidad de mantener toda una población de soluciones;
d) existen pocas alternativas –aparte de ellos– para el tratamiento adecuado de *MOPs*.

La característica principal de un *MOEA* es que optimiza simultáneamente un conjunto múltiple de objetivos.

En [VAN99] se puede hallar una lista prolija de reportes de utilización de *MOEAs* para la solución de *MOPs* complejos. Entre las proposiciones allí mencionadas, las diferencias principales radican en:

a) la manera en que se realizan los procedimientos para asignar *fitness* y seleccionar individuos para el cruzamiento; todo ello de modo tal que se guíe la búsqueda hacia zonas donde se encuentren miembros del conjunto Pareto óptimo;
b) la forma de mantener la diversidad de la población para evitar la convergencia prematura y lograr un resultado bien distribuido, de acuerdo con la aplicación particular.

En este sentido, se han empleado las siguientes técnicas:

i. Asignación de *fitness* y selección:
   - Selección con intercambio de objetivos [SHA84, SHA85].
   - Selección con adición de objetivos con variación de parámetros [VAN99].
   - Selección basada en dominancia o basada en Pareto [HOR94, SRI94, ZIT98a, ZIT98b].

ii. Diversificación de la población:
   - *Fitness sharing* [HOR97] que se puede implementar en el espacio de los individuos, en el espacio de decisión o en el espacio de objetivos.
   - *Crowding* [DEJ95].
   - Restricciones para el emparejamiento [HOR94].
   - Aislamiento por distancia [VAN99].

- Formación de sub-especies dentro de las especies (overspecification) [HOR97].
- Reinicialización [VAN99].

Otra técnica, que se ha aplicado con éxito a los *MOEAs*, es el elitismo. Por la naturaleza misma de estos algoritmos, el concepto de elitismo ha tenido que ser revisado y replanteado para su aplicación. El uso de elitismo en *MOEAs* obliga a considerar dos interrogantes bien definidas:

a) ¿Qué individuos se mantienen y por cuánto tiempo en el conjunto élite?

b) ¿Cuándo, cómo y cuáles individuos del conjunto élite se volverán a insertar a la población activa del *MOEA*?

Ambas cuestiones se han resuelto, esencialmente, de dos formas diferentes:

a) Copiar los individuos dominantes (o una fracción de los mismos) de la población actual a la población siguiente [RUD98, DEB00].

b) Mantener una población externa conformada por individuos dominantes que se van encontrando a medida que pasan las generaciones. Esta población deberá tener un mecanismo de control de tamaño y diversidad, ya que se irá enriqueciendo al paso del tiempo. Además, en cada generación, una fracción de esta población –escogida mediante un método dado– pasa a formar parte de la población de esa generación [ZIT99].

Ocasionalmente [OBA98], ambos métodos se han aplicado en conjunto.

De la serie de implementaciones propuestas, se presenta un resumen escrupulosamente clasificado en [VAN99]. A partir de este trabajo es posible identificar a las propuestas que han marcado más fuertemente el desarrollo del área y que pueden denominarse en la categoría de "más usualmente imitados". En particular, hoy día, existen dos hitos demarcados por el desarrollo del *NSGA* (Non-dominated Sorting Genetic Algorithm) [SRI94] y del *SPEA* (Strength Pareto Evolutionary Algorithm) [ZIT99]. Ambos algoritmos fueron sometidos a pruebas exhaustivas con problemas de test cuidadosamente seleccionados [SRI94, ZIT99] y han demostrado su capacidad de obtener buenos resultados.

Este capítulo se ocupa de la descripción detallada de ambas aproximaciones. En la sección 3.2 se describe al *NSGA* con sus técnicas propias, se incluye también la propuesta de paralelización del mismo y además se presenta una variante del *NSGA* que incorpora una población externa como forma de archivar las buenas soluciones halladas de modo que el flujo posterior del algoritmo no las pierda. Esta variante es propia del presente trabajo y se ha sugerido como concepto para mejorar el desempeño general del algoritmo. En la sección 3.3 se introduce al *SPEA* con su método de paralelización escogido.

3.2 *Non-dominated Sorting Genetic Algorithm* o *NSGA*

Es un algoritmo genético basado en no dominancia, propuesto por Srinivas y Deb en [SRI94] y posteriormente verificado en [DEB99a]. Tiene una segunda versión propuesta en [DEB00] pero que debido a su aparición muy reciente apenas empieza a conocerse y aplicarse. Un resumen de aplicaciones del mismo a diferentes problemas de test se puede hallar en [VAN99]. Entre las principales características del algoritmo se pueden citar:

a) Convergencia rápida de la población hacia las regiones no dominadas.

b) El procedimiento para compartir *fitness* (*fitness sharing* [GOL87]) ayuda a distribuir las soluciones encontradas a lo largo de todo el ámbito de búsqueda.

c) No tiene límites en cuanto a la cantidad de funciones objetivo a optimizar.

d) Puede lidiar tanto con problemas de maximización como con problemas de minimización, sencillamente cambiando la forma de determinar la dominancia entre los individuos.

La idea detrás de *NSGA* es la utilización de un método de selección basado en rangos que se asignan según las relaciones de dominancia entre los individuos. De esta manera, se da prioridad en la selección a los individuos dominantes, mientras que los dominados tendrán menores posibilidades de tener descendencia. La asignación de individuos a cada rango, por tanto, se realiza según la cantidad de soluciones que cada individuo domina.

Además de la selección basada en rangos, se aplica un método para preservar la diversidad genética de la población (*fitness sharing*). Así se pretende lograr que los individuos obtenidos al final de los ciclos generacionales sean bien diferenciados entre sí.

NSGA difiere de un algoritmo genético tradicional solo en la manera en que se realiza la selección. Los operadores de cruzamiento y mutación permanecen inalterados. El cruzamiento se realiza con una probabilidad $p_c = 1$. Mientras que la mutación escoge un porcentaje m% de los hijos, y somete a cada allele del cromosoma de éstos una probabilidad de mutación de r_m.

Antes de realizar la selección se debe asignar a cada individuo un nivel de aptitud o *fitness*, según su rango o nivel de dominancia. Para esto, primero es necesario establecer un procedimiento sistemático para determinar la dominancia entre individuos y los rangos correspondientes a los mismos.

3.2.1 Procedimiento para establecer la dominancia entre individuos

Considérese un conjunto de N miembros de una población P; $P \subset \boldsymbol{X}_f$, cada uno de los cuales tiene k (> 1) valores para la función objetivo. El procedimiento expresado en el pseudocódigo 3.1 puede usarse para hallar el conjunto de soluciones no dominadas. Se trata de recorrer la población de individuos P y comparar a cada individuo de P con todos los demás. Si alguno domina al individuo que se está examinando, tal individuo recibe una marca de dominado. Los que no recibieron tal marca al final del procedimiento conforman el conjunto de soluciones no dominadas de la población P.

```
Procedimiento MarcarSolucionesDominadas(P)
Inicio
        Desde i = 1 hasta i = N – 1
                Si x_i es no dominada
                        Desde j = i + 1 hasta j = N
                                Si  x_j domina a x_i
                                        Marcar x_i como dominado
                                        j = N
                                Fin Si
                                Si x_i domina a x_j
                                        Marcar a x_j como dominado
                                Fin Si
                        Fin Desde
                Fin Si
        Fin Desde
Fin
```

Pseudocódigo 3.1. Marcar soluciones dominadas

Para asignar el rango a cada individuo de la población primeramente se hallan a todas las soluciones dominantes de la población, usando el procedimiento descrito en el pseudocódigo 3.1. Todas las soluciones dominantes pertenecen al primer rango y se las ignora momentáneamente para seguir con el algoritmo. Entre las soluciones restantes, nuevamente se hallan las dominantes usando el mismo procedimiento; a éstas se asigna el rango 2. Tal procedimiento sigue hasta clasificar en rangos a todos los individuos de la población.

De esta manera, todas las soluciones que no están marcadas como “dominadas” constituyen un mismo rango en cada etapa de la clasificación. Además, las soluciones de rango menor representan mejores soluciones al problema.

El *NSGA* original no implementa en modo alguno el concepto de elitismo. Las soluciones finales están dadas por el conjunto de soluciones dominantes que se hallan en la última generación del algoritmo. Sin embargo, tal proceder no resulta del todo adecuado, ya que en el curso del ciclo generacional se pierden soluciones muy buenas, y no necesariamente se reemplazan por otras mejores. De esta manera, y debido también al método utilizado para preservar la diversidad genética, el algoritmo no otorga un conjunto con muchos miembros del conjunto Pareto real y además tiende a acabar con una población muy homogeneizada que no representa buenas soluciones en términos de optimalidad Pareto, i.e. no se hallan bien distribuidos en el espacio de soluciones posibles.

Para salvar este inconveniente, en este trabajo se proponen dos versiones del mismo: una versión ligeramente diferente al *NSGA* original con la incorporación de una población externa que sirve como archivo de buenas soluciones, y la implementación del *NSGA* tradicional. En el *NSGA* con población externa se propone que, en cada generación, las soluciones dominantes del primer rango o nivel de dominancia, se copien a una población externa, que se ha creado como una forma de implementación de elitismo. Esta población externa no participa del ciclo normal del algoritmo y no afecta mayormente al tiempo de ejecución del mismo. Si bien se añade

tiempo de cómputo, el mismo es solo para mantener consistente la población externa. Sin embargo, las ganancias reportadas son grandes, según se aprecia en los resultados experimentales presentados en capítulos posteriores. Además, el *NSGA* con población externa y mutación, cumple con la hipótesis del teorema para la convergencia de los algoritmos genéticos multiobjetivo [VAN99].

3.2.2 Asignación de *fitness* en el *NSGA*

Como se apuntó en capítulos previos, el potencial reproductivo de un individuo está fuertemente ligado a su nivel de aptitud o *fitness*. A mayores valores de *fitness* mayor posibilidad de sobrevivencia y posibilidad de generar descendientes. Por ende, el valor de *fitness* asignado a un individuo es crucial para el desempeño del algoritmo.

El procedimiento de asignación de *fitness* –que se puede citar entre las particularidades de este algoritmo– se detalla en esta sección. Primero se aplica el procedimiento explicado para establecer la dominancia entre los individuos, marcar las soluciones dominadas y asignar un rango a los individuos. Todas las soluciones no dominadas (por ende: no marcadas) constituyen el primer conjunto de aproximación al conjunto Pareto óptimo (primer nivel de dominancia identificado), y se les asigna un valor elevado de *fitness* (se les asigna el valor *N*). El mismo valor de *fitness* se asigna a todas, para dar el mismo potencial reproductivo a todas las soluciones de similar nivel de dominancia. Se dice que éstas soluciones son de rango 1.

Para mantener la diversidad en la población, éstas soluciones no dominadas pasan por un proceso para compartir sus valores de *fitness*. El mecanismo por el cual se realiza esto se describe más adelante, por ahora basta con especificar que el valor del *fitness* se divide entre una cantidad (denominada contador del nicho) proporcional al número de individuos de la población que tienen una cercanía o similitud grande con el individuo en cuestión. Dependiendo del problema en tratamiento, el contador de nicho puede fijarse en la similitud entre los individuos en el ámbito de los genotipos o en el de los fenotipos. En el primer caso se está distribuyendo la población de modo que los individuos resultantes tengan codificaciones bien diferenciadas unos de otros. En el segundo caso, la distribución permite que los individuos resultantes sean bien diferentes, sin importar la codificación de los mismos. En optimización multiobjetivo, la distancia entre dos individuos se puede computar como la distancia euclidiana de los mismos en el espacio objetivo; de este modo se pretende distribuir a los individuos en todo el espacio objetivo y obtener soluciones en todo el frente Pareto óptimo.

Luego de compartir el *fitness* de los individuos similares pertenecientes a este primer nivel de dominancia, se los ignora temporalmente para pasar a procesar a los demás miembros de la población. El mismo procedimiento, arriba descrito, se utiliza para hallar un segundo nivel de dominancia entre los individuos restantes. Una vez que se identifican las soluciones

correspondientes a este segundo grupo o rango, un valor de *fitness*, ligeramente inferior al peor *fitness* asignado a la población dominante de nivel 1, se asigna a todos los individuos de este segundo grupo. Nuevamente, se aplica el concepto de compartir *fitness* entre miembros similares. Este proceso iterativo continúa hasta que se haya asignado un *fitness* adecuado a cada miembro de la población.

El proceso se encuentra descrito en el siguiente pseudocódigo:

```
Procedimiento para Asignar Fitness (P)
Inicio
        RanMax = 0
        Desde i = 1 hasta i = N
                Calcular rango r_i del individuo x_i
                Si RanMax < r_i
                        RanMax = r_i
                Fin Si
        Fin Desde
        FitVir = N
        RanAc = 1
        Desde i = 1 hasta i = RanMax
                Asignar FitVir como valor de fitness virtual a los miembros de P que tienen rango i
                Compartir fitness virtual entre los individuos de rango i; para obtener los valores de
                        fitness reales
                FitVir = Mínimo entre todos los fitness reales asignados - ε_niv
        Fin Desde
Fin
```

Pseudocódigo 3.2. Asignar *fitness* a los miembros de una población *P*

Posteriormente se realiza el proceso de selección usando el sistema de selección proporcional al *fitness* o sistema de ruleta [GOL89]. De esta manera, los individuos del primer grupo tienen mayor probabilidad de ser escogidos que los de los grupos posteriores. Esto ayuda a orientar la búsqueda hacia regiones no dominadas.

3.2.3 Procedimiento para compartir el *fitness* (*fitness sharing*) *FS*

El procedimiento de *fitness sharing* es el utilizado para mantener la diversidad genética. Este procedimiento está inspirado en las ideas de Holland [HOL75] sobre formación de especies y nichos dentro de ciertos ecosistemas, mediante la distribución de recursos disponibles. Fue esquematizado por Goldberg en [GOL89].

Se considera que el *fitness* es un recurso general que pertenece a todos los individuos de un nicho y por tanto puede compartirse entre ellos.

Un nicho está conformado por individuos genéticamente parecidos o de "la misma especie". En el proceso de *fitness sharing* cada individuo en un nicho comparte su aptitud o *fitness* con todos los demás en el mismo nicho. De este modo los nichos con mayores valores de *fitness* son capaces de mantener un mayor número de individuos.

Por otro lado, como el *fitness* se comparte entre todos los individuos del nicho, éstos tienen, en particular, un menor valor de *fitness* del que tendrían si no se utilizase el *fitness sharing*. De este modo se mantiene el equilibrio natural y un nicho no tiende a mantener una cantidad de individuos superior a la que puede soportar.

Resumiendo, *fitness sharing* usa el concepto de compartir el *fitness*, y el proceso de selección natural para afectar a la diversidad de la población general. Por tanto requiere no solo alguna manera de medir la similitud genética (distancia) entre los individuos para indicar qué individuos pertenecen a qué nichos, sino también una distancia conocida como radio del nicho (σ_{share}), que efectivamente es una distancia umbral para definir los límites de cada nicho.

El valor de *fitness* real de un individuo $\boldsymbol{i} \in \boldsymbol{X}$ ($F'(\boldsymbol{i})$) se computa a partir de su *fitness* original ($F(\boldsymbol{i})$), dividido entre su contador de nicho cn_i. El contador de nicho estima la cantidad de individuos que pertenecen a un mismo nicho. Este valor se obtiene sumando valores indicados por una "*sharing function*" que recibe como parámetros al individuo y a todos los restantes individuos de la población. Si dos elementos de la población son idénticos, la "*sharing function*" retorna 1. Si dos elementos de la población exceden una distancia umbral (llamada radio del nicho), la "*sharing function*" retorna 0, indicando que los individuos pertenecen a nichos diferentes y por tanto no comparten sus valores de *fitness*. En los otros casos, la "*sharing function*" retorna un valor intermedio entre 0 y 1.

Una "*sharing function*" comúnmente utilizada es la siguiente

$$Sh(d_{ij}) = 1 - \left(\frac{d_{ij}}{\sigma_{share}} \right)^{\alpha} \qquad \text{si } d_{ij} \leq \sigma_{share} \qquad \text{o} \tag{3.1}$$

$$Sh(d_{ij}) = 0 \qquad \text{en otro caso}$$

El parámetro α controla la forma de la "*sharing function*" (la función es lineal para α = 1). El valor típico utilizado es α = 2.

Luego, el *fitness* degradado de un individuo $\boldsymbol{i}$ está dado por:

$$F'(i) = \frac{F(i)}{cn_i} \tag{3.2}$$

donde cn_i es el contador de nicho para el individuo $\boldsymbol{i} \in \boldsymbol{X}$ (población de soluciones); el mismo está dado por:

$$cn_i = \sum_{j=1}^{|X|} Sh(d_{ij}) \quad \textbf{(3.3)}$$

y la distancia d_{ij} es la distancia euclidiana entre los individuos ***i*** y ***j*** $\in$ ***X,*** computada considerando los valores de los diferentes objetivos que se desea optimizar.

La distancia d_{ij} también puede ser la distancia de *Hamming* entre las mismas soluciones. El hecho de optar por uno u otro método para computar la distancia dependerá del problema en cuestión, aunque –según se expresó previamente– en optimización multiobjetivo se busca distribuir todas las soluciones en todo el frente Pareto óptimo, por lo que casi siempre se computan las distancias a nivel del espacio objetivo.

Al momento de asignar los *fitness* reales a las soluciones presentes en la población, se procede por rangos o niveles de dominancia y no sobre toda la población. De esta manera, a las soluciones de rango 1, se asigna como *fitness* original el tamaño de la población ***N,*** y este *fitness* se comparte usando el procedimiento descrito entre todos los demás miembros del mismo rango. Luego se halla el valor mínimo de *fitness* real asignado a los individuos de rango 1. A este valor se sustrae un pequeño valor positivo ε y de esta manera se obtiene el siguiente valor de *fitness* original para los individuos de rango 2. Este procedimiento se repite por cada nivel e individuo de la población, hasta que todas las soluciones tienen su valor de *fitness* real asignado.

Una de las principales dificultades que presenta el *fitness sharing* es la determinación del parámetro σ_{share} o radio del nicho. En tal sentido, se pueden hallar valores empíricos sugeridos por estudios realizados sobre varias funciones multimodales en [DEB94]. Por otro lado, Deb también afirma [DEB94] que el *NSGA* no es muy sensible a este parámetro.

Existen numerosos procedimientos más para inducir a la formación de nichos en las poblaciones de los algoritmos genéticos. Una descripción interesante se puede hallar en [SAR98]. El método de *fitness sharing* se ha escogido por ser el de aplicación más extendida y porque quienes definieron el *NSGA* consideraron adecuado utilizarlo en su versión del algoritmo.

3.2.4 Otros métodos para inducir a la formación de nichos

Según se demuestra en los últimos trabajos de investigación en esta área [DEB00], el *fitness sharing* está dejándose a un lado para probar otras técnicas más eficaces como el *crowding* [DEJ95].

El *crowding* fue inicialmente introducido por De Jong. En esta formulación se escoge una cantidad fija de elementos (igual al tamaño fijo de la población) mediante selección proporcional al *fitness*, con ellos se crea un igual número de hijos. Por cada hijo, se toma una

muestra de la población actual, de tamaño indicado por un factor de *crowding*, que debe determinarse previamente. Los hijos, luego, reemplazan a los individuos de esta muestra con los que tienen mayor similitud o menos distancia. Para computar la similitud De Jong utilizaba la distancia de *Hamming*. Valores típicos del factor de *crowding* son 2 o 3, e incluso 10% del tamaño de la población actual.

Sin embargo, para el objetivo de mantener soluciones múltiples en la población, se ha demostrado [SAR98] que este sistema tiene apenas un uso bastante limitado, por lo que se ha dejado de lado por bastante tiempo.

Debido a esto, Mahfoud ha propuesto una variación del método, conocido como *crowding* determinístico [MAH95].

En este nuevo método primero se coloca en pares, de manera aleatoria, a todos los individuos de la población. Cada par genera dos hijos mediante la aplicación de los operadores genéticos. Cada hijo nuevamente compite con sus padres. Hay dos torneos padres-versus-hijos posibles, y los mismos se deciden en base a la distancias entre los padres y los hijos de ambas combinaciones posibles. El ganador de la competencia pasa a la siguiente generación.

Según [SAR98], el método propuesto por Mahfoud sobrepasa ampliamente al de De Jong en capacidad de mantener diversidad genética y agrupar a los miembros de la población en nichos, aún así no es completamente satisfactorio.

Otro método es el "*niching* secuencial". Este método es una variación del *fitness sharing* propuesta por Beasley, Bull y Martin [BEA93], pero no ha aún ampliamente evaluado.

Si bien en las últimas versiones del *NSGA* ya se utiliza el *crowding*, para la formulación original que se desea evaluar se había propuesto el *fitness sharing*. En consecuencia, como previamente se ha apuntado, el presente trabajo implementa *fitness sharing*.

3.2.5 Diagrama de Flujo del *NSGA*

En la figura 3.1 se presenta el diagrama de flujo del *NSGA* con población externa en su forma más sencilla.

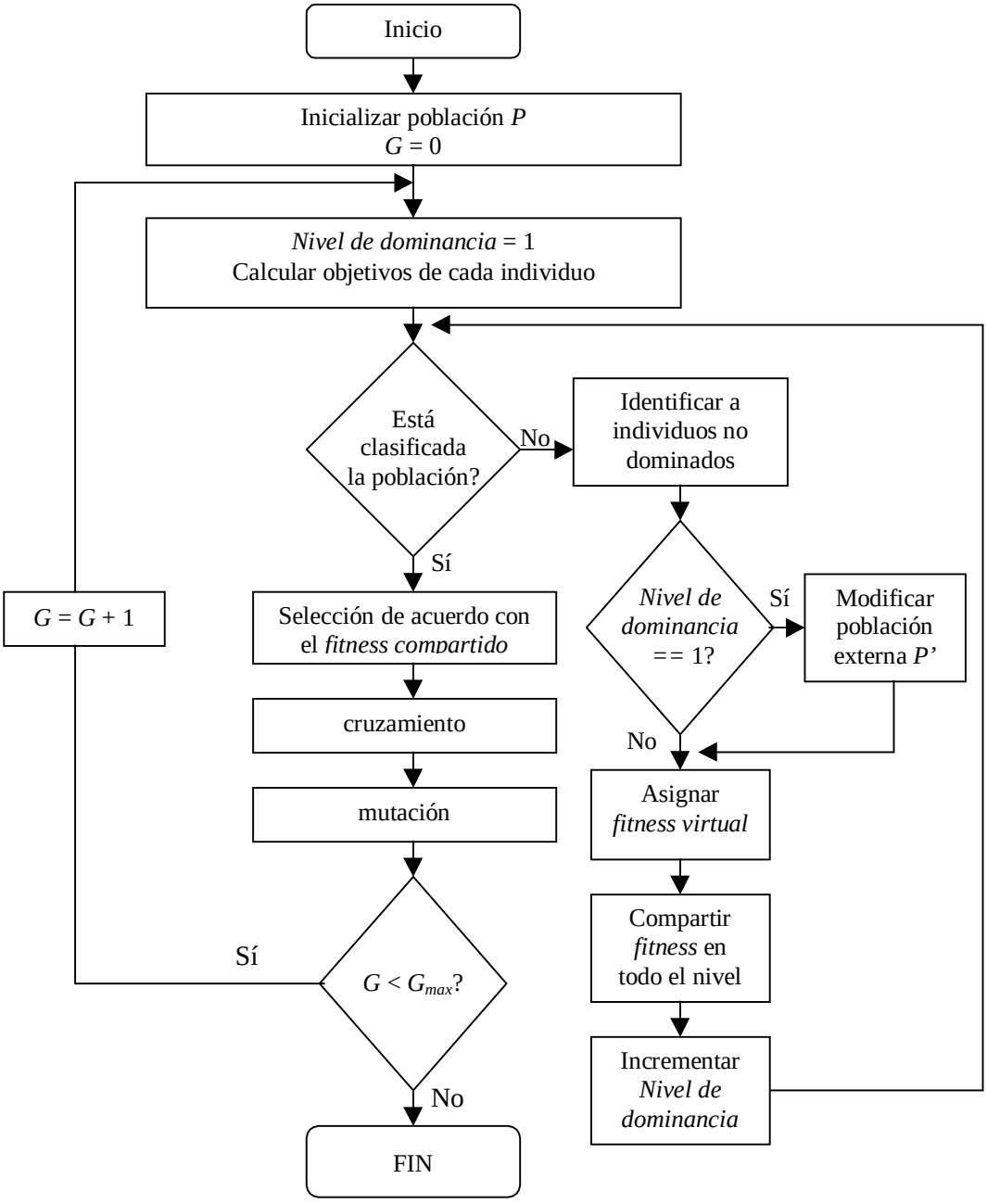


Figura 3.1. Diagrama de flujo de *NSGA*.

3.2.6 Esquema para la paralelización del *NSGA*

Debido a la gran cantidad de cómputo requerido para el cálculo de las funciones objetivo (que a su vez es necesario para determinar la dominancia entre individuos), y a que el orden de los algoritmos para la implementación del procedimiento de asignación de *fitness* es exponencial, las pruebas en redes de alrededor de 20 nodos pueden significar varias horas antes de obtener soluciones. En consecuencia, se ha decidido implementar una versión paralela del *NSGA*, manteniendo la existencia de una población externa en un caso, y sin ella para el segundo caso. Para el *NSGA* con población externa se ha seguido el esquema que se presenta a continuación.

Dado un sistema distribuido con H procesadores, en cada procesador h, $h \in \{1, .. , H\}$, se mantienen dos poblaciones $P_h(t)$ y $P'_h(t)$. La población $P_h(t)$ contiene a los miembros de la población generados por cruzamiento y mutación en la generación anterior $t-1$; mientras que $P'_h(t)$ contiene al conjunto de individuos dominantes encontrados desde el inicio del ciclo generacional, hasta la generación t.

Una vez identificados, los nuevos individuos, que se acoplan a las poblaciones $P'_h(t)$ de cada procesador h, en la generación t, se transmiten a los demás procesadores. Este procedimiento se conocerá con el nombre de envío y recepción de migrantes.

Considerando que se cuenta con H procesadores y que la población inicial es de N individuos para el algoritmo en su forma secuencial, el tamaño inicial de cada P_h es $\frac{N}{H}$, siempre y cuando los procesadores sean homogéneos, que ha sido el caso registrado para todos nuestros experimentos. Este tamaño crece con la recepción de migrantes, pero vuelve a ser el mismo luego de la aplicación de los operadores genéticos.

Sin embargo, el tamaño de las poblaciones externas de individuos dominantes P'_h está dado por N', que es el mismo tamaño de la población externa de individuos del algoritmo secuencial.

Para iniciar el procedimiento de paralelización se cuenta con un proceso maestro, que se ejecuta de manera independiente. Este proceso maestro debe disparar los procesos esclavos en todos los procesadores; además recolecta las señales de fin de cada proceso esclavo y los mata luego de obtener sus resultados. Cada proceso esclavo hace una transmisión, por difusión (*broadcasting*), de los nuevos individuos dominantes que identifique, de manera independiente a los demás procesos (asincronismo). Además, recibirá los migrantes antes de calcular el *fitness* de sus individuos. La migración se realiza con una frecuencia de una vez por generación computada. Cumplido el ciclo generacional (i.e. una vez alcanzada la máxima cantidad de generaciones g_{max} o una vez cumplido el criterio de parada), el proceso maestro recibe las poblaciones de los procesadores y arma con ellos el conjunto Pareto óptimo hallado. El cómputo real se realiza entonces en los procesos esclavos.

El pseudocódigo de un proceso esclavo es el siguiente:

```
Procedimiento NSGAEsclavos()
Inicio
        Leer parámetros iniciales para el algoritmo N/H, N', g_max, p_c, r_m, m%, σ_share
        Leer parámetros del problema: costos y confiabilidad de los enlaces
        Leer población inicial P generada aleatoriamente
        G = 1
        Mientras G < G_max y no se ha alcanzado aún la condición de parada
                Calcular valores de las funciones objetivo para todos los individuos en P
                Aplicar procedimiento para asignación de fitness
                Modificar la población externa P'
                Enviar todos los miembros nuevos de P' a los demás procesos
                Aplicar operadores evolutivos en la manera usual para generar una nueva población P
                Recibir migrantes y adjuntarlos a P
                G = G + 1
        Fin Mientras
        Informar al proceso coordinador que se ha terminando, enviando un flag
        Esperar la señal apropiada del coordinador para enviarle la población P'
        Esperar hasta recibir la señal de fin que enviará el coordinador
Fin
```

Pseudocodigo 3.3. Versión Paralela del NSGA. Procedimiento esclavo

Mientras que el pseudocódigo para el proceso maestro es:

```
Procedimiento NSGACoordinador()
Inicio
        Generar H nuevos procesos esclavos
        contador_señales = 0
        Mientras contador_señales < H
                Esperar hasta recibir señales de alguno de los H procesos
                Si una señal se recibe
                        Recoger resultados del proceso que envió la señal
                        Enviar al mismo proceso una señal de terminación
                        contador_señales = contador_señales + 1
                Fin Si
        Fin Mientras
        Hacer la operación de unión sobre los conjuntos obtenidos de todos los procesos
        Aplicar el concepto de dominancia Pareto sobre el conjunto resultante y obtener resultados
        Escribir resultados finales
Fin
```

Pseudocodigo 3.4. Versión Paralela del NSGA. Procedimiento Maestro

Este esquema de paralelización, muy parecido al que se presentará luego para el *SPEA*, es completamente diferente a otros sugeridos en la literatura para los *MOEAs*, ya que contempla la existencia de poblaciones independientes evolucionando en el tiempo a su propio ritmo e intercambiando información sobre buenos individuos cada tanto. Mientras que las otras propuestas más bien se basan en la paralelización de secciones del ciclo generacional, como cálculo de los objetivos o la asignación del *fitness*. Sin embargo, la presente propuesta ha demostrado ser tanto efectivo como eficiente, según se verá al discutir los resultados. Esta conclusión empírica es muy importante, ya que las otras propuestas no han sido probadas hasta

ahora en ningún tipo de problema. También se realizaron pruebas con otros esquemas similares, pero el presentado aquí es el más optimizado, por lo que no consideramos muy útil detenernos más en su discusión.

Para la paralelización del *NSGA* sin población externa el esquema utilizado ha sido idéntico, excepto que la elección de migrantes es directa, ya que en cada generación se envían a los demás procesadores a todos aquellos individuos que pertenecen al primer nivel de dominancia. Nuevamente se reciben todos los migrantes que llegan de los demás procesos, y se adjunta a los mismos a la población corriente. El tamaño de la población se mantiene nuevamente mediante la aplicación posterior de los operadores genéticos.

3.3 *Strength Pareto Evolutionary Algorithm* o *SPEA*

El segundo algoritmo escogido para su implementación es el *SPEA* propuesto por Zitzler et al. en [ZIT99]. Este algoritmo, debido a su novedad, ha sido aplicado a muy pocos problemas de prueba, aunque representa uno de los *MOEAs* más promisorios, ya que su planteamiento cumple con la hipótesis del teorema de convergencia de algoritmos evolutivos presentado en [VAN99].

A pesar de su novedad, ya se han identificado algunos puntos débiles en su formulación [ZIT01] y ya se ha presentado una segunda versión del mismo –SPEA2– [ZIT01]. Sin embargo esta segunda versión es de aparición muy reciente y no se ha implementado debido a que la fase experimental del presente trabajo había sido culminada ya al recibir las primeras noticias a este respecto.

SPEA usa una mezcla de técnicas previamente usadas y otras recién establecidas para encontrar soluciones Pareto óptimas diferentes y en paralelo. Las características que comparte con otros algoritmos previamente propuestos son:

a) guarda las soluciones no dominadas que ha encontrado hasta el momento en una población externa, de esta manera implementa elitismo;
b) usa el concepto de dominancia Pareto, para asignar un valor de fitness a los individuos;
c) realiza *clustering* [MOR80] para reducir el número de soluciones no dominadas que tiene almacenadas, sin destruir las características del frente Pareto delineado hasta el momento.

Por otro lado, *SPEA* presenta, de manera exclusiva, los siguientes conceptos:

a) combina las tres características arriba mencionadas en un único algoritmos (hasta ahora ellas se habían propuesto siempre aisladas unas de otras);
b) el *fitness* de un individuo está determinado a partir de las soluciones que se encuentran en la población de individuos no dominados solamente; de este modo, no es relevante que los miembros de la población general se dominen unos a otros;

c) todas las soluciones que forman parte del conjunto externo de soluciones no dominadas participan en la selección;

d) se sugiere un nuevo método para inducir a la formación de nichos dentro de la población y así preservar la diversidad genética; este método está basado en el concepto de Pareto y no requiere el conocimiento o determinación previa de algún parámetro de distancia (como es el radio del nicho en *FS*).

En la siguiente sub-sección se halla un diagrama de flujo del *SPEA* (figura 3.2), indicando los pasos fundamentales del algoritmo. Esencialmente, la diferencia entre el *SPEA* y el *GA* tradicional radica en la asignación de *fitness* y el mantenimiento de la población externa de dominantes.

Los operadores genéticos son los tradicionales. Como operador de selección se utilizan torneos binarios, mientras que el cruzamiento es el de un punto con probabilidad $p_c = 1$. El operador de mutación implementado en este trabajo toma un porcentaje de la población actual (m%) y cambia cada *allele* del cromosoma de estos individuos con una probabilidad previamente establecida r_m.

En la figura, *N'* representa un tamaño máximo de la población externa. Este valor se define previamente y sirve además para la decisión de la aplicación del proceso de *clustering*, que permite controlar el tamaño del a población externa.

La selección se realiza en toda la unión de los dos conjuntos *P* y *P'*, hasta que se obtiene la cantidad de padres deseada, en este caso esta cantidad es *N* o el tamaño de la población corriente.

El procedimiento de asignación de *fitness*, así como la reducción del conjunto de dominados utilizando *clustering* se especifican más adelante, en las siguientes secciones.

El proceso descrito como inicialización de la población *P* se refiere a la lectura de la población inicial aleatoriamente generada.

Los mismos criterios de parada que se aplican al *NSGA* (cantidad máxima de generaciones g_{max} y cambios en el conjunto de soluciones identificadas) se utilizan también para el *SPEA*.

3.3.1 Diagrama de Flujo del *SPEA*

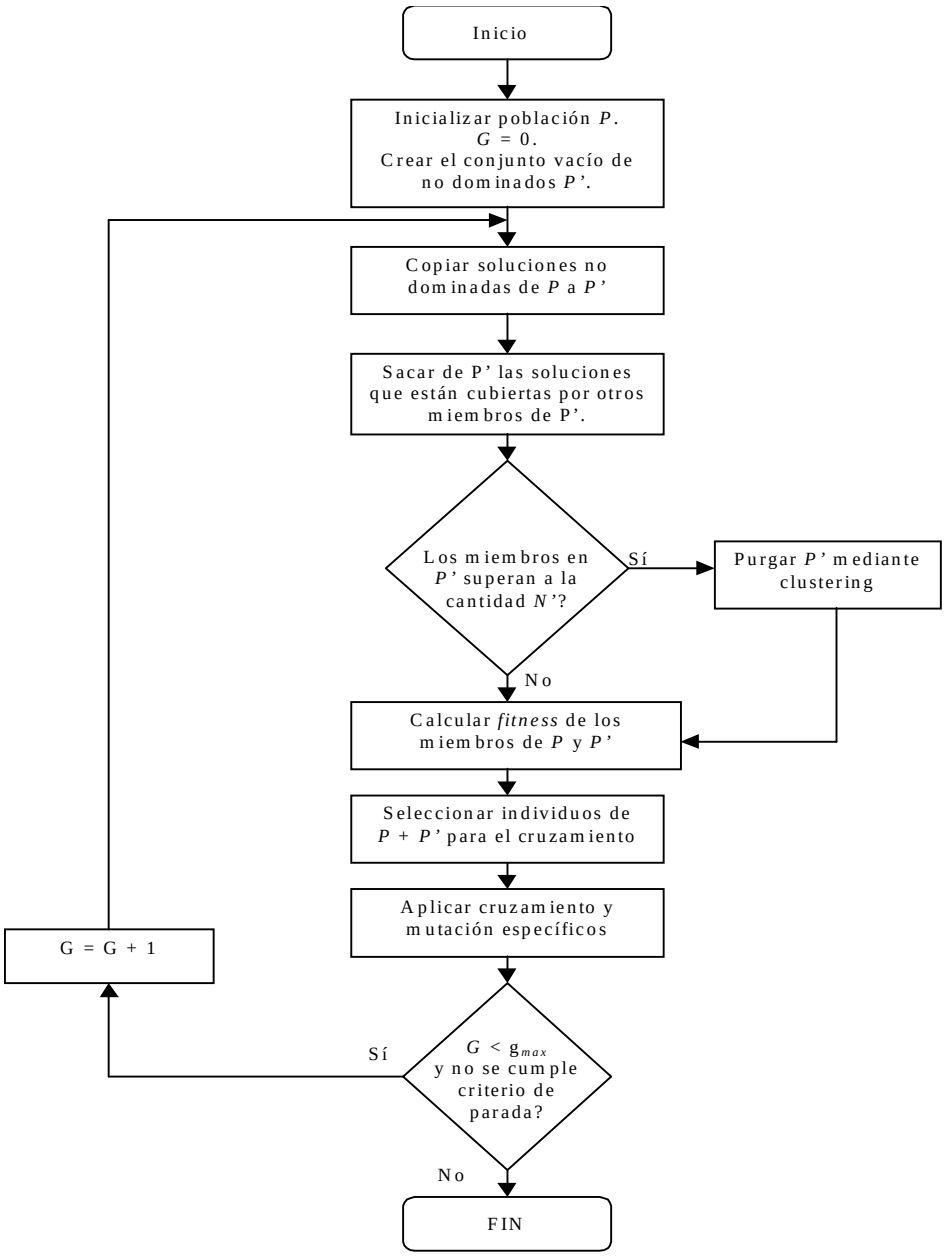


Figura 3.2. Diagrama de Flujo del *SPEA*.

3.3.2 Procedimiento para asignar *fitness* a los individuos

Una de las particularidades del *SPEA* es su forma de asignar *fitness* a cada individuo. El proceso se realiza de modo a lograr que los individuos dominantes y más representativos tengan mejores valores de *fitness*. A la vez, el proceso está pensado para inducir a mantener la diversidad genética y obtener soluciones distribuidas en todo el frente Pareto.

Para ello, el procedimiento de asignación de *fitness* se realiza en dos fases. Primero, se asigna un rango a los individuos del conjunto *P'* (conjunto externo de no dominados). Luego, se evalúan a los miembros de *P* (conjunto de soluciones generales).

Para asignar *fitness* a los miembros del conjunto de dominantes, a cada solución $\boldsymbol{i} \in P'$ se asigna un valor real $s_i \in [0,1)$, llamado la fuerza o *strength*[2]; s_i es proporcional al número de individuos $\boldsymbol{j} \in P$ para los cuales $\boldsymbol{i} \succ \boldsymbol{j}$ (es decir, es una medida de la cantidad de individuos dominados por ***i***). Sea *n* el número de individuos en *P* que están dominados por ***i***, y asumamos que *N* es el tamaño de *P*. Luego s_i se define como $s_i = \frac{n}{N+1}$. El *fitness* de ***i*** es igual a la inversa de su *strength*: $F(\boldsymbol{i}) = \frac{1}{s_i}$

El *strength* de un individuo $\boldsymbol{j} \in P$ se calcula como la suma de los *strengths* de todas las soluciones externas no dominadas $\boldsymbol{i} \in P'$ que dominan a ***j***. Se suma 1 a ese total para asegurar que los miembros de *P'* tengan mejor *strength* que los miembros de *P*. Así el *fitness* del individuo ***j*** se expresa como $f_j = \frac{1}{1+\sum_{i,i\succ j} s_i}$; donde $s_j \in [1, N)$.

La idea detrás de este mecanismo es preferir siempre los individuos que están más cerca del frente Pareto óptimo y al mismo tiempo, distribuirlos en toda la superficie factible. La principal diferencia con *fitness sharing* es que los nichos se definen en términos de la dominancia Pareto, y no en términos de distancia. Esto es más adecuado ya que en muchos problemas del mundo real la "distancia" no tiene un significado práctico si se computa en el espacio objetivo, ya que cada objetivo puede estar expresado en magnitudes totalmente diferentes y no comparables a las de los demás (millones de dólares, tiempo, metros, etc).

[2] Este término, según los creadores del algoritmo, se tomó de [ZIT98a], donde fue introducido en el contexto de sistemas clasificadores. Se refiere a una cantidad que resume la utilidad de una regla. Aquí refleja la utilidad de una solución no dominada.

3.3.3 Reducción del conjunto de No Dominados utilizando *clustering*

En algunos problemas, el conjunto Pareto óptimo puede ser muy grande o contener incluso un número infinito de soluciones. Sin embargo, desde el punto de vista del diseñador, obtener todas las soluciones no dominadas encontradas no agrega ventajas, una vez que se alcanzó un límite razonable.

Más aún, el tamaño de la población externa de no dominados altera el comportamiento del *SPEA* [ZIT99]. Esto es evidente si se observa con más detenimiento el procedimiento de asignación de *fitness* que el algoritmo utiliza. Cuando el conjunto externo de dominantes crece por encima de un límite impuesto, el algoritmo ya no encuentra buenas soluciones y además tiene mayor probabilidad de estancarse en un mínimo local. Por estas razones, purgar el conjunto externo de dominantes no solo puede ser necesario sino incluso obligatorio.

Un método que se ha aplicado a este problema con éxito y se ha estudiado extensivamente en el mismo contexto es el análisis de *cluster* [MOR80].

En general, el análisis de *cluster*, particiona una colección de *p* elementos en *q* grupos de elementos homogéneos, donde $q < p$. De cada grupo escoge luego un elemento representativo o centroide del *cluster*.

El método de enlace promedio (*average linkage method*) [MOR80], ha demostrado un comportamiento adecuado en este problema y es el escogido en [ZIT99] para el *SPEA*. Por ello, es también el método que se utiliza en las implementaciones descritas posteriormente, para controlar el tamaño de la población externa del *SPEA* evitando perder soluciones buenas y significativas ya identificadas. El procedimiento se expresa en el pseudocódigo siguiente:

Procedimiento Clustering()
Inicio
    Inicializar el conjunto de *clusters* C; cada individuo ***i*** $\in$ *P'* constituye un *cluster* distinto. Así: $C = \bigcup_i \{i\}$.
    Mientras C > *N'*
        Calcular la distancia de todos los posibles pares de *clusters*. La distancia d de 2 *clusters* c_1 y $c_2 \in C$ es la distancia promedio entre pares de individuos de los 2 *clusters* $d = \frac{1}{|c_1| \cdot |c_2|} \cdot \sum_{i_1 \in c_1, i_2 \in c_2} \| i_1 - i_2 \|$; donde la métrica $\| \ \|$ refleja la distancia entre 2 individuos i_1 e i_2 (en este estudio se usa una métrica Euclidiana sobre el espacio objetivo).
        Determinar dos *clusters* c_1 y c_2 con distancia mínima *d*; los *clusters* escogidos se unen para conformar uno solo.
    Fin Mientras
    Computar el conjunto no dominado reducido, seleccionando un individuo representativo de cada *cluster*. Se considera como el centroide (el punto con distancia promedio mínima con respecto a todos los otros puntos del *cluster*) como la solución representativa.
Fin

Pseudocodigo 3.5. Procedimiento de clustering

3.3.4 Esquema para la paralelización del *SPEA*

El esquema propuesto para la paralelización del *SPEA* es idéntico al propuesto para el *NSGA* por lo que no se presentará una discusión detallada del mismo. Más bien se indica directamente los algoritmos correspondientes al proceso organizador o maestro y al proceso esclavo.

```
Procedimiento SPEACoordinador()
Begin.
        Generar H nuevo procesos esclavos
        contador_señales = 0
        Mientras contador_señales < H
                Esperar señales de cualquiera de los H procesos
                Si se recibe una señal
                        Obtener los resultados del proceso que envío la señal
                        Enviar al mismo proceso una señal de parada
                        contador_señales = contador_señales + 1
                Fin Si
        Fin Mientras
        Hallar la union de todos los conjuntos obtenidos de todos los procesadores
        Aplicar concepto de dominancia sobre el conjunto resultante para obtener resultados finales
        Emitir resultados finales
Fin
```

Pseudocodigo 3.6. Version Paralela del SPEA. Procedimiento maestro

```
Procedimiento SPEAEsclavo()
Inicio
        Leer parámetros de entrada para los procesos: s, g_max, p_c, r_m, m%
        Leer parámetros del problema: costos y confiabilidad de enlaces
        Leer población inicial aleatoriamente generada P
        G = 1
        Mientras no se alcance la condición de parada y  < G_max
                Calcular los valores de cada objetivo para cada individuo
                Recibir migrantes desde otros procesos y agregarlos a la población actual P
                Encontrar individuos no dominados en la población actual P
                Modificar la población externa de no dominados P'
                Transmitir selectivamente las nuevas soluciones de P' a los demás procesos
                Si el número de soluciones en la población externa excede un máximo de N'
                        Purgar P' mediante clustering
                Fin Si
                Calcular el fitness de cada individuo en P y P'
                Elegir individuos de la union de P + P' hasta tener suficientes padres
                Aplicar operadores de cruzamiento y mutación para generar el nuevo conjunto P
                G = G + 1
        Fin Mientras
        Informar al coordinador que el proceso ha terminado enviando una señal
        Enviar los individuos desde P' al coordinador
        Esperar la señal de parada que envía el coordinador
Fin
```

Pseudocodigo 3.7. Versión Paralela del NSGA. Procedimiento esclavo.

3.4 Sumario

En el presente capítulo se ha presentado una discusión detallada de los algoritmos que se utilizan en las implementaciones, incluyendo las versiones paralelas. Se ha visto que cada algoritmo tiene sus propias técnicas y características y, por tanto, permiten evaluar los conceptos claves en el diseño de *MOEA*s, como son: el elitismo, la posibilidad de mantener la diversidad genética y la posibilidad de distribuir las soluciones halladas en todo el frente Pareto.

En el capítulo siguiente se discutirá la metodología a utilizarse para el diseño de los experimentos con los *MOEA*s descritos.

4 Diseño de Experimentos

4.1 Diseño de los experimentos con *MOEA*s

En el presente capítulo presentamos la metodología escogida y aplicada para el diseño del ámbito experimental del presente proyecto.

El diseño cuidadoso de los experimentos se realizó a partir de guías presentadas por Barr [BARR95] y Jackson [JAC91]. En estos artículos se discute el diseño de experimentos con métodos heurísticos y se dan algunas sugerencias para reportar resultados y asegurar que los mismos sean reproducibles. Específicamente sugieren seguir los siguientes pasos:

a) definir los objetivos experimentales;
b) escoger las métricas para medir el desempeño;
c) diseñar y ejecutar los experimentos;
d) analizar los datos y obtener las conclusiones; y
e) reportar los resultados experimentales.

Los mismos autores también indican que las métricas pueden categorizarse de la siguiente manera:

a) métricas de eficiencia: miden el esfuerzo computacional para obtener los resultados, por ejemplo tiempo de *CPU*, número de evaluaciones, iteraciones, tiempo, etc.;
b) métricas de efectividad: miden que tanto se ajustan las soluciones obtenidas a las soluciones correctas;
c) métricas de robustez: miden la capacidad del código para recuperarse de situaciones imprevistas en la entrada;
d) métricas de confiabilidad: miden la amplitud de tipos de problemas que el mismo código puede resolver; y
e) métricas de facilidad de uso: miden la cantidad de esfuerzo requerido para usar el software).

En especial, para el caso particular estamos interesados en las métricas de las categorías (a) y (b). Nótese que ninguna métrica única puede escogerse para lograr una medida adecuada del desempeño de un optimizador multiobjetivo, ya que determinar el nivel de desempeño de estos optimizadores es, a su vez, un problema multiobjetivo, porque se requiere el cumplimiento de varios objetivos (buen tiempo de ejecución, uso adecuado de recursos, buena aproximación al conjunto Pareto óptimo real, buena distribución de las soluciones halladas en todo el conjunto Pareto óptimo real, etc.). Aunque no se aplicarán métricas de las demás categorías, ello no implica que no se han considerado esos aspectos al momento del diseño de los programas.

Siguiendo las sugerencias apuntadas, primero se establecerán los objetivos de los experimentos en la sección 4.2. La metodología se presenta en la sección 4.3. Las métricas de desempeño se describen en la sección 4.4. Mientras que los resultados experimentales, –

obtenidos a partir de un problema de diseño real–, y el análisis posterior se presentan en los siguientes capítulos.

4.2 Motivación y objetivos de los experimentos

El principal objetivo de los experimentos es comparar el desempeño de dos *MOEA*s, tanto en sus versiones secuenciales como paralelas, en términos de efectividad y eficiencia, para el problema de prueba previamente establecido, ya publicado en [DEE98]. A partir de aquí, debería ser posible deducir la factibilidad de una utilización a mayor escala.

No se pretende demostrar que los *MOEA*s propuestos son las únicas herramientas que pueden resolver el problema del diseño de *backbones* de redes, sino sugerir que los *MOEAs* son herramientas con gran potencial para enfrentarlo. Además, se desea observar cuál de los *MOEAs* se comporta "mejor" a los demás en este problema, o en su defecto, determinar el porqué. Si todos los *MOEA*s se comportan de igual manera también se desea establecer las razones de ello, ya que esto conduciría a argumentar que la elección de una implementación particular no es un asunto crucial. También se pretende establecer el nivel de efectividad y eficiencia agregado por la paralelización de los algoritmos.

Los algoritmos secuenciales seleccionados son el *SPEA* y el *NSGA*, discutidos en detalle en el capítulo tres. El *NSGA* se ha implementado en sus dos versiones descritas: con población externa y sin población externa. Ambos son algoritmos promisorios e innovadores que deberían, –según el marco teórico en que fueron presentados–, resultar en un buen desempeño. Agregamos a la comparación la versión paralela propuesta para esos algoritmos sugerida también en el capítulo tres.

4.3 Metodología

Habiendo discutido a los *MOP*s y a los *MOEA*s en los capítulos previos podemos empezar a diseñar experimentos significativos sobre los mismos. En este punto conviene una aclaración: a pesar de que las métricas que podamos escoger conforman una base común sobre la cual comparar el desempeño de los diferentes *MOEA*s, los resultados no pueden alcanzar un nivel general a menos que se conozca el óptimo global, que es el que a su vez deseamos hallar con el uso de *MOEA*s.

Para el caso en estudio, se ha generado un conjunto aproximación al conjunto de soluciones Pareto óptimas, mediante la unión de todos los conjuntos Pareto obtenidos en todos los experimentos conducidos.

Los algoritmos implementados para las pruebas son:

- *NSGA1a*: versión secuencial del algoritmo propuesto por Srinivas y Deb en [SRI94] con el que exploramos las técnicas de *fitness sharing* y el sistema de asignación de rangos propuesto por Goldberg en [GOL89];
- *NSGA2a*: con el que exploramos el efecto del paralelismo en el *NSGA1a*;

- *NSGA1b*: versión secuencial del mismo *NSGA* pero con una población externa añadida, de modo que sea posible verificar el efecto de la implementación del elitismo para este algoritmo;
- *NSGA2b*: con el que exploramos el efecto del paralelismo en el *NSGA1b*;
- *SPEA1*: para verificar el efecto del sistema de asignación de *fitness* con *sharing* implícito y *clustering* propuesto por Zitzler en [ZIT99];
- *SPEA2*: para comprobar el efecto del paralelismo en el *SPEA1*.

Como ya se ha especificado previamente estos algoritmos se han seleccionado específicamente porque incorporan lo que parecen ser los conceptos teóricos claves de los *MOEA*s como asignación de rangos, *niching*, *fitness sharing* y elitismo; además el *SPEA* tiene la convergencia asegurada por el teorema presentado en [VAN99].

Los parámetros para los experimentos con los algoritmos propuestos se especifican en la sección de características experimentales del capítulo 5.

Primeramente, para la aplicación de los algoritmos escogidos, es necesario realizar la codificación de las redes. Luego, es necesario escoger los parámetros para cada algoritmo genético.

El primer paso de los experimentos, que consiste en la codificación de las redes, se realizó utilizando la matriz de adyacencia correspondiente al grafo [HORO83] que representa a la red. Como ejemplo, supongamos que se desea codificar la red de la figura 4.1, que cuenta con 5 nodos y 3 tipos de enlaces diferentes.

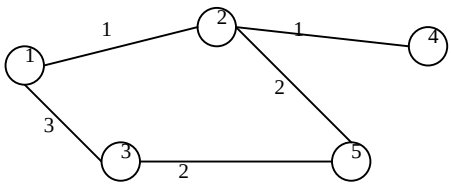


Figura 4.1. Grafo que representa a una red de computadoras.

El primer paso consiste en determinar la matriz de adyacencia:

	1	2	3	4	5
1	0	1	3	0	0
2	1	0	0	1	2
3	3	0	0	0	2
4	0	1	0	0	2
5	0	2	2	0	0

Figure 4.2. Matriz de adyacencia para la red de la figura 4.1

Como la misma es simétrica, solo se utilizará el triángulo superior, ubicado por encima de la diagonal principal, que en la gráfica se encuentra pintada en un tono de gris. Leyendo este triángulo de izquierda a derecha y de arriba hacia abajo (en el mismo sentido que utilizamos para las lecturas de palabras escritas) obtenemos una cadena de números, en este caso la cadena 1300012022. Sabemos, por consiguiente, que el primer uno de la cadena indica que entre los nodos 1 y 2 existe un enlace bidireccional del tipo 1 (habiéndose escogido previamente la codificación para los tipos de enlaces), mientras que entre los nodos 1 y 4 no existe un enlace directo, i.e. la posición correspondiente en la cadena contiene un 0.

La población inicial para los algoritmos se genera de manera probabilística, i.e. se generan cadenas $\boldsymbol{x}$ de enteros, donde cada $x_i \in \{0, 1, 2, \ldots, T\}$. El valor 0, que indica la ausencia del enlace, aparece con probabilidad 0.5, mientras que los otros enlaces aparecen con probabilidad $0.5/T$; de esta manera, se pretende iniciar la búsqueda en redes con pocos enlaces y bajos costos, e ir agregando los enlaces a la medida de las necesidades de confiabilidad.

El cálculo de los objetivos se realiza utilizando los procedimientos descritos en el capítulo 2, y los parámetros para ambos procedimientos se describen en el capítulo siguiente.

Con respecto a la implementación de restricciones de emparejamiento, aunque en la literatura se menciona bastante esta técnica [VAN99, DEB94], no la hemos incorporado en ninguna de sus variantes. Esto se debe a que los algoritmos mismos no las mencionan en sus versiones originales; además requieren gran cantidad de codificación y añaden mayor complejidad. Por último, en varios estudios (por ejemplo en [VAN99]) se sugiere que muy probablemente no mejorarán los resultados de manera considerable, por lo que no se justifican.

El proceso de asignación de *fitness*, el *fitness sharing* y el *clustering* se han implementado también en la manera descrita para ambos algoritmos en el capítulo tres.

En cuanto a la aplicación de elitismo hay una observación importante que debe hacerse. El *SPEA*, por su propia naturaleza incorpora el concepto de elitismo al mantener una población externa con los individuos no dominados hallados hasta un dado momento; sin embargo el *NSGA* reemplaza su población completa con la generada con los operadores evolutivos y no mantiene información sobre individuos buenos ya explorados de manera explícita. En primeras instancias es lícito sugerir que por este comportamiento el *SPEA* generará mejores resultados que el *NSGA*. Tal intuición ha sido comprobada con los experimentos prácticos, por lo que si deseamos establecer un marco de comparación justo y balanceado es necesario introducir alguna forma de elitismo al *NSGA*, o eliminarlo del *SPEA*. La primera aproximación es la más adecuada ya que no violamos los conceptos de diseño de ninguno de los algoritmos si

adjuntamos al *NSGA* una población externa que sirva solamente para propósitos de archivar las soluciones no dominadas halladas y no participe de los operadores genéticos. Esta es la aproximación que hemos escogido para nuestros experimentos, según sugerimos previamente. De todos modos, también se incluyen los resultados obtenidos con el *NSGA* tradicional, a modo de soporte y para verificación experimental del buen efecto del elitismo en este algoritmo.

Para determinar el momento adecuado en que se debe parar el algoritmo hemos diseñado un método de parada que también representa una contribución original del presente trabajo al ámbito de los *MOEAs*. Según este método, se establece la convergencia de los *MOEA*s cuando la población externa no ha sufrido alteraciones en su composición luego de una cantidad dada de generaciones. Tal criterio resulta extremadamente sencillo de implementar y no altera mayormente el tiempo de cómputo del algoritmo. Además, resulta flexible e intuitivo, ya que siempre es posible aumentar el número de generaciones alterando el parámetro y es muy lógico pensar que si el algoritmo no detecta nuevas soluciones para el frente Pareto que propone es debido a que ya alcanzó algún tipo de óptimo.

4.4 Elección de las métricas para medir el desempeño

La elección de las métricas que sean capaces de dar valores adecuados para el desempeño de los *MOEA*s en diferentes aspectos, y por ende permitir comparaciones significativas entre diferentes algoritmos e implementaciones es todavía un ámbito de discusión e investigación. En tal sentido gran parte de las métricas propuestas se han aplicado solamente a problemas que no provienen del mundo real, sino que se han diseñado para propósitos teóricos y de prueba solamente. Queda aún abierto el tema de escoger cuáles de estas métricas se pueden aplicar exitosamente a problemas reales y qué tan significativos pueden ser los resultados obtenidos con ellas.

La selección de las métricas es una tarea que se debe realizar cuidadosamente so pena de obtener resultados poco útiles. Es conveniente recordar que ningún criterio único puede dar una idea acabada del desempeño general de los *MOEAs*, ya que algunos se enfocan en la efectividad y otros en la eficiencia. También puede ser interesante tener una idea de la eficiencia y efectividad medidas en el tiempo, i.e. medir el progreso de un *MOEA* en cada generación; esto podría ser muy útil para establecer un criterio de parada que se ajuste a las necesidades. Todos los aspectos son importantes al momento de juzgar a un *MOEA*. En las secciones siguientes se presentan las métricas escogidas para analizar los resultados de los experimentos presentados. La lista no debe considerarse un completo compendio; más bien, sus componentes se han escogido de modo a tener un espectro de discusión lo suficientemente amplio y extendido.

Las métricas presentadas miden el desempeño en el dominio de los fenotipos (espacio objetivo). En otros trabajos [VAN99] se han presentado métricas que se enfocan en el genotipo (espacio de búsqueda). Como existe una correspondencia directa entre las soluciones Pareto óptimas y el frente Pareto óptimo, no se puede afirmar que un método es “mejor” que el otro.

Sin embargo, es útil notar que diferentes soluciones podrían mapearse a un único vector objetivo.

Por otro lado, es conveniente recordar que como las métricas reflejan la similitud entre el frente Pareto óptimo real $\boldsymbol{Y}_{true}$ y el frente Pareto computado $\boldsymbol{Y}_{known}$ y como $\boldsymbol{Y}_{true}$ es nuestra incógnita, hemos construido $\boldsymbol{Y}_{true}$ juntando todos los individuos no dominados hallados en todos los conjuntos solución propuestos, i.e. el frente Pareto óptimo real se aproxima mediante las mejores soluciones halladas en el curso de los experimentos.

El conjunto de test comprende las métricas que se listan en las siguientes secciones.

4.4.1 Generación de vectores no dominados (*GVND*)

Esta métrica cuenta el número de soluciones en el frente Pareto propuesto $\boldsymbol{Y}_{known}$. Se puede definir mediante la siguiente ecuación:

$$GVND \stackrel{\Delta}{=} |\, Y_{known} \,|_c \tag{4.1}$$

donde $|\ |_c$ denota cardinalidad.

Schott [SCH95] usa esta métrica (aunque definida sobre el conjunto Pareto óptimo $\boldsymbol{X}_{known}$). Definir esta métrica en términos de genotipo o fenotipo es algo más bien subjetivo y tiene que ver con la propia preferencia; aunque conviene notar nuevamente que múltiples soluciones pueden mapearse a idénticos vectores objetivo. A pesar de que contar el número de soluciones no dominadas obtenidas ayuda a tener una idea de la efectividad del *MOEA* para generar soluciones deseadas, la métrica no refleja que tan "lejos" se encuentran los vectores de $\boldsymbol{Y}_{known}$ de los de $\boldsymbol{Y}_{true}$.

4.4.2 Razón de generación de vectores no dominados (*RGVND*)

Como un valor muy bajo de *GVND* puede implicar poca efectividad, y un valor muy alto puede conducir a confusiones o mayor tarea para quien toma la decisión final, es difícil establecer un rango de valores deseables para *GVND*. Además, dependiendo del problema en cuestión y el modo en que se trata, la cardinalidad de $\boldsymbol{Y}_{known}$ puede variar considerablemente. Por tanto, resulta más útil indicar la razón entre la cantidad de vectores no dominados hallados y la cantidad de vectores no dominados existentes. A esta métrica conocemos como razón de generación de vectores no dominados (*RGVND*) y matemáticamente se define como:

$$RGVND \stackrel{\Delta}{=} \frac{GVND}{|\, Y_{true} \,|_c} \tag{4.2}$$

Como el objetivo es obtener un frente Pareto $\boldsymbol{Y}_{known}$ tan similar al verdadero frente Pareto como sea posible, un valor cercano a 1 es deseable.

4.4.3 Generación real de vectores no dominados (*GRVND*)

Aunque las métricas anteriores brinden buenos resultados, aún no dan idea clara del buen desempeño del algoritmo en la búsqueda, ya que el mismo puede dar como resultado un conjunto con igual cantidad de elementos que el frente Pareto óptimo real, pero sin que exista una correspondencia entre éstos y los elementos hallados. La métrica denominada generación real de vectores no dominados cuenta la cantidad de elementos en el frente Pareto hallado que en efecto pertenecen al frente Pareto óptimo real. Se define de la siguiente manera

$$GRVND \overset{\Delta}{=} \left| \left\{ y \mid y \in Y_{known} \quad \wedge \quad y \in Y_{true} \right\}_c \right| \tag{4.3}$$

4.4.4 Razón de Error (*E*)

Esta razón reporta la proporción de vectores objetivo en $\boldsymbol{Y}_{known}$ que no son miembros de $\boldsymbol{Y}_{true}$. Por ello una razón cercana a 1 indica una baja correspondencia entre el frente obtenido y el real; i.e. $E = 0$ es deseable. La definición matemática es:

$$E \overset{\Delta}{=} \frac{\sum_{i=1}^{N} e_i}{GVND}$$

$$e_i \begin{cases} 0 & si \text{ un vector en } \mathrm{Y_{known}} \text{ esta también en } \mathrm{Y_{true}} \\ 1 & de\ otro \bmod o \end{cases} \tag{4.4}$$

Nótese que:

$$E = \frac{GVND - GRVND}{GVND} \tag{4.5}$$

por lo que en algunas publicaciones no se lo usa en conjunto con *GRVND* [VAN99].

4.4.5 Distancia generacional (*G*)

Presentada primeramente en [VAN98b]; esta métrica es un valor que representa que tan lejos está $\boldsymbol{Y}_{known}$ de $\boldsymbol{Y}_{true}$. Se define como:

$$G \stackrel{\Delta}{=} \frac{\left(\sum_{i=1}^{N} d_i^2\right)^{1/2}}{GVND} \qquad \textbf{(4.5)}$$

donde d_i es la distancia euclidiana (en el espacio objetivo) entre cada vector objetivo $\boldsymbol{F}$ en $\boldsymbol{Y}_{known}$ y su miembro correspondiente más cercano en el frente Pareto óptimo real $\boldsymbol{Y}_{true}$. Un valor grande de G indica que $\boldsymbol{Y}_{known}$ está alejado de $\boldsymbol{Y}_{true}$; $G = 0$ es la situación ideal.

4.5 Entorno computacional e implementación

Todos los *MOEA*s se han ejecutado en la misma plataforma computacional para mantener la consistencia. Utilizamos un conjunto de hasta ocho computadores personales con procesador AMD-K6 de 350 MHz y 128 Mb de memoria RAM. Estas computadoras se han conectado a través de un *switch Ethernet* IBM 524, formando una red de área local (LAN). El sistema operativo base es Linux, en la versión 7.0 de la distribución de Mandrake. El lenguaje de programación en que se escribió el código es ANSI C, y el mismo se ha compilado usando los compiladores de GNU. Para las versiones paralelas se utilizaron las librerías proveídas con la distribución 3.0 de PVM (*Parallel Virtual Machine*).

El código de las implementaciones está disponible sobre pedido. Los pedidos se pueden realizar a la siguiente dirección de correo electrónico: sduarte@cnc.una.py.

Tanto el software como el hardware para las implementaciones se han escogido de manera a abaratar costos en la medida de lo posible, ya que pretendemos no solo resolver el problema, sino además hacerlo de la manera más adecuada, en el marco de la situación económica de los usuarios finales de un país en desarrollo como el nuestro, que no pueden permitirse derroches vanos y que no tienen acceso plataformas computacionales muy potentes.

Con estas aclaraciones, se presentan los resultados experimentales en el capítulo siguiente.

5 Resultados Experimentales

5.1 Introducción

En el presente capítulo se presentan los resultados de los experimentos con los algoritmos discutidos y según la metodología propuesta. El trabajo realizado se centra principalmente en los siguientes aspectos:

a) el análisis de rendimiento del proceso de paralelización, para determinar el grado de escalabilidad de la solución planteada;
b) la calidad de las soluciones halladas por los algoritmos, de modo que se pueda determinar el grado de efectividad de los algoritmos evolutivos especializados en optimización multiobjetivo, para el problema propuesto; y,
c) el efecto del elitismo para mejorar el rendimiento de los algoritmos multiobjetivo, en el problema de estudio.

En la sección 5.2 se plantea el problema de referencia escogido para llevar a cabo los experimentos. Los parámetros para los algoritmos evolutivos se incluyen en la sección 5.3. En la sección 5.4 se presentan los resultados de los experimentos con todos los algoritmos propuestos y se discuten los mismos. En la sección 5.5 se resumen las conclusiones experimentales y se resaltan las diferencias entre los resultados obtenidos con aproximaciones previas y con las aproximaciones multiobjetivo secuenciales y paralelas.

5.2 Problema de referencia

El problema de prueba se basa en la expansión de la red de ULAK-NET, primeramente publicado en [DEE98]. Este problema es una versión simplificada de un problema de diseño de redes real.

Debido al continuo crecimiento del uso de Internet, el Tubitak (*Technological Research Council of Turkey*) inició el proyecto ULAK-NET cuyo propósito era la construcción de una red de alta velocidad.

Se trata del diseño de una espina dorsal de alto rendimiento, concebida para enlazar 19 nodos en distintas universidades y centros de investigación de 9 ciudades diferentes, ubicadas en Turkía. En consecuencia, el número de posibles enlaces es de 171 (según la ecuación 2.1).

Se ha escogido a este problema debido a que es el problema ejemplo de mayor tamaño encontrado en el curso de las investigaciones realizadas para este trabajo. Además, los resultados previamente hallados con otras aproximaciones están disponibles en [LAU00, DEE98] y pueden ser utilizados para comparaciones.

La tabla 5.1 muestra la matriz de distancia en kilómetros para cada par de nodos.

	v2	***v3***	***v4***	***v5***	***v6***	***v7***	***v8***	***v9***	***v10***	***v11***	***v12***	***v13***	***v14***	***v15***	***v16***	***v17***	***v18***	***v19***
v1	111	126	120	122	115	116	132	346	968	343	344	106	107	105	454	613	828	1261
v2	-	15	15	17	5	6	243	458	1079	454	456	10	11	5	565	724	939	1342
v3		-	15	17	13	14	258	473	1094	469	471	25	26	23	580	740	954	1357
v4			-	2	5	6	248	460	1082	456	457	12	13	15	570	730	943	1353
v5				-	8	9	251	463	1085	459	460	15	16	18	573	733	946	1355
v6					-	1	246	457	1080	454	455	10	9	12	568	728	940	1350
v7						-	245	456	1079	453	454	9	8	11	567	727	939	1351
v8							-	384	383	380	381	235	236	240	322	542	831	1301
v9								-	766	3	4	450	451	453	580	542	487	920
v10									-	763	764	1074	1075	1077	1345	1307	972	624
v11										-	1	450	451	453	582	544	489	921
v12											-	449	450	452	583	545	490	922
v13												-	1	4	560	720	932	1337

v14													-	3	561	721	933	1338
v15														-	563	723	934	1340
v16															-	469	898	1424
v17																-	553	1079
v18																	-	526
Tabla 5.1. Matriz de distancia para el problema de test.																		

Tres tipos de tecnología de comunicación basadas en fibras ópticas se utilizan para interconectar los nodos (t = 3). Los costos y confiabilidades correspondientes de los tres tipos se indican en la tabla 5.2.
Los valores de costo mostrados en la tabla 5.2 corresponden al costo unitario, por unidad de distancia (dólares/kilómetros) e incluyen material, costos por empalmes, repetidores e instalación. Los costos de los nodos de comunicación no se incluyen, pero pueden agregarse muy fácilmente.

Nº del tipo	***Costo ($/Km)***	***Confiabilidad***
0	0	0
1	333	0.96
2	433	0.975
3	583	0.99
Tabla 5.2. Costos y Confiabilidades asociadas a los diferentes tipos de enlace.		

De esta manera, el tamaño del espacio de búsqueda es de $3.8*10^{81}$ individuos. Cada individuo tiene la forma (x_1,…, x_{171}), donde cada $x_i \in \{0, 1, 2, 3\}$.

A partir de la cadena que representa a cada individuo, se puede, entonces, calcular el costo y la confiabilidad asociados al mismo mediante los métodos presentados en el capítulo 2.

5.3 Parámetros de los algoritmos evolutivos

Para establecer el rendimiento de los algoritmos en estudio se realizaron corridas empleando las implementaciones mencionadas en el capítulo 4. Los parámetros generales de los algoritmos son los siguientes:

- Tamaño de la población (N): 100 individuos para la versión secuencial, 50 para la versión paralela con 2 procesadores, 25 para la versión paralela con 4 procesadores y 12 para la versión paralela con 8 procesadores.
- Tamaño de la población externa de no dominados (N'), si la misma existe: 100 individuos para todos los casos.
- Máxima cantidad de generaciones (g_{max}): 10000.
- Probabilidad de cruzamiento (p_c): 1.
- Razón de mutación (r_m): 0.3.

- Porcentaje de la población mutada en cada generación (*m*%): 5%

La población inicial para cada algoritmo se ha generado mezclando azar y probabilidades, de manera que se generen más frecuentemente individuos con menores cantidades de enlaces y los mismos se inserten a la población inicial. Se logra esto dando a los enlaces de tipo 0 una probabilidad de 0.5 de aparecer en cada *allele* del individuo, mientras que los otros tipos de enlaces (1, 2 y 3) se reparten equitativamente el otro 0.5 de probabilidad, i.e. cada uno tiene 0.17 de probabilidad de aparecer en cada *allele* del cromosoma). Se optó por este método ya que el mismo ha demostrado gran utilidad para acelerar la convergencia. De esta manera se parte de individuos con mala confiabilidad y bajo costo, hacia redes de mejor confiabilidad pero sin aumentar mucho el costo. Si inicialmente ya se generan redes con muchos enlaces, el proceso de convergencia se retarda por perderse mucho tiempo de búsqueda en zonas donde existe muy poca probabilidad de encontrar soluciones con buena confiabilidad y bajo costo.

El cálculo de las funciones objetivo para cada individuo se realiza según los algoritmos descritos en el capítulo 2. Para calcular la confiabilidad se utilizan las simulaciones Monte Carlo. Solo 10000 replicaciones se realizaron por cada red a evaluar, debido al enorme costo computacional implicado. Como se ha establecido previamente que las redes tienen que tener confiabilidad mínima superior a 0, las redes que no cumplen tal requerimiento nunca se insertan en la población externa, aunque sí pueden ser parte de la población genética normal que podría obtener buenos individuos a partir de la aplicación de operadores genéticos sobre ellas. La excepción es la implementación del *NSGA* sin población externa, que incluye en su población normal a los dominantes; las redes no conectadas son partes de esta población a pesar de no cumplir el requerimiento, pero no se reportan en los resultados finales del algoritmo.

Luego de hallar los individuos dominantes, los mismos se copian a la población externa, si la misma existe. Esta siempre se mantiene consistente y no debe superar su tamaño máximo predefinido. El procedimiento utilizado para mantener consistente la población externa en el NSGA es el mismo que el utilizado en el *SPEA*.

La asignación de *fitness* o *strength* (según sea el caso) se realiza de acuerdo con los algoritmos descritos en el capítulo 3, para cada algoritmo evolutivo implementado. Como el *NSGA* realiza un procedimiento para compartir *fitness* (*fitness sharing*), requiere como parámetro adicional el radio del nicho, que se ha fijado en los experimentos siguientes en 0.01.

El procedimiento de selección se implementa en el caso del *NSGA* utilizando ruletas de probabilidades distribuidas según el *fitness* de los individuos. En el *SPEA* se utilizan torneos binarios con reemplazo.

La nueva generación se obtiene mediante cruzamientos de 1 punto sobre los individuos escogidos como padres.

Todas estas decisiones se tomaron en base a lo propuesto por los mismos creadores de ambos algoritmos, según se puede hallar en [SRI94, ZIT99].

El operador de mutación, por su parte, toma *m*% de individuos de la población y cambia cada *allele* del mismo con una probabilidad de 0.3.

El criterio de parada que se ha implementado permite parar el ciclo generacional del algoritmo una vez que el mismo ya no encuentra nuevas soluciones y por ende, no vale la pena seguir hasta alcanzar el número máximo de generaciones. Específicamente, el ciclo generacional del *SPEA* y del *NSGA* con población externa sigue siempre y cuando se continúen insertando nuevos individuos en la población externa antes de que ocurran más de 10 generaciones. Se ha comprobado, sin embargo, que tal número es pequeño, por lo que en las implementaciones paralelas (que son más rápidas) se ha aumentado a 50. Esta decisión no afecta al marco de desarrollo de los experimentos, ya que no se pretende realizar un análisis del nivel de aceleración obtenido con mayor cantidad de procesadores (*speedup*), sino de la factibilidad de obtener mejores soluciones en el espacio multiobjetivo.

En el *NSGA* que no posee población externa, el criterio de parada es el mismo, aunque implementado en una manera ligeramente diferente. Como no existe población externa, en cada generación se cuenta la cantidad de dominantes y se para el algoritmo si dicha cuenta no varía en el curso de 50 generaciones.

5.4 Resultados

Los resultados presentados a continuación se obtuvieron de corridas sucesivas sobre la red descrita en el capítulo 4, utilizando hasta 8 computadores personales. Se utilizaron diferentes configuraciones de la red para obtener los resultados mostrados en la tabla 5.3 y en la tabla 5.4. La tabla 5.3 representa a las corridas del *SPEA*, la 5.4 se refiere a las corridas del *NSGA* sin elitismo, mientras que la 5.5 contiene los resultados de las corridas del *NSGA* realizadas con una población externa que no participa de los operadores genéticos.

En todas las tablas, la primera columna indica el número de la corrida. Como se realizaron 10 corridas con cada configuración, las corridas 1 al 10 corresponden a corridas del algoritmo secuencial, las corridas numeradas del 11 al 20 se realizaron en paralelo sobre 2 procesadores; las corridas del 21 al 30 se implementaron sobre 4 procesadores; y las corridas del 31 al 40 corresponden a los resultados obtenidos en las 10 corridas paralelas con 8 procesadores.

La segunda columna de las tablas indica la cantidad de procesadores utilizada en cada corrida.

La columna 3 presenta la cantidad de soluciones Pareto halladas en la corrida, valor que se conoce como *GNVD* según se presentó en el capítulo 4.

La columna cuarta contiene el número de soluciones no dominadas que se han hallado y que pertenecen al conjunto Pareto Optimo real $\boldsymbol{Y}_{true}$, i. e. contiene los valores de *GRVND* para cada corrida.

La columna número 5 representa el cómputo de *GRVND*.

La columna 6 contiene el error según se definió el mismo en el apartado 4.3.4.

La columna 7 expresa el tiempo, medido en horas, que duró cada corrida.

La última columna se obtiene aplicando el concepto de dominancia Pareto sobre las columnas 3, 4, 5, 6, 7, en cada tabla; es decir, esta columna representa una clasificación de todas las corridas de un algoritmo dado, según un rango. El rango se obtiene en base al tiempo y las demás métricas utilizadas. Los rangos de mejores corridas son menores a los rangos de peores corridas. Este rango se ha hallado en base a las observaciones formuladas en capítulos previos acerca de la imposibilidad de establecer el buen desempeño de un algoritmo mediante una única métrica, i.e. el problema de decidir qué algoritmo se desempeña mejor es también un problema de optimización de objetivos múltiples.

Nº	Cant. Proc.	GVND	GRVND	RGVND	E	Tiempo	RANGO
1	1	41	0	0.323	1	8.64	7
2	1	42	0	0.331	1	8.712	7
3	1	46	0	0.362	1	8.95	7
4	1	46	0	0.362	1	8.45	6
5	1	51	0	0.402	1	9.003	5
6	1	44	3	0.346	0.932	8.35	6
7	1	43	1	0.339	0.977	8.96	7
8	1	51	0	0.402	1	8.472	5
9	1	51	2	0.402	0.961	8.269	3
10	1	57	0	0.449	1	8.726	4
11	2	45	4	0.354	0.911	5.946	5
12	2	47	6	0.370	0.872	5.267	4
13	2	52	0	0.409	1	5.002	4
14	2	56	2	0.440	0.964	5.637	3
15	2	57	0	0.449	1	5.891	3
16	2	41	5	0.323	0.878	5.236	4
17	2	47	7	0.370	0.851	5.189	3
18	2	50	2	0.394	0.96	4.968	3
19	2	54	0	0.425	1	4.256	3
20	2	46	3	0.362	0.935	4.781	3
21	4	49	1	0.386	0.980	2.78	3
22	4	52	1	0.409	0.981	2.64	2
23	4	56	2	0.441	0.964	2.859	2
24	4	41	10	0.323	0.756	3.05	1
25	4	59	3	0.465	0.949	2.956	2
26	4	47	3	0.370	0.936	2.567	3
27	4	49	8	0.386	0.837	2.368	2
28	4	54	3	0.425	0.944	2.945	2
29	4	53	2	0.417	0.962	2.847	2
30	4	51	12	0.402	0.765	2.369	1
31	8	55	0	0.433	1	1.498	1
32	8	50	1	0.394	0.98	1.486	1
33	8	49	6	0.386	0.878	1.56	1
34	8	56	11	0.441	0.804	1.689	1
35	8	53	9	0.417	0.830	1.67	1
36	8	61	3	0.480	0.951	1.547	1
37	8	47	2	0.370	0.957	1.689	1
38	8	51	6	0.402	0.882	1.487	1
39	8	58	5	0.457	0.914	1.694	1
40	8	59	4	0.465	0.932	1.567	1

Tabla 5.3. Resultados de Corridas con el *SPEA*.

Nº	Cant. Proc.	GVND	GRVND	RGVND	E	Tiempo	RANGO
1	1	15	0	0.242	1	11.894	8
2	1	12	0	0.194	1	9.956	10
3	1	15	1	0.242	0.933	9.036	3
4	1	11	0	0.177	1	10.894	11
5	1	15	0	0.242	1	11.563	7
6	1	14	0	0.226	1	10.547	8
7	1	11	1	0.177	0.909	11.451	12
8	1	19	0	0.306	1	11.354	5
9	1	18	0	0.290	1	10.256	5
10	1	16	0	0.258	1	11.378	6
11	2	18	4	0.290	0.777	5.946	2
12	2	14	5	0.226	0.643	6.784	2
13	2	12	0	0.194	1	5.781	8
14	2	13	0	0.210	1	6.124	8
15	2	21	0	0.339	1	5.787	3
16	2	16	0	0.258	1	6.003	5
17	2	12	0	0.194	1	5.961	9
18	2	13	0	0.210	1	5.649	7
19	2	10	5	0.161	0.5	5.891	2
20	2	14	0	0.226	1	5.678	7
21	4	16	0	0.258	1	2.78	2
22	4	15	0	0.242	1	3.208	6
23	4	19	0	0.306	1	2.859	3
24	4	23	8	0.371	0.652	3.05	1
25	4	21	0	0.339	1	2.956	2
26	4	20	0	0.323	1	3.103	3
27	4	17	6	0.274	0.647	2.864	2
28	4	19	0	0.306	1	2.945	4
29	4	19	4	0.306	0.789	2.847	2
30	4	15	0	0.242	1	2.965	5
31	8	19	0	0.306	1	1.598	2
32	8	23	6	0.371	0.739	1.678	1
33	8	16	0	0.258	1	1.697	3
34	8	17	0	0.274	1	1.7	3
35	8	16	0	0.258	1	1.764	4
36	8	15	9	0.242	0.4	1.767	1
37	8	19	7	0.306	0.632	1.77	1
38	8	14	0	0.226	1	1.801	2
39	8	17	0	0.274	1	1.807	4
40	8	18	6	0.290	0.666	1.853	1

Tabla 5.4. Resultados de Corridas con el *NSGA* sin población externa.

Nº	Cant. Proc.	GVND	GRVND	RGVND	E	Tiempo	RANGO
1	1	32	4	0,224	0,875	11.894	8
2	1	41	0	0,287	1	9.956	10
3	1	29	0	0,203	1	9.036	3
4	1	31	5	0,217	0,839	10.894	11
5	1	45	0	0,315	1	11.563	7
6	1	44	0	0,308	1	10.547	8
7	1	34	2	0,238	0,941	11.451	12
8	1	36	3	0,252	0,917	11.354	5
9	1	44	1	0,308	0,977	10.256	5
10	1	46	2	0,322	0,957	11.378	6
11	2	35	0	0,245	1	5.946	2
12	2	44	6	0,308	0,864	6.784	2
13	2	42	8	0,294	0,810	5.781	8
14	2	33	14	0,231	0,576	6.124	8
15	2	31	0	0,217	1	5.787	3
16	2	45	8	0,315	0,822	6.003	5
17	2	39	0	0,273	1	5.961	9
18	2	42	5	0,294	0,881	5.649	7
19	2	50	0	0,350	1	5.891	2
20	2	43	2	0,301	0,953	5.678	7
21	4	39	2	0,273	0,949	2.78	2
22	4	46	11	0,322	0,761	3.208	6
23	4	51	8	0,357	0,843	2.859	3
24	4	47	0	0,329	1	3.05	1
25	4	39	3	0,273	0,923	2.956	2
26	4	37	5	0,259	0,865	3.103	3
27	4	45	6	0,315	0,867	2.864	2
28	4	35	0	0,245	1	2.945	4
29	4	39	4	0,273	0,897	2.847	2
30	4	45	0	0,315	1	2.965	5
31	8	39	0	0,273	1	1.598	2
32	8	47	6	0,329	0,872	1.678	1
33	8	52	3	0,364	0,942	1.697	3
34	8	46	0	0,322	1	1.7	3
35	8	43	18	0,301	0,581	1.764	4
36	8	41	4	0,287	0,902	1.767	1
37	8	42	3	0,294	0,929	1.77	1
38	8	39	1	0,273	0,974	1.801	2
39	8	48	5	0,336	0,896	1.807	4
40	8	56	4	0,392	0,929	1.853	1

Tabla 5.5. Resultados de Corridas con el *NSGA* con población externa.

Es interesante notar que a mayor cantidad de procesadores, tanto en el *SPEA* como en el *NSGA* (en sus dos versiones), se obtienen mejores rangos, i.e. mejores soluciones considerando el problema como multiobjetivo. Más aún, las tablas 5.6, 5.7 y 5.8 representan un estudio estadístico de los rangos de las corridas, para el *SPEA*, *NSGA* y *NSGA* elitista respectivamente. La suma de los rangos, así como el promedio de los mismos, decrece a medida que se incrementa el número de procesadores para todos los algoritmos. La columna cuarta de las

tablas representa la desviación estándar de la muestra, mientras que la quinta y sexta columnas contienen los rangos máximos y mínimos del conjunto, respectivamente.

Cant. Proc.	Suma de Rangos	Promedio	Desviación Estándar	Máximo Rango	Mínimo Rango
1	57	5.7	1.41814	7	3
2	35	3.5	0.70711	5	3
4	20	2	0.66667	3	1
8	10	1	0	1	1

Tabla 5.6. Estudio estadístico de experimentos con *SPEA*.

Cant. Proc.	Suma de Rangos	Promedio	Desviación Estándar	Máximo Rango	Mínimo Rango
1	75	7.5	2.8771	12	3
2	53	5.3	2.8304	9	2
4	30	3	1.5635	6	1
8	22	2.2	1.2293	4	1

Tabla 5.7. Estudio estadístico de experimentos con *NSGA* sin población externa.

Cant. Proc.	Suma de Rangos	Promedio	Desviación Estándar	Máximo Rango	Mínimo Rango
1	38	3.8	1,13529	6	2
2	29	2.9	1,37032	5	1
4	24	2.4	0,96609	4	1
8	15	1.5	0,70711	3	1

Tabla 5.8. Estudio estadístico de experimentos con *NSGA* con población externa.

Estas tablas muestran, con toda claridad, que el paralelismo es benéfico para el desempeño multiobjetivo de cada uno de los algoritmos evolutivos multiobjetivo propuestos para el problema del diseño de redes. Es decir, si se consideran todos los objetivos (métricas de test y tiempo de ejecución), las versiones paralelas están por encima de las secuenciales. Además, todos los algoritmos representan modelos escalables hasta donde hemos llegado a probarlo, ya que se obtienen “mejores” soluciones en la medida que aumenta la cantidad de procesadores.

Tal resultado resulta muy promisorio no solo para la resolución completa del problema del diseño de espinas dorsales de redes, sino para cualquier problema complejo de ingeniería que involucre la realización de diseños u optimizaciones, y donde el cómputo de las funciones objetivo haga imposible la solución a través de búsquedas exhaustivas, y donde otros métodos tradicionales tampoco provean soluciones aceptables.

Sin embargo, es muy útil una comparación un poco más detallada entre los resultados obtenidos por el conjunto de corridas del *SPEA* y aquel obtenido por las corridas del *NSGA* en sus dos versiones. Tal comparación provee información importante para decidir sobre el algoritmo que otorga mejores soluciones para el problema propuesto.

La tabla 5.9, muestra la cantidad de soluciones Pareto óptimas identificadas en los algoritmos, en sus diferentes versiones. El tiempo promedio de las corridas, considerando el número de procesadores, se muestra en la tabla 5.10; mientras que las relaciones de dominancia entre las soluciones de un algoritmo y otro, en sus múltiples versiones, se muestran en la tabla 5.11.

Descripción	***SPEA***		***NSGA* tradicional**		***NSGA* elitista**	
	Total	**Dominantes**	**Total**	**Dominantes**	**Total**	**Dominantes**
Cantidad de soluciones halladas con 1 procesador	472	6	146	2	382	17
Cantidad de soluciones halladas con 2 procesadores	495	29	143	14	404	43
Cantidad de soluciones halladas con 4 procesadores	511	45	184	18	423	39
Cantidad de soluciones halladas con 8 procesadores	539	47	174	28	453	44
Cantidad Total de Soluciones halladas	2017	127	647	62	1662	143

Tabla 5.9. Cantidad de Soluciones Pareto identificadas por cada algoritmo

Descripción	***SPEA***	***NSGA* tradicional**	***NSGA* elitista**
Tiempo promedio de las corridas con 1 procesador	8.6532 horas	10.8329 horas	16.12 horas
Tiempo promedio de las corridas con 2 procesadores	5.2173 horas	5.9604 horas	9.7895 horas
Tiempo promedio de las corridas con 4 procesadores	2.7381 horas	2.9577 horas	4.9118 horas
Tiempo promedio de las corridas con 8 procesadores	1.5887 horas	1.7435 horas	2.9674 horas

Tabla 5.10. Tiempo promedio de corridas de cada algoritmo

Descripción	**Cantidad**
Soluciones halladas por el *SPEA* y que dominan a alguna hallada por el *NSGA*, con 1 procesador	3
Soluciones halladas por el *SPEA* y que dominan a alguna hallada por el *NSGA*, con 2 procesadores	5
Soluciones halladas por el *SPEA* y que dominan a alguna hallada por el *NSGA*, con 4 procesadores	8
Soluciones halladas por el *SPEA* y que dominan a alguna hallada por el *NSGA*, con 8 procesadores	14
Soluciones halladas por el *NSGA* y que dominan a alguna hallada por el *SPEA*, con 1 procesador	0
Soluciones halladas por el *NSGA* y que dominan a alguna hallada por el *SPEA*, con 2 procesadores	4
Soluciones halladas por el *NSGA* y que dominan a alguna hallada por el *SPEA*, con 4 procesadores	6
Soluciones halladas por el *NSGA* y que dominan a alguna hallada por el *SPEA*, con 8 procesadores	11
Soluciones halladas por el *SPEA* que dominan a alguna hallada por el *NSGA* (TOTAL)	30
Soluciones halladas por el *NSGA* que dominan a alguna hallada por el *SPEA* (TOTAL)	21

Tabla 5.11. Relaciones de dominancia entre las soluciones halladas por los algoritmos

De la observación de los valores de éstas tablas resulta evidente que el *SPEA* otorga mejores soluciones al problema que se desea resolver, ya que otorga mayor cantidad de soluciones dominantes y en un menor tiempo.

El *NSGA* con población externa es capaz de obtener una mayor cantidad de soluciones, por lo que se supone que el procedimiento de *fitness sharing* utilizado para mantener la diversidad genética no opera de manera satisfactoria, ya que tiene el efecto adverso de perder buenas soluciones. Si bien, el desempeño general del *NSGA* con población externa es comparable al del SPEA, el tiempo de cómputo total del *NSGA* sigue siendo muy superior.

Comparando particularmente al *NSGA* tradicional con aquel que posee una población externa para almacenar las buenas soluciones que halla, resulta evidente que el efecto del elitismo es altamente benéfico para obtener mejores resultados. Se concluye que el *NSGA* es un algoritmo evolutivo multiobjetivo capaz de realizar una exploración adecuada en el espacio de búsqueda, pero requiere alguna forma de elitismo para no perder las buenas soluciones ya identificadas. Un tema interesante para un trabajo futuro es el de diseñar una forma más eficiente de implementar el elitismo y verificar si resulta útil que las soluciones Pareto halladas hasta un momento dado sean insertadas de algún modo en la población y/o participen de los operadores genéticos.

5.5 Comparación de los resultados obtenidos con los logrados en otros trabajos.

Una comparación directa y completa con otras implementaciones, previamente propuestas para resolver el mismo problema no es posible estrictamente hablando. Esto se debe a que esta es la primera aproximación al problema que considera al mismo como de optimización de objetivos múltiples y busca hallar todo un conjunto de soluciones óptimas al mismo. Las aproximaciones de [DEE98 y LAU00] trataban al problema como a un problema de objetivo único, y obtenían una única solución.

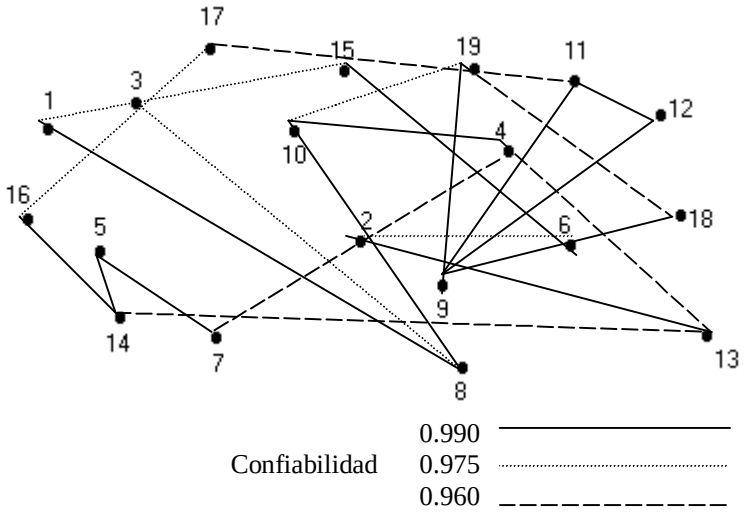
Sin embargo, comparando los resultados propuestos por [DEE98, LAU00], con algunas soluciones no dominadas halladas en el curso de los experimentos, es posible notar que no solo es posible hallar muchas soluciones explorando el espacio de búsqueda de esta manera, sino que también las soluciones finalmente encontradas pueden ser mejores, aún para el algoritmo secuencial. Esto se evidencia en la tabla 5.10, que muestra las mejores soluciones halladas por las diversas implementaciones realizadas en este trabajo y que cumplen con las restricciones propuestas en [DEE98, LAU00] que requerían una confiabilidad mínima de 0.9, también muestra las mejores soluciones publicadas en trabajos precedentes.

Referencia de Publicación	Tiempo para hallar la solución (horas)	Costo de la mejor solución (millones de $)	Confiabilidad de la mejor solución
[DEE98]	15	7694708	0.999
[LAU00]	1.6	1987805	0.991
Trabajo Presente	1.6	1755474	0.991

Tabla 5.10. Comparación de Algoritmos Evolutivos Multiobjetivos con Algoritmos Evolutivos Tradicionales

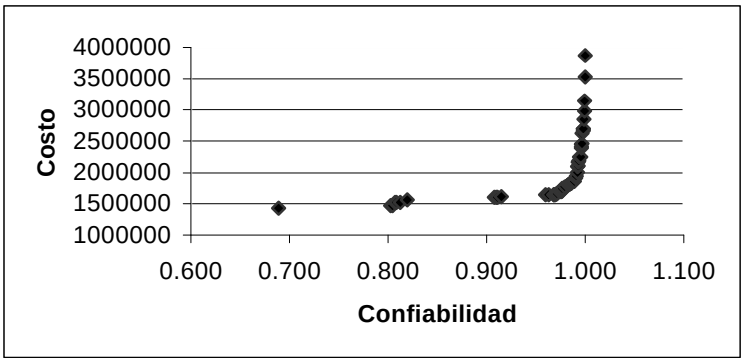
Surge, de manera innegable, la ventaja de solucionar el problema en la forma que hemos propuesto, ya que no otorgamos al responsable de las decisiones una única solución –como lo hacían las aproximaciones anteriores–, sino todo un conjunto de soluciones de entre las cuáles él puede decidir. Tal conjunto está disponible en una sola corrida del algoritmo y además, concluyendo estrictamente a partir de los resultados publicados por [DEE98, LAU00] y los obtenidos en el curso de nuestras experimentaciones, ese conjunto contiene elementos que dominan completamente a soluciones halladas en las aproximaciones mono-objetivas.

En la figura 5.1 se muestra una red típica hallada con la aproximación multiobjetivo propuesta, indicando la forma en que se expresa la misma, a partir de la codificación seleccionada. La red fue escogida al azar y los tipos de enlaces se representan con líneas de diferente estilo.



Gráfica 5.1. Red típica hallada en una de las corridas.

En la figura 5.2, se presenta un frente Pareto tomado de una de las corridas escogidas al azar. Es posible apreciar en ella todos los resultados obtenidos en la corrida y a partir de los cuales el responsable de las decisiones debe escoger a alguna, según esté dispuesto a sacrificar dinero para obtener mayor confiabilidad, o a sacrificar confiabilidad a costa de ahorrarse dinero. Las soluciones presentadas en la misma gráfica se encuentran descritas, en términos de costo y confiabilidad en la tabla 5.11.



Gráfica 5.2. Frente Pareto obtenido de una corrida típica con 4 procesadores

Reliability	Cost
0.68920	1433898
0.80280	1473659
0.80470	1477188
0.80720	1519613
0.80830	1523142
0.81250	1524807
0.81930	1566877
0.90800	1604727
0.90990	1605527
0.91110	1609056
0.91475	1613852
0.95960	1639359
0.96300	1641158
0.96810	1642023
0.96850	1646352
0.96980	1648017
0.97080	1657808
0.97170	1661337
0.97390	1689776

Reliability	Cost
0.97470	1693106
0.97530	1694771
0.97620	1698300
0.97630	1699965
0.97640	1742035
0.97940	1776381
0.98220	1804194
0.98250	1808523
0.98370	1809323
0.98390	1810988
0.98420	1814517
0.98560	1820844
0.98600	1823508
0.98660	1851947
0.98820	1854477
0.98890	1855277
0.99050	1914106
0.99120	1942881
0.99200	2008323

Reliability	Cost
0.99210	2094903
0.99260	2105892
0.99310	2167986
0.99360	2183481
0.99430	2249748
0.99540	2252568
0.99580	2389966
0.99620	2405748
0.99630	2445266
0.99650	2458872
0.99660	2631366
0.99780	2666686
0.99790	2697633
0.99830	2704293
0.99850	2848482
0.99910	2980683
0.99960	3142521
0.99980	3528468
1.00000	3873123

Tabla 5.11. Resultados obtenidos en una corrida típica con 4 procesadores.

5.6 Conclusiones Experimentales.

A partir del primer análisis de las corridas, considerando todas las métricas propuestas en el capítulo 4 y el tiempo de ejecución, es posible concluir que los algoritmos propuestos son capaces de otorgar soluciones aceptables al problema propuesto y que las soluciones obtenidas son mejores al aumentar el número de procesadores. Tal patrón es observable tanto para las implementaciones del *SPEA* como del *NSGA*, considerando el estudio de rangos que se presenta respectivamente en las tablas 5.5 y 5.6, i.e. las corridas con múltiples procesadores tienen mejores rangos. Esto conduce a pensar que la aproximación es escalable y que aumentando el

número de procesadores, será posible resolver el problema de diseño redes de mayor cantidad de nodos en un tiempo razonable.

En particular es posible observar que las soluciones propuestas por las corridas del *SPEA* son generalmente mejores que las propuestas por el *NSGA*. Eso se puede concluir debido a que la cantidad de soluciones propuestas por el *SPEA* es mayor, por ende su capacidad de hallar soluciones no dominadas es superior. Además, las soluciones halladas por el *SPEA* dominan a muchas de las halladas por el *NSGA* (en sus dos versiones), con lo que se demuestra que el *SPEA* posee mayor capacidad para identificar la zona del espacio de búsqueda donde se hallan las soluciones Pareto óptimas reales, para el problema en cuestión. También cabe notar que el tiempo promedio de las corridas de las implementaciones del *SPEA* es inferior al que se toman las corridas de las implementaciones del *NSGA*, en sus dos versiones.

Si consideramos como un problema multiobjetivo el de decidir qué algoritmo se adecua mejor a este problema, es posible afirmar que la respuesta obvia es el *SPEA*, ya que es superior al *NSGA* en cuanto a capacidad de generación de vectores no dominados, aproximación al frente Pareto óptimo real y tiempo de ejecución. Aunque también se puede argumentar, que el *NSGA* con población externa es capaz de competir con el *SPEA* en igualdad de condiciones en lo que respecta a la cantidad de vectores no dominados encontrados y en la aproximación a la solución real del problema. De todos modos, el tiempo de ejecución del *SPEA* sigue siendo menor. Aún así, es útil recordar que debido a la poca cantidad de soluciones Pareto identificadas, no se ha necesitado utilizar el procedimiento de *clustering* en el *SPEA*. Es útil recordar que el *clustering* también agrega valores al tiempo de ejecución total del algoritmo.

Finalmente, la aproximación propuesta de utilizar algoritmos evolutivos multiobjetivo y paralelos ha hallado mejores resultados que las aproximaciones previas que atacaban al problema considerando un solo objetivo.

6 Conclusiones Finales y Trabajos Futuros

6.1 Conclusiones Finales

A partir de los resultados experimentales presentados en el capítulo previo es posible responder a las preguntas planteadas en los primeros capítulos del presente trabajo, y que representaron la motivación para el mismo.

En primera instancia, según el resultado de los experimentos del capítulo anterior, se puede concluir que el *SPEA* se desempeña mejor que el *NSGA* para el problema propuesto. Se recuerda que para llegar a este resultado se ha comparado a las soluciones halladas por uno y otro algoritmo, las relaciones de dominancia entre las mismas, la aproximación al frente Pareto que las mismas representan y la cantidad de soluciones encontradas. Luego se ha considerado el tiempo de ejecución de ambos algoritmos, ya que este factor resulta clave para permitir la escalabilidad anhelada y quizás incluso la posibilidad de la realización de búsquedas interactivas.

En cuanto a la efectividad del desempeño de los *MOEAs* se ha concluido que es imposible determinarla sin la utilización de métricas múltiples de diversa índole. Esto conduce a plantear el problema de determinación de la efectividad y eficiencia, como un problema a su vez multiobjetivo, según se ha mencionado en los capítulos previos.

Por otro lado, se ha comprobado experimentalmente que el *NSGA* se beneficia enormemente con la aplicación de elitismo. Sin embargo, todavía resta un estudio detallado de las formas en que es posible hacer esto, y las ventajas y desventajas de una y otra.

En lo que respecta a la paralelización de los algoritmos, se ha comprobado que las versiones paralelas realizan búsquedas más efectivas en cuanto a tiempo y calidad de las soluciones halladas. Sería útil verificar si tal resultado persiste con el uso de mayor cantidad de procesadores.

En cuanto a las formas de paralelizar los algoritmos estudiados, la aproximación presentada ha mostrado buenos resultados y no se ha visto la necesidad de implementar más estudios al respecto.

A pesar de lo expuesto, se han identificado también algunos puntos débiles en los algoritmos estudiados. Las desventajas halladas en el curso de los experimentos se resumen de la siguiente manera. Para el *NSGA* en su formulación tradicional, falta principalmente el uso de elitismo, esto conduce a perder buenas soluciones previamente identificadas. Esto queda

parcialmente subsanado con el uso de una población externa (según se demostró con el *NSGA* elitista implementado), sin embargo el tiempo de ejecución del algoritmo crece debido a la necesidad de mantener consistente la población externa con cada generación. Por otro lado, la complejidad computacional de ordenar y colocar los individuos de la población en su rango correspondiente es muy elevada y esto se refleja en la diferencia de tiempos de ejecución que presenta el *NSGA* al compararlo con el *SPEA*.

Para el *SPEA* esencialmente la desventaja principal reside en su proceso de *clustering*, que resulta computacionalmente muy caro de implementar; otra desventaja consiste en que la forma de asignación del *strength* y la asignación posterior del *fitness*. Según la misma, se consideran solo las relaciones de dominancia entre los individuos de una población con otra y no las que existen entre los miembros de una misma población. Esto hace que el algoritmo tenga comportamientos inesperados en algunos casos particulares, por ejemplo, cuando existe un único elemento dominante en la población externa, el algoritmo es casi una búsqueda ciega; cuando existen demasiados elementos dominantes, ya no se realiza la búsqueda eficientemente y se pierde la posibilidad de ir encontrando siempre mejores soluciones.

Todas estas desventajas –y aún otras que no resultaron evidentes en el curso del desarrollo del presente trabajo– han sido ya tratadas en las versiones posteriores presentadas por los autores de ambos algoritmos: el *NSGA-II* [DEB00] y el *SPEA2* [ZIT01]. Tales versiones aparecieron hace muy poco tiempo y cuando ya se había finalizado la fase experimental de este trabajo, por lo que ya no fueron probadas extensivamente y con resultados concluyentes. De hecho, el *SPEA2* no ha sido aún formalmente publicado. De todos modos resulta interesante como un tema de trabajo futuro el estudio cuidadoso de estos algoritmos, el contraste entre las técnicas de los mismos (que han sufrido mucha variación respecto a las versiones originales) y las implementaciones de ambos.

El *NSGA II*, inspirado en trabajos de Rudolph [RUD98], incorpora el elitismo de una manera controlada realizando comparaciones sistemáticas de individuos tomados de la población de padres y de la población de hijos; las soluciones no dominadas de la población de padres se comparan con las no dominadas de la población de hijos de modo a conformar un conjunto completo de soluciones no dominadas; este conjunto se convierte luego en padre de la siguiente generación. Con esta estrategia, ha sido incluso posible la demostración de la convergencia del algoritmo al conjunto Pareto óptimo real. Sin embargo, esta técnica aún no aporta gran ayuda para mantener la diversidad de los miembros del conjunto Pareto óptimo obtenido. Para ello, es necesario incluir un mecanismo explícito para preservar la diversidad que además sea más efectivo que el inicialmente sugerido *fitness sharing*. En esta versión, Deb propone el uso de *crowding* [DEJ95] que es una técnica de formación de nichos, al igual que *fitness sharing*. Los nichos son zonas del espacio de búsqueda en las que se encuentran individuos especializados en un objetivo en particular, a tales individuos en conjunto se conoce

con el nombre de especies. *Crowding* preserva la diversidad genética a partir de la realización de competencias entre los individuos.

EL *SPEA2*, por su parte propone un esquema de asignación de *fitness* diferente y que según las afirmaciones de sus autores es más eficiente. Según el mismo, por cada individuo se considera a cuántos domina y por cuántos es dominado. Además se sugiere el uso de un estimador de densidad [SIL86] que permita realizar una búsqueda guiada más precisa en los alrededores de cada individuo. Y como última innovación, implementan un nuevo método de truncado para reemplazar al *clustering* que tiende a perder las soluciones que se hallan en los límites. En palabras de sus autores: "este algoritmo provee de buen desempeño en términos de convergencia y diversidad, presenta mejores soluciones que el *SPEA*, y es comparable al *NSGA-II* en varios problemas de prueba típicos".

6.2 Trabajos futuros

Los trabajos futuros planteados se pueden resumir de acuerdo con las dos grandes áreas que se conjugan en este trabajo: diseño de redes de computadoras y algoritmos evolutivos multiobjetivo paralelos.

En cuando al diseño de redes quedan aún abiertas las siguientes cuestiones:

a) El diseño de algún método que permita calcular la confiabilidad de manera eficiente y totalmente escalable.
b) El uso de mayor cantidad de funciones objetivo de modo que se refleje completamente al problema de diseño real, según las necesidades reales de los usuarios.
c) Verificar en qué manera incide la optimización de mayor cantidad de objetivos en el problema presentado y formalizarlo de esta manera.
d) Elaborar un esquema formal para el cálculo de los diversos objetivos que puedan entrar en juego. Tal esquema debe contar con las características de la exactitud, la eficiencia y la robustez.

En cuanto a los algoritmos genéticos multiobjetivo paralelos, el trabajo aún por realizar es mucho mayor, y se considera que se mantendrá como uno de los temas de investigación más populares aún por algún tiempo. Algunas áreas abiertas y estrechamente relacionadas al presente trabajo son:

a) Determinar un conjunto de métricas que no requiera el conocimiento previo de los resultados del problema para indicar acerca del desempeño de los algoritmos, tal conjunto de métricas debe cubrir todos los múltiples objetivos que se consideran al momento de determinar el desempeño de un algoritmo de optimización multiobjetivo.
b) Implementar otros algoritmos evolutivos multiobjetivo que han surgido en los últimos meses y realizar pruebas exhaustivas con los mismos para verificar su desempeño en el problema del diseño de redes y otros problemas complejos de ingeniería. Esta

indicación se muestra como la extensión natural y obvia del presente trabajo y se está realizando actualmente.

c) Determinar formas alternativas para la implementación de elitismo y para distribuir las soluciones halladas en todo el frente Pareto, implementar estas formulaciones y realizar pruebas exhaustivas. En este sentido se vuelve a mencionar las sugerencias de Rudolph y Deb [RUD98] que han resultado bastante inspiradoras.

d) Realizar las implementaciones paralelas con mayor cantidad de computadores y verificar que los resultados mantengan la consistencia. En caso que no sea así, verificar el número óptimo de procesadores con el cual trabajar y determinar el tamaño máximo de la red (en cuanto a cantidad de nodos y enlaces, así como cantidad de tipos de tecnologías de comunicación diferentes) que es posible tratar.