PEDECIBA Informática
Instituto de Computación – Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay

TESIS DE MAESTRÍA EN INFORMÁTICA

ALGORITMOS GENÉTICOS PARALELOS Y SU APLICACIÓN AL DISEÑO DE REDES DE COMUNICACIONES CONFIABLES

SERGIO NESMACHNOW

JULIO 2004

Orientador de Tesis: Dr. Ing. Héctor Cancela
Supervisor: Dr. Ing. Héctor Cancela

Algoritmos Genéticos Paralelos y su Aplicación al
Diseño de Redes de Comunicaciones Confiables
Sergio Nesmachnow

ISSN
Tesis de Maestría en Informática
Reporte Técnico RT04-07
PEDECIBA
Instituto de Computación – Facultad de Ingeniería
Universidad de la República.

Montevideo, Uruguay, Julio de 2004.

ALGORITMOS GENÉTICOS PARALELOS Y SU APLICACIÓN AL DISEÑO DE REDES DE COMUNICACIONES CONFIABLES

RESUMEN

Esta Tesis presenta el estudio de las técnicas de computación evolutiva y la aplicación de técnicas de procesamiento de alta performance para implementar modelos de algoritmos genéticos capaces de ejecutar en un ambiente paralelo−distribuido. Se aborda la aplicación de algoritmos evolutivos al caso concreto de problemas que surgen al diseñar redes de comunicaciones de alta conectividad topológica. En particular, se concentró el estudio sobre una clase de problemas de diseño de redes de comunicaciones que pueden modelarse bajo el denominado *Problema de Steiner Generalizado*. Dada una red de comunicaciones con ciertos nodos distinguidos denominados nodos terminales, el Problema de Steiner Generalizado propone diseñar una subred de mínimo costo que verifique un conjunto de requisitos prefijados de conectividad entre pares de nodos terminales.

En el trabajo se evalúan diferentes algoritmos evolutivos puros e híbridos en sus versiones secuenciales y paralelas, presentando un estudio comparativo que reporta resultados satisfactorios tanto desde el punto de vista de la calidad de resultados obtenidos como desde el punto de vista de la mejora de eficiencia computacional alcanzada por las versiones paralelas de los algoritmos con respecto a sus contrapartes secuenciales.

Palabras clave: Algoritmos genéticos paralelos, redes de comunicaciones confiables.

ÍNDICE

RESUMEN ..... **I**

ÍNDICE ..... **III**

CAPÍTULO 1. INTRODUCCIÓN **1**

CAPÍTULO 2. TÉCNICAS DE COMPUTACIÓN EVOLUTIVA **5**

2.1. INTRODUCCIÓN 5

2.2. TÉCNICAS DE COMPUTACIÓN EVOLUTIVA 6

2.3. RESEÑA HISTÓRICA 7

2.3.1. PRIMERAS PROPUESTAS 7

2.3.2. ESTRATEGIAS DE EVOLUCIÓN 10

2.3.3. PROGRAMACIÓN EVOLUTIVA 11

2.3.4. ALGORITMOS GENÉTICOS 12

2.4. ALGORITMOS GENÉTICOS 13

2.4.1. REPRESENTACIÓN DE SOLUCIONES 14

2.4.2. FUNCIÓN DE FITNESS 15

2.4.3. OPERADORES 16

2.4.4. MODELOS DE EVOLUCIÓN 18

2.4.5. FORMALIZACIÓN DEL MECANISMO DE LOS ALGORITMOS GENÉTICOS 18

2.4.6. COMPARACIÓN DE LOS ALGORITMOS GENÉTICOS CON OTRAS TÉCNICAS HEURÍSTICAS 19

2.5. CONCLUSIONES 20

2.6. NOTA SOBRE LAS REFERENCIAS BIBLIOGRÁFICAS 20

CAPÍTULO 3. ALGORITMOS GENÉTICOS Y TÉCNICAS DE PROGRAMACIÓN DE ALTA PERFORMANCE **21**

3.1. INTRODUCCIÓN 21

3.2. TÉCNICAS DE PROGRAMACIÓN DE ALTA PERFORMANCE 22

3.2.1. ARQUITECTURAS DE COMPUTADORES PARALELOS 22

3.2.2. CLASIFICACIÓN DE ARQUITECTURAS PARALELAS 24

3.2.3. PARADIGMAS DE PROGRAMACIÓN PARALELA 26

3.2.4. TÉCNICAS Y HERRAMIENTAS PARA EL DISEÑO E IMPLEMENTACIÓN DE APLICACIONES PARALELAS 27

3.2.5. LA BIBLIOTECA MPI 28

3.2.6. MEDIDAS DE PERFORMANCE 29
3.2.6.1. MODELO DE PERFORMANCE 29
3.2.6.2. SPEEDUP Y EFICIENCIA DE UN ALGORITMO PARALELO 31
3.2.6.3. SPEEDUP Y EFICIENCIA EN EL CONTEXTO DE LOS ALGORITMOS EVOLUTIVOS PARALELOS 33
3.3. TÉCNICAS DE PROGRAMACIÓN DE ALTA PERFORMANCE APLICADAS A LOS ALGORITMOS GENÉTICOS 35
3.3.1. MOTIVOS PARA LA APLICACIÓN DE LAS TÉCNICAS DE PROCESAMIENTO DE ALTA PERFORMANCE A LOS ALGORITMOS GENÉTICOS 36
3.3.2. CRITERIOS DE CLASIFICACIÓN DE ALGORITMOS GENÉTICOS PARALELOS 36
3.3.3. MODELOS DE ALGORITMOS GENÉTICOS PARALELOS 36
3.4. ETAPAS EN EL DESARROLLO DE ALGORITMOS GENÉTICOS PARALELOS . 38
3.4.1. 1970–1989: LAS INVESTIGACIONES PIONERAS 39
3.4.2. 1990–1995: LA INFLUENCIA DEL DESARROLLO DE LAS ARQUITECTURAS PARALELAS 42
3.4.3. 1995–2003: LA GENERALIZACIÓN Y UNIFICACIÓN DE LOS MODELOS 51
3.4.4. PANORAMA DEL ESTADO ACTUAL 58
3.5. CONCLUSIONES 61
CAPÍTULO 4. EL PROBLEMA DE STEINER GENERALIZADO 63
4.1. INTRODUCCIÓN 63
4.2. EL PROBLEMA DE STEINER GENERALIZADO 65
4.2.1. FORMULACIÓN DEL PROBLEMA DE STEINER GENERALIZADO .. 65
4.2.2. UN EJEMPLO DE PROBLEMA DE STEINER GENERALIZADO 65
4.2.3. MODELO MATEMÁTICO DEL PROBLEMA DE STEINER GENERALIZADO 68
4.2.4. VARIANTES DE PROBLEMAS DE STEINER 70
4.2.5. COMPLEJIDAD DE LOS PROBLEMAS DE STEINER 71
4.2.6. ENFOQUES PARA LA RESOLUCIÓN DE LA CLASE DE PROBLEMAS DE STEINER 71
4.2.7. CASOS PARTICULARES DEL PROBLEMA DE STEINER GENERALIZADO 72
4.3. TÉCNICAS EVOLUTIVAS APLICADAS A LA RESOLUCIÓN DE LOS PROBLEMAS DE STEINER 73
4.4. CONCLUSIONES 78
CAPÍTULO 5. UN ALGORITMO GENÉTICO PARALELO PARA EL PROBLEMA DE STEINER GENERALIZADO 79
5.1. INTRODUCCIÓN 79
5.2. CODIFICACIÓN DEL PROBLEMA 80
5.2.1. ANTECEDENTES 80

5.2.2. LA CODIFICACIÓN BINARIA UTILIZADA 80
5.2.3. CODIFICACIONES ALTERNATIVAS 82
5.3. FUNCIÓN DE FITNESS 84
5.4. OPERADORES 85
5.4.1. CÁLCULO DE FACTIBILIDAD 85
5.5. GENERACIÓN DE NÚMEROS PSEUDOALEATORIOS 86
5.6. MODELO DE PARALELISMO 87
5.7. IMPLEMENTACIÓN Y PLATAFORMA DE EJECUCIÓN 89
5.8. RESULTADOS OBTENIDOS 89
5.8.1. INSTANCIAS DE PRUEBA PARA EL PROBLEMA DE STEINER GENERALIZADO 89
5.8.2. CONFIGURACIÓN Y VALIDACIÓN DEL ALGORITMO GENÉTICO .. 90
5.8.3. CONFIGURACIÓN DEL MODELO DE MIGRACIÓN 92
5.8.4. TOPOLOGÍA DE MIGRACIÓN 94
5.8.5. FRECUENCIA DE MIGRACIÓN 94
5.8.6. TASA DE MIGRACIÓN 95
5.8.7. CONCLUSIONES SOBRE LA CALIBRACIÓN 96
5.8.8. COMPARACIÓN ENTRE MODELO PARALELO Y PANMÍCTICO 96
5.9. CONCLUSIONES 99

CAPÍTULO 6. TÉCNICAS EVOLUTIVAS APLICADAS AL PROBLEMA DE STEINER GENERALIZADO 101

6.1. INTRODUCCIÓN 101
6.2. ENFOQUE DEL PROBLEMA 102
6.3. ALGORITMOS CONSIDERADOS EN EL ESTUDIO 102
6.3.1. ALGORITMO GENÉTICO 102
6.3.2. ALGORITMO CHC 102
6.3.3. SIMULATED ANNEALING 104
6.3.4. ALGORITMOS HÍBRIDOS 106
6.3.5. ALGORITMOS PARALELOS 107
6.4. OPERADORES Y PARÁMETROS 109
6.4.1. OPERADORES 109
6.4.2. CONFIGURACIÓN DE PARÁMETROS 109
6.5. LA BIBLIOTECA MALLBA 111
6.6. PLATAFORMA DE EJECUCIÓN 111
6.7. ANÁLISIS EMPÍRICO 112
6.7.1. RESULTADOS DE LOS ALGORITMOS SECUENCIALES 112
6.7.2. ANÁLISIS DE LA EVOLUCIÓN DEL FITNESS 115
6.7.3. RESULTADOS DE LOS ALGORITMOS PARALELOS 117
6.7.4. RESULTADOS COMPARATIVOS 119
6.7.5. ANÁLISIS DE LA EFICIENCIA COMPUTACIONAL 120
6.8. CONCLUSIONES Y TRABAJO FUTURO 122

CAPÍTULO 7. CONCLUSIONES Y TRABAJO FUTURO 125
7.1. INTRODUCCIÓN 125
7.2. CONCLUSIONES 125
7.3. TRABAJO FUTURO 127
ANEXOS 129
AI.1. PROBLEMAS DE VALIDACIÓN 131
AI.2. INSTANCIAS DE PRUEBA DEL PROBLEMA DE STEINER GENERALIZADO . 139
AI.3. ESTUDIO EMPÍRICO DE OPERADORES DE CRUZAMIENTO EN UN ALGORITMO GENÉTICO APLICADO AL PROBLEMA DE STEINER GENERALIZADO 145
AI.4. AGRADECIMIENTOS 159
REFERENCIAS BIBLIOGRÁFICAS 161

CAPÍTULO 1

INTRODUCCIÓN

"I have called this principle, by which, each slight variation,
if useful, is preserved, by the term of Natural Selection.
… … …
The expression often used by Mr. Herbert Spencer of the Survival of the Fittest
is more accurate, and is sometimes equally convenient."

CHARLES DARWIN
On the Origin of Species by means of Natural Selection, 1859.

Al tratar con problemas de optimización combinatoria NP difíciles, para los cuales la complejidad de los algoritmos conocidos aumenta de manera superpolinomial con el tamaño del problema, la aplicabilidad de los métodos exactos de resolución se encuentra limitada por el enorme tiempo y consumo de recursos computacionales que demandan.

Cuando las técnicas de resolución analítica, los métodos de búsqueda exhaustiva o enumerativa y las técnicas derivadas de la programación matemática no son aplicables, las técnicas heurísticas aparecen como la única alternativa viable para abordar problemas NP difíciles con espacio de soluciones de dimensión elevada. A diferencia de las técnicas exactas, las técnicas heurísticas no pueden garantizar a priori la obtención de la solución óptima del problema. En la mayoría de los casos ni siquiera pueden garantizar que la solución obtenida tenga un cierto margen de error con respecto a la solución óptima. Pero en la práctica, numerosas técnicas heurísticas se comportan de manera sumamente satisfactoria para la resolución de complejos problemas de optimización, posibilitando la obtención de buenas soluciones aproximadas en tiempos razonables. En un nivel superior de abstracción, las *técnicas metaheurísticas* proporcionan esquemas o enfoques genéricos para la resolución de problemas complejos. Estos enfoques genéricos pueden ser instanciados para producir algoritmos específicos que trabajan bajo un mismo lineamiento general.

Las técnicas de computación evolutiva constituyen una familia de metaheurísticas estocásticas utilizadas exitosamente para la resolución de variados problemas en las áreas de búsqueda y análisis de información, optimización combinatoria, y diseño, control y aprendizaje de dispositivos, entre otras. Estas técnicas basan su funcionamiento en mecanismos análogos a los principios que rigen la evolución natural de las especies biológicas. En la naturaleza, la propia evolución de las especies puede interpretarse como un proceso de optimización para la resolución de los complejos problemas vinculados con la supervivencia de individuos que interaccionan entre sí y con un entorno de recursos limitados. Los algoritmos evolutivos se basan en aplicar una analogía que utiliza las bases del proceso evolutivo biológico, la selección natural y la supervivencia de los individuos más aptos, para la resolución de problemas de optimización combinatoria y similares.

Los algoritmos evolutivos proponen trabajar sobre un conjunto de *individuos* –que representan a soluciones potenciales del problema de optimización– y *evolucionar* esta población inicial siguiendo los conceptos de adaptación al entorno y supervivencia de los individuos más aptos con la idea de lograr mejores soluciones al problema. La evolución determina el modo de exploración del espacio de soluciones, que se recorre de modo guiado de acuerdo a una función de aptitud que determina la calidad de los individuos. La diversidad necesaria para explorar diferentes sectores del espacio de soluciones se obtiene mediante procedimientos de modificación y/o recombinación basados en reglas aleatorias, denominados *operadores de evolución.*

Existen numerosas variantes de algoritmos evolutivos, diferenciadas entre sí por los modelos utilizados para la evolución de poblaciones y por el tipo de operadores de evolución empleados (Goldberg, 1989a) (Back et al, 1997) (Mitchell, 1996). Los *algoritmos genéticos* constituyen una de las técnicas de computación evolutiva de uso más difundido. Diseñados a partir de una formulación simple, con una operativa basada en el uso de un operador de recombinación y mutaciones aleatorias de soluciones, estos algoritmos han sido reportados como exitosamente aplicables a una amplia gama de problemas de optimización y búsqueda en los últimos 25 años.

El auge de la computación paralela y distribuida ha renovado el interés práctico por los algoritmos basados en analogías de procesos naturales aplicados a la resolución de problemas de optimización sobre espacios de soluciones que contienen complejas interacciones entre diferentes componentes, donde el impacto de cada parte sobre la función de evaluación es difícil de especificar. Asimismo, el propio comportamiento de los modelos paralelos y distribuidos de algoritmos evolutivos constituye un punto interesante de estudio. En general sus modelos de evolución difieren de los modelos de las implementaciones secuenciales dependiendo fuertemente de los esquemas de paralelización lógica o física utilizada.

El incremento en la demanda de comunicaciones de datos y el auge en las tecnologías de comunicación ha propulsado en los últimos años el diseño y desarrollo de la infraestructura de redes. El continuo crecimiento en el tamaño de los problemas ha conducido a proponer alternativas a los enfoques exactos tradicionales para la resolución de problemas complejos. En este contexto, varias heurísticas se han aplicado a los problemas relacionados con el diseño de redes de comunicaciones. Las técnicas de computación evolutiva, y los algoritmos genéticos en particular, se han manifestado como métodos flexibles y robustos para resolver los complejos problemas de optimización subyacentes al diseño de redes de comunicaciones confiables.

Combinando las tres líneas de investigación presentadas en el párrafo precedente, el trabajo que se reporta en esta Tesis de Maestría tuvo como objetivos el estudio del paradigma de los algoritmos evolutivos, la implementación de diferentes modelos de algoritmos genéticos capaces de ejecutar en un ambiente paralelo–distribuido, y su aplicación al caso concreto de problemas que surgen al diseñar redes de comunicaciones de alta conectividad topológica. En particular, se concentró el estudio sobre una clase de problemas de diseño de redes de comunicaciones que pueden modelarse bajo el denominado *Problema de Steiner Generalizado*. Dada una red de comunicaciones con ciertos nodos distinguidos denominados *nodos terminales*, el Problema de Steiner Generalizado propone diseñar una subred de mínimo costo que verifique un conjunto de requisitos prefijados de conectividad entre pares de nodos terminales.

El trabajo de Maestría se desarrolló en diferentes ámbitos. Una parte importante se realizó en el marco del proyecto de Iniciación a la Investigación "*Algoritmos Genéticos Paralelos y su Aplicación al Diseño de Redes de Comunicaciones Confiables*", financiado por la Comisión Sectorial de Investigación Científica de la Universidad de la República, Uruguay. Conjuntamente, otra actividad consistió en la dirección de proyectos de grado de la carrera de Ingeniería en Computación en los cuales se investigaron aspectos relacionados con los objetivos de la Tesis. Por último, otra parte del trabajo se realizó en una pasantía en el Departamento de Lenguajes y Ciencias de la Comunicación, en la Universidad de Málaga, España.

El documento se estructura del modo que se describe a continuación.

Los conceptos generales sobre las técnicas de computación evolutiva se ofrecen en el Capítulo 2. Se presenta el paradigma desde un punto de vista genérico y su aplicación a la resolución de problemas de optimización combinatoria y problemas análogos en otras áreas. Se incluye una breve reseña histórica del desarrollo de los métodos algorítmicos basados en la emulación de los procesos de la evolución natural, presentando globalmente los modelos más populares de técnicas evolutivas: las estrategias de evolución, la programación genética y los algoritmos genéticos. En particular, se profundiza en la presentación de las características de los algoritmos genéticos, explicando los detalles de su funcionamiento y sus ventajas y desventajas respecto a otros métodos exactos y heurísticos de resolución de problemas de optimización.

El Capítulo 3 introduce los conceptos genéricos relacionados con las técnicas de computación de alta performance y su aplicación al diseño de modelos paralelos y distribuidos de algoritmos evolutivos, con el objetivo de mejorar su desempeño computacional y la calidad de sus resultados. A continuación se ofrecen las propuestas de su aplicación a los algoritmos genéticos, incluyendo una reseña de la evolución histórica en el diseño y clasificación de algoritmos genéticos paralelos. El capítulo culmina con un panorama del estado actual en el área, presentando una visión unificadora de las diferentes propuestas y nomenclaturas utilizadas.

El Problema de Steiner Generalizado se presenta formalmente en el Capítulo 4, ofreciendo su modelo matemático, la descripción de sus variantes y métodos aproximados y heurísticos propuestos para su resolución. La sección final del capítulo presenta una reseña de estrategias de aplicación de técnicas evolutivas a la resolución de versiones más simples del problema estudiado. En esta reseña se describen en detalle aquellos enfoques considerados como relevantes para el diseño de nuestros propios modelos de algoritmos evolutivos aplicados a la resolución del caso genérico del problema.

La primera propuesta de un modelo de algoritmo genético paralelo específico para la resolución del Problema de Steiner Generalizado se describe en el Capítulo 5, comentando las características de representación del problema, detalles de implementación y el modelo de paralelismo utilizado. Asimismo se incluyen los detalles de los experimentos de validación del diseño y la comparación con resultados obtenidos con otras técnicas de resolución disponibles en la literatura para instancias sencillas del problema. Complementariamente, se introduce un conjunto de instancias de prueba del Problema de Steiner Generalizado diseñadas específicamente para evaluar la calidad de resultados de los algoritmos implementados en el trabajo. Se presenta un análisis de la configuración de los múltiples parámetros del algoritmo genético paralelo. Por último, se ofrece el análisis de calidad de resultados obtenidos y eficiencia computacional en los experimentos realizados sobre el conjunto de problemas de prueba diseñado.

El Capítulo 6 presenta los detalles de la aplicación de otros algoritmos evolutivos y de búsqueda local para la resolución del Problema de Steiner Generalizado, que complementa el estudio presentado en el capítulo anterior. Se presentan las características de los nuevos algoritmos considerados (Simulated Annealing, la variante CHC de algoritmos genéticos y algoritmos híbridos que combinan Simulated Annealing y algoritmos genéticos), los detalles de implementación y un análisis comparativo de la calidad de resultados obtenidos por las versiones secuenciales y paralelas de los algoritmos al aplicarlos a la resolución de las instancias de prueba utilizadas del Problema de Steiner Generalizado. Complementando el trabajo, se presenta un estudio comparativo de la eficiencia computacional de los modelos paralelos de diferentes algoritmos al ejecutar sobre un cluster de ocho procesadores.

Por último, las conclusiones del trabajo se presentan en el Capítulo 7, conjuntamente con las numerosas propuestas que han surgido como posibles líneas de trabajo futuro.

CAPÍTULO 2

TÉCNICAS DE COMPUTACIÓN EVOLUTIVA

"Evolution is cleverer than you are."

FRANCIS CRICK

Elbow Room: the varieties of free will worth wanting,
D. Dennett, 1984

2.1. **Introducción**

Las técnicas de computación evolutiva constituyen un conjunto de heurísticas emergentes, utilizadas exitosamente para la resolución de una variada gama de problemas en las áreas de optimización combinatoria, diseño de artefactos, búsqueda de información, control de dispositivos y aprendizaje automático, entre otros. Estas técnicas basan su operativa en la emulación de los mecanismos de la evolución natural, identificados por Charles Darwin en su célebre obra *El Origen de las Especies por medio de la Selección Natural*: la *selección natural*, la *reproducción* y la *diversidad genética* de individuos (Darwin, 1859).

Las técnicas de computación evolutiva trabajan sobre una *población* compuesta por un conjunto de codificaciones de soluciones candidatas para el problema a resolver. Estas soluciones interactúan entre sí, siguiendo los principios darwinianos de la evolución natural con la idea de producir iterativamente mejores soluciones al problema. Las soluciones potenciales se evalúan mediante una función de adecuación o *función de fitness*, que toma en cuenta el problema que se plantea resolver. En la naturaleza, durante el proceso evolutivo los seres vivos tratan de resolver los problemas relacionados con la supervivencia para garantizar la perpetuación de la especie. Mediante el mecanismo comentado, las técnicas de computación evolutiva emulan el proceso biológico de adaptación de los organismos vivos al entorno y las condiciones del medio, aplicándolo a la resolución de problemas en variadas áreas.

Este capítulo presenta a las técnicas de computación evolutiva en general, y a los algoritmos genéticos en particular, como mecanismos de resolución de problemas de optimización combinatoria, y problemas análogos en otras áreas de aplicación. El capítulo comienza con una introducción a las técnicas de computación evolutiva que presenta sus principales características. A continuación se ofrece un panorama histórico del desarrollo de las técnicas algorítmicas basadas en la emulación de los procesos de la evolución natural, presentando globalmente los modelos más populares de técnicas evolutivas: las *Estrategias de Evolución*, la *Programación Genética* y los *Algoritmos Genéticos*. Las características de los algoritmos genéticos son presentadas en la sección siguiente, explicando los detalles de su funcionamiento y sus ventajas y desventajas respecto a otros métodos exactos y heurísticos de resolución de problemas de optimización.

2.2. Técnicas de Computación Evolutiva

La expresión genérica *computación evolutiva* designa a un amplio conjunto de técnicas heurísticas de resolución de problemas complejos que basan su funcionamiento en un mecanismo análogo a los procesos de la evolución natural. Trabajando sobre un conjunto de soluciones a un problema determinado, la metodología utilizada por estas técnicas se fundamenta en el uso de mecanismos de selección de las mejores soluciones potenciales y de construcción de nuevas soluciones candidatas mediante recombinación de características de las soluciones seleccionadas.

Varios esquemas algorítmicos han sido propuestos para los algoritmos evolutivos. La idea más generalizada sobre el mecanismo de un algoritmo evolutivo que trabaja sobre una población **P** se presenta en la Figura 2.1.

```
Inicializar(P(0))
generación=0;
mientras (no CriterioParada) hacer
    Evaluar(P(generación))
    Padres = Seleccionar(P(generación))
    Hijos = Aplicar Operadores Evolutivos (Padres)
    NuevaPoblacion = Reemplazar(Hijos, P(generación))
    generación ++
    P(generación) = NuevaPoblacion
fin
retornar Mejor Solución Encontrada
```

Figura 2.1: Esquema algorítmico genérico de un algoritmo evolutivo.

El algoritmo evolutivo trabaja sobre individuos que representan potenciales soluciones al problema, codificados de acuerdo a un mecanismo prefijado. Los individuos son evaluados de acuerdo a una *función de fitness* que toma en cuenta la adecuación de cada solución al problema que se intenta resolver.

La operativa del algoritmo evolutivo comienza con una etapa de inicialización de los individuos, que puede ser completamente aleatoria, muestreando al azar diferentes secciones del espacio de soluciones, o guiada de acuerdo a características del problema a resolver. El algoritmo evolutivo podría inclusive tomar como población inicial individuos resultantes como salida de algún otro algoritmo heurístico de resolución que permitiera calcular buenas soluciones iniciales aproximadas para el problema.

La evolución propiamente dicha se lleva a cabo en el ciclo que genera nuevos individuos a partir de la población actual mediante un procedimiento de aplicación de operadores estocásticos. En este ciclo se distinguen cuatro etapas:

- *Evaluación*: etapa que consiste en asignar un valor de adecuación (*fitness*) a cada individuo en la población. Este valor evalúa que tan bien resuelve cada individuo el problema en cuestión, y es utilizado para guiar el mecanismo evolutivo.
- *Selección*: proceso que determina candidatos adecuados, de acuerdo a sus valores de fitness, para la aplicación de los operadores evolutivos con el objetivo de engendrar la siguiente generación de individuos.
- *Aplicación de los operadores evolutivos*: etapa que genera un conjunto de descendientes a partir de los individuos seleccionados en la etapa anterior.
- *Reemplazo*: mecanismo que realiza el recambio generacional, sustituyendo individuos de la generación anterior por descendientes creados en la etapa anterior.

Diversas políticas para la selección y el reemplazo de individuos permiten modificar las características del algoritmo evolutivo. Aplicando políticas adecuadas es posible privilegiar los individuos más adaptados en cada generación (*estrategias de elitismo*), aumentar la presión selectiva sobre individuos mejor adaptados, generar un número reducido de descendientes en cada generación (*modelos de estado estacionario*), y otras muchas variantes.

Los *operadores evolutivos* determinan el modo en que el algoritmo explora el espacio de soluciones del problema. Una gran diversidad de propuestas de operadores evolutivos han surgido en los casi cuarenta años de vida de la computación evolutiva. Los diferentes operadores y las particularidades en su modo de aplicación dan características peculiares a las distintas variantes de algoritmos evolutivos. Los *operadores de recombinación*, que permiten combinar características de dos o más individuos con la idea de obtener descendientes mejor adaptados y los *operadores de mutación*, que introducen diversidad mediante modificaciones aleatorias, son los operadores evolutivos más difundidos.

La condición de parada de la fase iterativa del algoritmo evolutivo usualmente toma en cuenta la cantidad de generaciones procesadas, deteniéndose el ciclo evolutivo al alcanzar un número prefijado de generaciones. Otras alternativas consideran la variación de los valores de fitness –deteniendo el ciclo evolutivo cuando el proceso se estanca y no obtiene mejoras considerables en los valores de fitness− o estimaciones del error cometido respecto al valor óptimo del problema o una aproximación, en caso de conocerse.

2.3. Reseña histórica

Las técnicas de computación evolutiva surgieron sobre 1960, interpretando la naturaleza como una formidable máquina de resolver problemas y tratando de encontrar el origen de dicha potencialidad para utilizarla en la resolución de problemas complejos.

Desde los inicios de la era de la computación varios referentes vislumbraron la idea de aplicar los mecanismos naturales para diseñar dispositivos capaces de evolucionar en su operativa y aplicarse a la resolución de complejos problemas en las áreas de aprendizaje automático, problemas de optimización y búsqueda de información. Luego de la época inicial, en donde se propusieron las primeras ideas sobre el funcionamiento de los mecanismos evolutivos, tres modelos algorítmicos diferenciados se desarrollaron en forma simultánea, aunque en algunas ocasiones han interactuado entre sí. Las *Estrategias de Evolución*, la *Programación Evolutiva* y los *Algoritmos Genéticos* constituyen hoy en día las principales líneas de trabajo en el área de la computación evolutiva. Esta sección presenta un breve resumen histórico de su desarrollo, desde las propuestas pioneras hasta su consolidación como potentes mecanismos de resolución de problemas en los últimos quince años.

2.3.1. Primeras propuestas

Las relaciones detectadas entre los procesos de aprendizaje y la evolución natural dieron el marco para las primeras ideas sobre algoritmos evolutivos en la década de 1950. Ya en 1948 Alan Turing había sugerido la conexión entre ambos aspectos, proponiendo desarrollar programas *automodificables* capaces de jugar ajedrez y simular otras actividades *inteligentes* desarrolladas por los seres humanos, utilizando técnicas evolutivas. Los detalles de las ideas y las propuestas de Turing pueden consultarse en la sección correspondiente en la Enciclopedia de Filosofía de la Universidad de Stanford (Hodges, 2002).

John von Neumann también se interesó en la combinación de técnicas evolutivas y computación, en especial en el caso de los autómatas celulares. Sobre el final de su vida se encontraba trabajando en el área, tal como lo testimonia su texto inconcluso ***Teoría de Autómatas AutoReplicables*** (Von Neumann, 1966) que sería editado luego de la muerte del autor por su colega A. Burke. Von Neumann propuso mecanismos evolutivos basados en la programación para implementar autómatas con un poder computacional equivalente a una máquina universal de Turing. Además, conjeturó sobre el comportamiento de ***poblaciones*** de autómatas capaces de abordar problemas complejos ***comunicándose*** entre sí. Al respecto de las ideas y los trabajos de Von Neumann es posible consultar el texto de Dyson (1999) y la reseña de Mitchell (1998) sobre autómatas celulares.

El matemático y biólogo Nils Barricelli tomó la posta y utilizó la infraestructura montada por Von Neumann en el Instituto para Estudios Avanzados en la Universidad de Princeton, para llevar a cabo la primera simulación de "vida artificial" basada en principios evolutivos, en el período comprendido entre los años 1953 y 1956. Su trabajo consistió en diseñar un modelo computacional para la evolución darwiniana e investigar el rol de la simbiogénesis en el origen y desarrollo de la vida. La simbiogénesis considera a los organismos vivientes como una sucesión de asociaciones simbióticas entre formas sencillas de vida. Al respecto de su trabajo, Barricelli comentó: "La distinción entre la evolución entre números en una computadora y entre nucleótidos en un laboratorio químico es muy sutil" (citado por Dyson (1999)). Barricelli trabajó directamente sobre código de máquina para implementar un "universo" compuesto de 512 celdas a ser ocupadas por números de 8 bits a los cuales denominó ***genes***. Un conjunto de reglas evolutivas gobernaban la propagación de los números en la grilla..

Con su simulación de vida artificial, Barricelli sentó el precedente sobre el uso de los métodos evolutivos para la resolución de problemas. En su trabajo valoró adecuadamente la utilidad de los mecanismos de recombinación para mejorar la adaptación de individuos, indicando que el proceso evolutivo basado exclusivamente en la selección por aptitud y un operador probabilístico de mutación de individuos no era capaz en general de producir buenos resultados en tiempos razonables, y que un operador de recombinación, que según Barricelli debía estar basado en conceptos de simbiogénesis, era necesario para acelerar la evolución.

Un estupendo y ameno resumen del trabajo de Barricelli se presenta en el texto de Dyson (1999). Los conceptos de la simbiogénesis y su relación con la computación evolutiva pueden consultarse en los trabajos recientes de Watson y Pollack (1999, 2000) y Corning (1998).

Uno de los primeros intentos de aplicar técnicas evolutivas para la resolución de problemas prácticos de ingeniería, se propuso en el área de control estadísticos de procesos. En 1957, George Box propuso modificar los sistemas de operaciones estáticas tradicionales por mecanismos dinámicos capaces de realizar ajustes en las variables de control, evaluar sus efectos y modificar el proceso para mejorar los resultados obtenidos, siguiendo una analogía con el desarrollo de los procesos químicos en la naturaleza. Aplicó su idea a procesos de manufactura, proponiendo el método Evolutionary Operation –EVOP–, que introduce variaciones deliberadas en los parámetros tratando de mejorar el objetivo del proceso. Para evitar cambios abruptos en la calidad de los productos, se limita a introducir pequeños cambios en las variables de control.

Debido a limitaciones al poder computacional, el esquema de Box nunca fue implementado en la práctica como un algoritmo para computadora, pero en ciertos casos la técnica fue aplicada con controles de calidad realizados por un ***comité técnico de evaluación*** compuesto por personas idóneas. Con el posterior incremento en la habilidad de los procesadores para asimilar, manejar y evaluar grandes volúmenes de información, la técnica recibió interés creciente. Al respecto del método Evolutionary Operation pueden consultarse los propios trabajos de Box (1957, 1998). Asimismo, el texto de Goldberg (1989a) brinda una breve descripción de la técnica propuesta por Box, aplicada al control de un sencillo proceso dependiente de tres variables.

Otras técnicas que simulaban a la evolución natural para intentar el aprendizaje automático de dispositivos, simular entornos de vida biológica o resolver problemas genéricos fueron propuestas a fines de la década de 1950.

R. Friedberg trabajó en 1958 con un sistema de recompensas para calificar instrucciones que influían en la calidad de los resultados de programas sencillos (Friedberg, 1958). El mecanismo de aprendizaje se basaba, al igual que en la propuesta de Box, en realizar pequeñas modificaciones aleatorias y luego evaluar los programas modificados. Friedberg llegó a evolucionar secuencias de instrucciones −codificadas en lenguaje de máquina− que realizaban cómputos modestos, capaces de realizar operaciones como el desplazamiento de un bit desde una celda de memoria a otra. El enfoque de Friedberg contaba con la particularidad de representar comportamientos mediante un procedimiento generalizado (es decir, a través de programas completos) y de proponer modificaciones basadas en procedimientos muy especializados. Esta característica ha sido indicada como un defecto por otros investigadores en el área de la Inteligencia Artificial (Mc Carhty, 1990), dado que pequeñas modificaciones en la operativa de una máquina no pueden relacionarse en general con pequeñas modificaciones a su programa. Otros investigadores formularon críticas más duras al trabajo de Friedberg: M. Minsky argumentó que un método basado en una búsqueda puramente aleatoria era mucho mejor que el algoritmo de Friedberg, mientras que J. McCarthy y H. Simon propusieron seguir un enfoque más relacionado con los procesos de la evolución natural, duplicando subrutinas, realizando copias modificadas y dejando otras sin modificar.

Aunque Friedberg nunca consideró estrictamente que su trabajo fuera una emulación de la evolución natural, es posible interpretar de este modo su propuesta. El sistema de recompensas de Friedberg se basaba en la idea de aplicar un mecanismo de selección de instrucciones asociado con la frecuencia con la cual producían resultados exitosos. Para mayores detalles sobre las ideas de Friedberg es posible consultar los artículos de Mc Carthy (1990, 2003).

Una ***simulación de sistemas genéticos*** utilizando computadoras fue realizada por S. Fraser en 1957. Aunque su línea de trabajo era la biológica, Fraser logró interpretar parte de la potencialidad de los sistemas computacionales para emular sistemas biológicos evolutivos. Pero analizando el mecanismo evolutivo desde su punto de vista biológico, no fue capaz de proponer una extensión de su algoritmo a condiciones artificiales o resolución de problemas no relacionados con la biología (Fraser, 1957).

En la década de 1960, A. Newell y H. Simon propusieron un sistema al que llamaron “solucionador general de problemas” (*General Problem Solver*) que permitía al usuario especificar un escenario compuesto por ***objetos*** y definir ***operadores*** a aplicar sobre los objetos. El sistema se mostró capaz de resolver problemas sencillos definidos en espacios con número reducido de parámetros, utilizando técnicas heurísticas especificadas por el programador y que permitían al método evolucionar sus resultados utilizando un método al estilo del ensayo y error. Algunos argumentos formulados contra el mecanismo evolutivo del *General Problem Solver* se centraron el criticar el hecho de que la "inteligencia" del sistema estuviera completamente determinada por las heurísticas especificadas por el programador y que el sistema no era capaz de tomar "iniciativas" por su propia cuenta, al no incluir operadores evolutivos genéricos. De todos modos, el *General Problem Solver* consistió en una de las primeras propuestas de un sistema capaz de hallar soluciones genéricas, independientes del dominio de los problemas a resolver (Newell y Simon, 1963).

Los estudios de W. Bledsoe y H. Bremermann son generalmente mencionados como los trabajos precursores del concepto moderno de Algoritmos Genéticos. Estos reconocidos científicos, reconocidos como los "padres de la inteligencia artificial" junto con J. McCarthy, M. Minsky y S. Amarel entre otros, fueron capaces de interpretar la potencialidad de los mecanismos evolutivos para la resolución de problemas. Respecto a su trabajo con Bledsoe, Bremermann ofrece una descripción de los primeros intentos de aplicar técnicas evolutivas para mejorar algoritmos de reconocimiento de patrones que utilizaban redes neuronales artificiales: "Sobre 1960-61 experimentamos con la idea de aplicar *algoritmos genéticos* para optimizar la eficiencia de los procesos de percepción: tratar los pesos de las conexiones sinápticas como nucleótidos de ADN; mutarlos, recombinarlos y seleccionarlos, como en la evolución darwiniana ... El método funcionó en principio, pero el problema de entrenamiento de las redes neuronales resultó tener extraordinarios requerimientos computacionales, ya que involucraba el uso de un gran número de conexiones sinápticas con pesos asociados" (Bremermann y Anderson, 1991).

De acuerdo a Goldberg (1989a) y De Jong et al. (1997), Bledsoe y Bremermann sugirieron la idea de codificación binaria, y el uso de un valor de aptitud en la aplicación de los algoritmos evolutivos para la resolución de problemas de optimización numérica. Bledsoe propuso el esquema de generar individuos, aplicar mutaciones y seleccionar los que produjeran mejores resultados y Bremermann lo extendió para considerar poblaciones de individuos. (Boyer et al, 1996, 1998). Bremermann notó la importancia del operador de mutación para evitar el estancamiento del proceso de búsqueda en mínimos locales del problema. El primer resultado teórico sobre la operativa de los algoritmos evolutivos fue alcanzado por Bremermann al determinar la probabilidad de mutación óptima para resolver problemas linealmente separables, presentado en Bremermann et al. (1965).

Aunque la mayoría de sus conceptos se utilizan en la actualidad, y de hecho Bremermann es aceptado como el "creador" de los algoritmos genéticos (Fogel y Anderson, 2000), en su trabajo obtuvo algunos resultados decepcionantes al aplicarlos para optimizar funciones lineales y convexas. Con frecuencia sus algoritmos evolutivos no lograban converger a la solución del problema, y requerían ser complementados con otras heurísticas para lograr buenos resultados. En la actualidad se conoce que precisamente en esos dominios de trabajo, los algoritmos evolutivos no son capaces de competir con las técnicas heurísticas tradicionales de optimización, pero en contrapartida son capaces de obtener buenos resultados para numerosos tipos de problemas donde las heurísticas fallan con frecuencia.

2.3.2. Estrategias de Evolución

Las estrategias de evolución –*Evolutionsstrategie*– fueron introducidas por Rechenberg en 1965. En su propuesta inicial, consistía en un método de optimización que trabajaba sobre individuos compuestos por números reales para optimizar parámetros en problemas de diseño en ingeniería (Rechenberg, 1965). El método modelaba la evolución al nivel de los propios valores a optimizar –sin utilizar codificaciones específicas– aplicando un mecanismo de selección determinística y un operador de mutación aleatoria basada en distribuciones de probabilidad, en general distribuciones gaussianas.

A partir de esa primera propuesta, los métodos de estrategias de evolución se han difundido ampliamente. La utilización del operador de mutación como base del mecanismo evolutivo caracteriza a esta familia de métodos, aunque en versiones recientes se han propuesto modelos que incorporan mecanismos de recombinación como operador secundario, como la familia de *cruzamientos aritméticos* (Back et al, 1997).

La propuesta inicial de Rechenberg fue desarrollada luego por Schwefel en la década de 1970. Su modelo de evolución característico, guiado por el operador de mutación, los distingue de las otras técnicas evolutivas, y por este motivo fueron estudiados de forma independiente. En su versión más simple, el proceso evolutivo se basa en la generación de un descendiente por parte de un individuo padre, mediante el operador de mutación, reemplazando el nuevo individuo generado a su progenitor en la población. Dos modelos avanzados de Estrategias de Evolución fueron formulados por Schwefel en su tesis doctoral de 1975. Ambas variantes trabajan con un conjunto de padres que genera un conjunto de descendientes y se diferencian por el modo de reemplazar los individuos progenitores por sus descendientes. Estos modelos avanzados son actualmente conocidos como Estrategias de Evolución (μ,λ) y Estrategias de Evolución $(\mu+\lambda)$. En ambos modelos μ padres generan λ descendientes, pero mientras en el modelo (μ,λ) la selección se realiza solamente entre los descendientes, en el modelo $(\mu+\lambda)$ los padres y los descendientes compiten entre sí (Schwefel, 1975).

Ninguno de los modelos de estrategias de evolución fue estudiado con profundidad ni aplicados hasta fines de los años 80, cuando la influencia del poder de cómputo de las computadoras paralelas hizo resurgir el interés por esta técnica evolutiva. Diversos esquemas han sido propuestos en los años 90, aplicando conceptos de paralelismo, optimización multiobjetivo y esquemas de adaptación. Sobre 1995 fueron presentados nuevos resultados teóricos sobre los modelos $(\mu+\lambda)$ y (μ,λ) (Back et al, 1997).

2.3.3. Programación Evolutiva

Las técnicas de programación evolutiva fueron propuestas por Fogel, Owens y Walsh en 1966 como un método para evolucionar comportamientos de autómatas de estado finito utilizados para la predicción de series temporales (Fogel et al, 1966). El enfoque presentado en el artículo referido ha sido considerado “moderno”, en el sentido de que sigue la línea de trabajo actual de la computación evolutiva. Introduce el concepto de población y dimensiona adecuadamente la importancia del mecanismo de selección en el proceso evolutivo, tomando en cuenta el “entorno” al cual se encuentran expuestos los individuos. El método fue aplicado con éxito para la resolución de problemas sencillos, aunque no faltó la crítica habitual sobre que la técnica propuesta no era superior que una búsqueda completamente aleatoria.

En 1975, Holland sugirió que un algoritmo evolutivo podría ser utilizado con una codificación que representara un programa (Holland, 1975). Luego de una década en el silencio, la programación evolutiva se renovó a mediados de 1980, modificando su planteo para permitir representaciones que extendieron su aplicabilidad para la resolución de variados problemas de optimización. En la década de 1980, algunos investigadores retomaron la sugerencia de Holland y realizaron implementaciones de programas evolutivos aplicados a heurísticas de optimización (Smith, 1985), al dilema del prisionero (Fujicki, 1986), y a la evolución de expresiones LISP para programas capaces de resolver juegos sencillos (Hicklin, 1986). Estos trabajos y otras aplicaciones de la programación evolutiva se presentan en el artículo de P. Angeline (1998), que ofrece una perspectiva histórica de las propuestas de estructuras ejecutables evolutivas.

La principal línea de trabajo de programas evolutivos se basa en el trabajo de John Koza, quien le dio el nombre de ***programación genética***. Koza propuso un mecanismo evolutivo para la generación de programas, extendiendo las ideas previas de Cramer respecto al uso de árboles de parsing para representar programas (Koza, 1996). A partir de la propuesta de Koza, diversos investigadores han estudiado sobre temas relacionados con la programación genética, incluyendo la aplicabilidad de la técnica a un amplio conjunto de problemas, fundamentos teóricos del mecanismo, diseño de operadores evolutivos adecuados y aplicación de técnicas de procesamiento paralelo, entre otros (O' Reilly, 1995).

En los últimos años, las líneas de trabajo en el área de la programación evolutiva se han concentrado principalmente en aplicaciones como el entrenamiento de redes neuronales y la evolución de sistemas difusos. Algunos nuevos fundamentos matemáticos han sido presentados por parte del creador de las técnicas (Fogel, 1995).

2.3.4. Algoritmos Genéticos

Aún reconociendo los trabajos pioneros de Bremermann, en la práctica se considera a Holland como el fundador de los Algoritmos Genéticos, por ser quien formalizó su mecanismo de funcionamiento en su texto *Adaptation in Natural and Artificial Systems* (Holland, 1975). El trabajo de Holland se remonta a la década de 1960, donde comenzó su estudio de las características de los sistemas adaptativos. A lo largo de sus publicaciones es posible reconocer que sus modelos iban tomando las características de los modernos algoritmos genéticos, al trabajar con representaciones (genomas) y operadores de mutación, cruzamiento e inversión.

Otros investigadores trabajaron con Holland en la Universidad de Michigan realizando sus tesis de doctorado en el área de los nóveles algoritmos genéticos. Bagley propuso utilizarlos para determinar los parámetros en funciones de evaluación de programas capaces de jugar juegos sencillos. Rosenberg y Weinberg los aplicaron en la simulación de organismos *alife* o de vida artificial. Cavicchio trabajó sobre problemas de reconocimiento de patrones, siguiendo las ideas de Bledsoe y Browning. Hollstein aplicó algoritmos genéticos a un problema de optimización matemática pura. Frantz estudió el fenómeno de la epístasis, el efecto posicional y los operadores de inversión, y colaboró proponiendo nuevos operadores evolutivos. Todos estos trabajos son comentados en detalle en el *Handbook of Evolutionary Computation* (Back et al, 1997), y en el texto de Goldberg (1989a).

La tesis doctoral de De Jong (1975) presentó una serie de experimentos computacionales aplicados a la optimización de un conjunto de funciones que, con el transcurrir del tiempo, se consolidaron como el conjunto de funciones de prueba estándar para determinar la utilidad de las técnicas evolutivas. De Jong combinó sus análisis con los resultados teóricos presentados por Holland sobre el mecanismo de búsqueda de los algoritmos genéticos –el *teorema de los esquemas*– definiendo las bases teóricas para el estudio de las técnicas evolutivas.

En la década de 1980 los algoritmos genéticos fueron ganando popularidad como estrategia de resolución de problemas combinatorios y de diseño. El interés creciente y la formación de una comunidad de investigadores en el área condujeron a la realización de la primera Conferencia Internacional en Algoritmos Genéticos en 1985. La publicación del texto de Goldberg (1989a) contribuyó a difundir aún más esta técnica evolutiva, presentando los fundamentos teóricos y una variada gama de aplicaciones en las áreas de optimización, búsqueda y aprendizaje. Desde entonces, los intereses de la comunidad científica que trabaja en el área de algoritmos genéticos se han diversificado, extendiendo los análisis teóricos, proponiendo nuevos modelos y abordando una amplia gama de aplicaciones.

La próxima sección presenta en detalle las características de los algoritmos genéticos y las particularidades de su mecanismo evolutivo.

2.4. Algoritmos Genéticos

Los algoritmos genéticos constituyen una de las técnicas de computación evolutiva más difundidas en la actualidad, como consecuencia de su versatilidad para resolver un amplio rango de problemas. Al constituir un caso de técnica evolutiva, los algoritmos genéticos basan su operativa en una emulación de la evolución natural de los seres vivos, trabajando sobre una población de soluciones potenciales evoluciona de acuerdo a interacciones y transformaciones únicas. Los individuos que constituyen la población se esfuerzan por sobrevivir: una selección programada en el proceso evolutivo, inclinada hacia los individuos más aptos, determina aquellos individuos que formarán parte de la siguiente generación. El grado de adaptación de un individuo se evalúa de acuerdo al problema a resolver, mediante la definición de una función de adecuación al problema, la *función de fitness*. Bajo ciertas condiciones, el mecanismo definido por los operadores inspirados por la genética natural y la evolución darwiniana lleva a la población a converger hacia una solución aproximada al óptimo del problema, luego de un determinado número de generaciones.

La formulación tradicional de un algoritmo genético fue presentada de excelente manera en el texto de Goldberg (1989a), el cual popularizó el uso de los Algoritmos Genéticos para una variada gama de problemas en las áreas de búsqueda, optimización y aprendizaje automático.

En su formulación clásica, los algoritmos genéticos se basan en el esquema genérico de un Algoritmo Evolutivo presentado en la Figura 2.1. A partir de este esquema, el algoritmo genético define "Operadores Evolutivos" que implementan la recombinación de individuos (el operador de *cruzamiento*) y la variación aleatoria para proporcionar diversidad (el operador de *mutación*), resultando el esquema presentado en la Figura 2.2.

```
Inicializar(P(0))
generación=0
mientras (no CriterioParada) hacer
    Evaluar(P(generación))
    Padres = Seleccionar(P(generación))
    Hijos = Aplicar Recombinación (Padres)
    Hijos = Aplicar Mutación (Hijos)
    NuevaPob = Reemplazar(Hijos, P(generación))
    generación ++
    P(generación) = NuevaPob
fin
retornar Mejor Solución Encontrada
```

Figura 2.2: Esquema algorítmico de un algoritmo genético.

La característica distintiva de los algoritmos genéticos respecto a las otras técnicas evolutivas consiste en su uso fundamental del cruzamiento como operador principal, mientras que la mutación se utiliza como operador secundario tan solo para agregar una nueva fuente de diversidad en el mecanismo de exploración del espacio de soluciones del problema. Inclusive la mutación puede llegar a ser un operador opcional o estar ausente en algunas variantes de algoritmos genéticos que utilizan otros operadores para introducir diversidad.

En general, los algoritmos genéticos se han utilizado para trabajar con codificaciones binarias para problemas de búsqueda en espacios de cardinalidad numerable, aunque su alto nivel de aplicabilidad ha llevado a proponer su trabajo con codificaciones reales, e inclusive con codificaciones no tradicionales, dependientes de los problemas a resolver.

Algunos autores consideran que el uso del mecanismo de selección denominado *selección proporcional*, que determina la cantidad de copias de individuos a considerar en la recombinación de forma proporcional a sus valores de fitness, es una característica que distingue a los algoritmos genéticos (Back et al, 1997). Pero tomando en cuenta la diversidad de mecanismos de selección utilizados en las propuestas de algoritmos genéticos en los últimos años es posible concluir que si bien en general se recurre a la selección proporcional, otros mecanismos son igualmente utilizados para modificar el proceso de exploración del espacio de soluciones de los diferentes problemas abordados.

2.4.1. Representación de soluciones

Los algoritmos genéticos no trabajan directamente sobre las soluciones del problema en cuestión, sino que lo hacen sobre una abstracción de los objetos solución, usualmente denominadas *cromosomas* por analogía con la evolución natural biológica. Un cromosoma es un vector de *genes*, mientras que el valor asignado a un gen se denomina *alelo*.

En la terminología biológica, *genotipo* denota al conjunto de cromosomas que definen las características de un individuo. El genotipo sometido al medio ambiente se denomina *fenotipo*. En términos de los algoritmos genéticos el genotipo también esta constituido por cromosomas, utilizándose generalmente un único cromosoma por individuo solución al problema. Por ello suelen utilizarse indistintamente los términos *genotipo*, *cromosoma* e *individuo*. Por su parte, el *fenotipo* representa un punto del espacio de soluciones del problema.

Dado que un algoritmo genético trabaja sobre *cromosomas*, se debe definir una función de *codificación* sobre los puntos del espacio de soluciones, que mapea todo punto del espacio de soluciones en un genotipo, tal como se indica en la Figura 2.3. La función inversa de la codificación, denominada *decodificación* permite obtener el fenotipo asociado a un cromosoma.

$$Codificación : Espacio\ de\ soluciones \rightarrow cromosoma$$

Figura 2.3: Especificación de la función de codificación de un algoritmo genético.

Tomando en cuenta la observación anterior, los mecanismos de codificación de individuos solución resultan importantes para el proceso de búsqueda de los algoritmos genéticos. Habitualmente los algoritmos genéticos utilizan codificaciones *binarias* de largo fijo. Los individuos se codifican por un conjunto de cardinalidad conocida de valores binarios (ceros y unos) conocido como *string de bits* o *bitstring*. Cada *bitstring* representa a una solución potencial del problema de acuerdo al mecanismo de codificación predefinido, en general dependiente del problema.

Otros esquemas de codificación han sido utilizados con menor frecuencia en los algoritmos genéticos. En particular las codificaciones basadas en números reales son útiles para representar soluciones cuando se resuelven problemas sobre espacios de cardinalidad no numerable, como en el caso de determinación de parámetros en problemas de control o entrenamiento de redes neuronales. Los esquemas basados en permutaciones de enteros son útiles para problemas de optimización combinatoria que involucran hallar ordenamientos óptimos, como los problemas de *scheduling* o el reconocido Traveling Salesman Problem. Codificaciones dependientes de los problemas se han propuesto con frecuencia, como un mecanismo que permite incorporar conocimiento específico en la resolución de problemas complejos.

La Figura 2.4 resume gráficamente la relación entre el espacio de soluciones de un problema, la población de cromosomas con la cual trabaja el algoritmo genético y las funciones de codificación y decodificación.

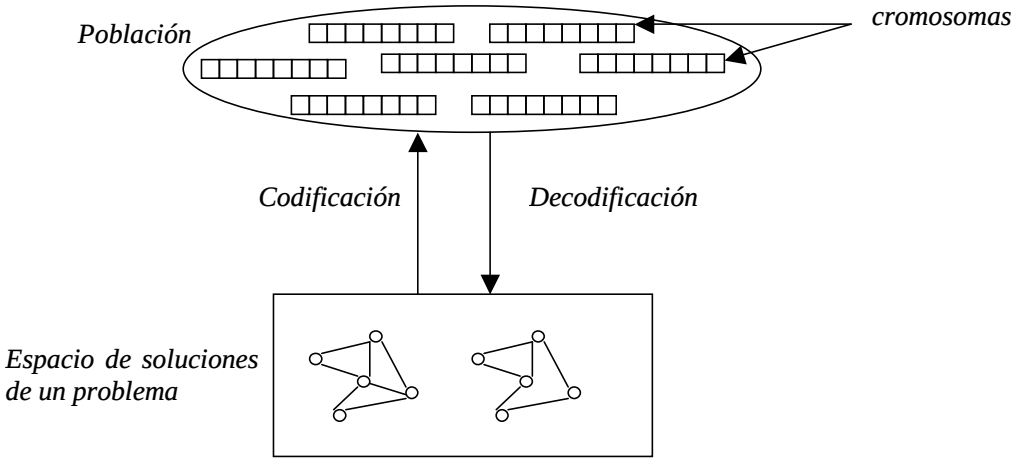


Figura 2.4: Codificación de soluciones en un algoritmo genético.

2.4.2. Función de fitness

Todo cromosoma tiene un valor asociado de *fitness* que evalúa aptitud del individuo para resolver el problema en cuestión. La función de fitness tiene el mismo tipo que la función objetivo del problema, lo cual implica que el cálculo del valor de fitness se realiza sobre el fenotipo correspondiente al cromosoma, como se presenta en la Figura 2.5.

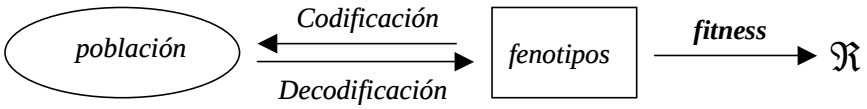


Figura 2.5: Codificación y función de fitness.

La función de fitness tiene una influencia importante en el mecanismo del AG. Si bien actúa como una caja negra para el proceso evolutivo, la función de fitness guía el mecanismo de exploración, al actuar representando al entorno que evalúa la bondad de un individuo solución para la resolución del problema.

Varios puntos han sido identificados como importantes al momento de construir la función de fitness. De acuerdo a Alba y Cotta (2003), es posible indicar que:

- La función de fitness debe contemplar el criterio del problema de optimización (minimización o maximización de un objetivo) y las restricciones presentes en el problema de optimización. En caso de surgir soluciones no factibles del problema, la función de fitness deberá asignarle valores adecuados que garanticen que tales individuos no se perpetúen durante el proceso evolutivo (valores muy pequeños en el caso de un problema de maximización y muy elevados en el caso de un problema de minimización). A tales efectos diversas transformaciones a aplicar sobre funciones de fitness han sido definidas para mapear problemas de minimización a maximización y aplicar penalizaciones a soluciones no factibles (Goldberg, 1989).

- Deben contemplarse los casos en que el entorno presente problemas para la evaluación de la función de fitness, utilizando evaluaciones parciales cuando existen valores de fitness no definidos. Asimismo, debe considerarse el uso de funciones de fitness multivaluadas que asignen diferentes valores a un mismo individuo para simplificar la labor del operador de selección.
- El caso de funciones de fitness que varían dinámicamente durante la evolución del algoritmo genético debe tenerse en cuenta. En este caso mecanismos complejos como las representaciones múltiples son útiles para introducir memoria en la operativa del algoritmo genético.
- En general, la función de fitness será compleja de evaluar, en particular demandará un esfuerzo computacional mucho mayor que el requerido para realizar los operadores evolutivos. Inclusive podría ocurrir que el proceso de evaluación sea tan complejo que solamente valores aproximados pudieran obtenerse en tiempos razonables. Este aspecto será importante al momento de proponer técnicas de alta performance para mejorar la eficiencia de los algoritmos genéticos.
- En caso de problemas con objetivos múltiples, todos ellos deben estar contemplados en la función de fitness. La utilización de algoritmos genéticos para la resolución de problemas multiobjetivo constituye en sí misma una subárea de investigación con complejidades inherentes.
- Para resolver problemas de dominancia de soluciones muy adaptadas en generaciones tempranas de la evolución, y para evitar el estancamiento en poblaciones similares al final de la evolución, deben considerarse mecanismos de *escalado* de los valores de fitness (Michalewicz, 1992).

2.4.3. Operadores

La gran mayoría de las variantes de algoritmos genéticos utiliza como principales operadores a la selección, la recombinación y la mutación.

El mecanismo de selección determina el modo de perpetuar ***buenas*** características, que se asumen son aquellas presentes en los individuos más adaptados. El mecanismo de ***selección proporcional*** o ***selección por ruleta*** elige aleatoriamente individuos utilizando una ruleta sesgada, en la cual la probabilidad de ser seleccionado es proporcional al fitness de cada individuo. Otros mecanismos de selección introducen diferentes grados de elitismo, conservando un cierto número prefijado de los mejores individuos a través de las generaciones. En el caso de la ***selección por torneo*** se escogen aleatoriamente un determinado número de individuos de la población, los cuales compiten entre ellos para determinar cuáles se seleccionarán para reproducirse, de acuerdo a sus valores de fitness. El mecanismo de ***selección basado en el rango*** introduce el mayor grado de elitismo posible, al mantener entre generaciones un porcentaje generalmente elevado, de los mejores individuos de la población. Estas diferentes políticas de selección, conjuntamente con políticas similares utilizadas para determinar los individuos reemplazados por los descendientes generados posibilitan el diseño de diferentes modelos evolutivos para los algoritmos genéticos.

Los esquemas de codificación binaria de largo fijo tienen como ventaja principal que resulta sencillo definir operadores evolutivos simples sobre ellos. En la formulación clásica de un algoritmo genético, denominada por Goldberg (1989b) como *Algoritmo Genético Simple*, se propone como operador de recombinación el *cruzamiento de un punto*, que consiste en obtener dos descendientes a partir de dos individuos padres seleccionando un punto al azar, cortando los padres e intercambiando los trozos de cromosoma, tal como se presenta en la Figura 2.6.

Figura 2.6: Esquema del cruzamiento de un punto.

El operador tradicional de mutación introduce diversidad en el mecanismo evolutivo, simplemente modificando aleatoriamente uno de los valores binarios del cromosoma. Sobre un esquema de codificación binaria la modificación consiste en invertir el valor binario de un alelo, y por ello recibe el nombre de mutación de inversión del valor de un bit (*flip-bit*); su operativa se ejemplifica en la Figura 2.7.

Figura 2.7: Esquema de la mutación de inversión del valor de un alelo.

Tanto la recombinación como la mutación son operadores probabilísticos, en el sentido en que se aplican o no, teniendo en cuenta una tasa de aplicación del operador. Generalmente la tasa de aplicación del operador de cruzamiento es elevada en un algoritmo genético simple (entre 0,5 y 0,9) mientras que la tasa de aplicación del operador de mutación es muy baja, del orden de 0,001 para cada bit en la representación.

Operadores evolutivos más complejos han sido propuestos como alternativas para modificar el comportamiento del mecanismo de exploración del espacio de soluciones. Es habitual encontrar operadores de cruzamiento ***multipunto*** en donde se utilizan dos o más puntos de corte (la operativa del cruzamiento de dos puntos se ejemplifica en la Figura 2.8) o ***uniformes*** en donde para cada posición en el cromosoma se decide intercambiar material genético de acuerdo a una probabilidad prefijada (su operativa se ejemplifica en la Figura 2.9).

Figura 2.8: Esquema del cruzamiento de dos puntos.

Figura 2.9: Esquema del cruzamiento uniforme.

2.4.4. Modelos de evolución

En el modelo tradicional de algoritmos genéticos el paso básico de evolución consiste en el reemplazo de la totalidad de la población por sus descendientes en cada generación (*modelo generacional*). Otros modelos diferentes de evolución han sido propuestos por los investigadores en el área. Entre ellos, es posible destacar el modelo *de estado estacionario* (Whitley, 1991). En este paradigma se crea un único nuevo individuo en cada generación, tratando de balancear el compromiso entre exploración (dada por el conjunto de individuos que forman la población) y explotación (realizada por el mecanismo de generación del único individuo creado en cada generación, generalmente utilizando técnicas de elitismo) para la resolución del problema. Un modelo intermedio entre el tradicional y el de estado estacionario es el *modelo de gap generacional* (De Jong, 1981), en el cual un porcentaje de la población (denominado *gap*) se genera en cada paso evolutivo.

Otros modelos de evolución más complejos han sido formulados para los algoritmos genéticos (modelos jerárquicos, no heterogéneos, híbridos, inspirados en los métodos de las estrategias de evolución). Estos modelos se diferencian de los tradicionales, incluyendo complejidades inherentes al mecanismo evolutivo que proponen. En este trabajo presentaremos una de las variantes no tradicionales de algoritmos genéticos, el algoritmo CHC (*Cross generational elitist selection, Heterogeneous recombination and Cataclysmic mutation*) propuesto por Eshelman (1991).

2.4.5. Formalización del mecanismo de los algoritmos genéticos

La teoría tradicional de exploración del espacio de soluciones por parte del mecanismo evolutivo de los algoritmos genéticos fue formalizada por Holland en 1975. Holland definió el concepto de *esquema*, para representar a conjuntos de individuos con características comunes de codificación y enunció el *teorema de los esquemas* para formalizar el muestreo de hiperplanos del espacio de soluciones. Como consecuencia de este teorema se propuso la denominada *hipótesis de los building blocks* que fundamenta el éxito del mecanismo de exploración de los algoritmos genéticos en el número exponencial de muestras que reciben aquellos esquemas cortos, con pocas posiciones definidas, pero que tienen un valor promedio de fitness superior al del resto de esquemas de la población. El mecanismo de recombinación, fundamental en el proceso evolutivo de un algoritmo genético, tenderá a crear individuos cada vez más aptos, perpetuando aquellas características adecuadas y combinándolas con otras de similar calidad (Holland, 1975). Trabajos teóricos posteriores han complementado las estimaciones originales de Holland (Whitley, 1997), e inclusive se han presentado formalizaciones matemáticas exactas del mecanismo (Poloni, 1999, 2000), aunque también ofrecido puntos de vista alternativos y críticos a la *hipótesis de los building blocks* (Thornton, 1998).

Goldberg (1991) demostró que el tiempo que un algoritmo genético emplea en converger hacia una solución única depende del tamaño de la población considerada. Utilizando los mecanismos de selección adecuados –aquellos que favorecen la reproducción de los mejores individuos–, el tiempo de convergencia del algoritmo genético es del orden de O($n \log n$), siendo n el tamaño de la población utilizada.

Adicionalmente a sus estudios sobre la formación de bloques básicos de construcción de soluciones, Holland se percató que el mecanismo evolutivo de los algoritmos genéticos tiene una característica denominada ***paralelismo intrínseco***, mediante la cual el ciclo evolutivo procesa un número mayor de esquemas que la cantidad de individuos presentes en la población. En efecto, trabajando sobre una población de n individuos, y por tanto requiriendo un esfuerzo computacional para realizar solamente n evaluaciones de la función de fitness, el algoritmo genético procesa útilmente O(n^3) esquemas en cada generación. Por este motivo, al examinar un elevado número de secciones del espacio de soluciones, el mecanismo de los algoritmos genéticos logra compensar la disrupción de esquemas de mayor largo y de alto número de posiciones definidas cuando se aplican los operadores evolutivos de cruzamiento y mutación.

2.4.6. Comparación de los algoritmos genéticos con otras técnicas heurísticas

Si bien no existe una opinión unánime sobre la aplicabilidad de los algoritmos genéticos, la amplia gama de problemas resueltos exitosamente en diversas áreas han popularizado a esta técnica evolutiva como una de las más versátiles y robustas. Las opiniones de los investigadores en el área concuerdan en señalar a los algoritmos genéticos como altamente útiles para resolver problemas con espacio de soluciones de dimensión elevada o no muy bien comprendido de antemano, donde el diseño de operadores específicos de ***hill-climbing*** no sea sencillo, en problemas multimodales donde los métodos heurísticos tradicionales pueden quedar atrapados en óptimos locales o en casos donde no es necesario obtener una solución óptima al problema, sino que una buena solución aproximada sería suficiente.

Los algoritmos genéticos presentan ciertas características que los hacen ventajosos cuando se los compara con otras técnicas de resolución de problemas. La operativa del mecanismo evolutivo de los algoritmos genéticos es independiente de particularidades del dominio, permitiendo definir esquemas genéricos capaces de abordar diferentes clases de problemas. Esta característica, conjuntamente con el hecho de que el mecanismo evolutivo se aplique sobre representaciones, hace a los algoritmos genéticos aplicables a una amplia gama de problemas.

Además, los algoritmos genéticos no son sensibles a efectos de no linearidad de la función a optimizar o características del problema que afectan a algoritmos estándar de optimización que utilizan información adicional, como las técnicas de hill-climbing o algoritmos que asumen aspectos de linearidad, convexidad, diferenciabilidad u otras características sobre la función a optimizar. Estas característica brindan a los algoritmos genéticos una significativa robustez de funcionamiento y de aplicabilidad.

Desde el punto de vista de la resolución del problema, el algoritmo genético funciona como una caja negra que solamente utiliza como dato de entrada la función de fitness definida para el problema para evaluar a los individuos en el ciclo evolutivo. No utiliza información adicional sobre las características del problema, aunque ciertas particularidades de la codificación pueden complementar la operativa simple de los operadores evolutivos y dirigir la búsqueda. La independencia del mecanismo de búsqueda evolutivo de las características del problema permite a los algoritmos genéticos evitar, en ocasiones, el estancamiento en mínimos locales del problema en los cuales son proclives a caer ciertos algoritmos tradicionales basados en gradientes u otras búsquedas locales dirigidas.

Una descripción detallada del mecanismo operativo de los algoritmos genéticos, que complemente los breves detalles que se resumen en este capítulo, puede encontrarse en los textos de Goldberg (1989a) o de Mitchell (1996).

2.5. Conclusiones

En los últimos quince años, las técnicas de computación evolutiva se han consolidado como un conjunto de metaheurísticas efectivas y robustas como consecuencia de su alta aplicabilidad a la resolución de problemas en múltiples áreas.

Tomando en cuenta los numerosos y exitosos antecedentes que refieren a la utilización de técnicas evolutivas para la resolución de problemas de optimización combinatoria, surgen como una alternativa natural para la resolución de los complejos problemas de optimización subyacentes a los problemas de diseño de redes de comunicaciones confiables, que constituyen el objetivo de esta Tesis.

Este capítulo ha presentado los conceptos genéricos vinculados con las técnicas de computación evolutiva y ha comentado en detalle las características de los algoritmos genéticos en particular, al considerarlos como mecanismos de resolución de problemas de optimización combinatoria y problemas análogos en otras áreas de aplicación.

2.6. Nota sobre las referencias bibliográficas

Las referencias bibliográficas mencionadas en este capítulo son muy diversas y varias de ellas han sido tomadas de citas de los textos de Goldberg (1989a), Mitchell (1996) y Dyson (1999). Otros documentos son citados y comentados en las reseñas de Angeline (1998) y Mitchell (1998) y en el *Handbook of Evolutionary Computation* (Back et al, 1997). Se presentan a continuación los detalles de las referencias, indicándose las citas correspondientes en aquellos casos de artículos y textos no consultados directamente.

CAPÍTULO 3

ALGORITMOS GENÉTICOS Y TÉCNICAS DE PROGRAMACIÓN DE ALTA PERFORMANCE

"Civilization advances by extending the number of important operations that we can perform without thinking about them"

ALFRED NORTH WHITEHEAD

An Introduction to Mathematics, 1911

3.1. **Introducción**

Los algoritmos genéticos se han consolidado como una de las técnicas de computación evolutiva más difundidas en los últimos años, como consecuencia de su utilidad para resolver un amplio rango de problemas en diversas áreas. Sin embargo, en muchos casos la aplicabilidad de las técnicas evolutivas se ve reducida por el gran tamaño o complejidad de los problemas a abordar, que requieren la utilización de poblaciones con numerosos individuos o la evaluación de costosas funciones de fitness.

Para resolver tales problemas de eficiencia, en los últimos veinte años se han utilizado las técnicas de procesamiento de alta performance. Estas técnicas se aplican sobre el modelo tradicional de los algoritmos genéticos, permitiendo diseñar modelos paralelos que mejoran la eficiencia de los algoritmos genéticos.

Este capítulo ofrece una descripción de las técnicas de programación de alta performance y las propuestas de su aplicación con el objetivo de mejorar el desempeño y calidad de resultados de los algoritmos genéticos. El capítulo comienza con una introducción a la programación de alta performance, que presenta detalles básicos del procesamiento paralelo y distribuido y las principales técnicas genéricas existentes para la resolución de problemas. A continuación se ofrecen las propuestas de su aplicación a los algoritmos genéticos, incluyendo una reseña de la evolución histórica en el diseño y clasificación de algoritmos genéticos paralelos. El capítulo culmina con un panorama del estado actual en el área, presentando una visión unificadora de las diferentes propuestas y nomenclaturas utilizadas.

3.2. Técnicas de programación de alta performance

En la actualidad, un ambiente de computación común consiste en un conjunto de estaciones de trabajo interconectadas por una red de área local. En los últimos años, las computadoras han incrementado su capacidad de cálculo, y las redes han crecido en velocidad de transmisión de datos hasta el estado actual, en que dicho ambiente de trabajo puede ser considerado como una única gran fuente de recursos. Los programas que pretendan aprovechar al máximo estos ambientes no pueden ser escritos usando las técnicas tradicionales de programación, dado que éstas fueron diseñadas para programas que se ejecutan en un único procesador en un mismo instante de tiempo, y no en varios procesadores trabajando concurrentemente y comunicándose a través de la red.

Las técnicas de programación de alta performance, más comúnmente llamadas *técnicas de procesamiento paralelo* o *técnicas de paralelismo*, constituyen un conjunto de métodos computacionales utilizados para la resolución de problemas sobre sistemas con una capacidad de procesamiento superior al tradicional modelo de computadora de Von Neumann. Estos sistemas, conocidos como *multiprocesadores*, permiten abordar la resolución computacional de problemas demasiado complicados –de acuerdo a la complejidad computacional inherente al propio problema, de acuerdo a la dimensión de las instancias consideradas, o debido a que trabajan sobre un enorme volumen de datos– que serían virtualmente imposibles de resolver mediante el paradigma tradicional. La característica principal de los sistemas multiprocesadores consiste en disponer de un conjunto de unidades de procesamiento interconectadas por algún medio que posibilita la comunicación de datos y de control entre ellos.

Las técnicas de procesamiento paralelo especifican paradigmas para el diseño de programas capaces de tomar ventaja de los múltiples recursos computacionales existentes, estableciendo un modelo robusto que permita al programador independizarse de la arquitectura subyacente. A continuación se presenta una descripción de las arquitecturas de hardware, las técnicas de programación que permiten el diseño, implementación y ejecución de programas paralelos y distribuidos y las medidas utilizadas para evaluar la performance de las aplicaciones.

3.2.1. Arquitecturas de computadores paralelos

Desde el punto de vista del hardware, una variada gama de modelos han sido propuestos para los sistemas multiprocesadores. Una de las diferencias principales entre los modelos de arquitecturas paralelas está dada por el hecho de si los procesadores comparten o no una misma memoria para almacenamiento de datos. De acuerdo a esta clasificación, existen dos grandes familias de arquitecturas paralelas: con memoria compartida y con memoria distribuida.

Un multiprocesador con memoria compartida puede verse como una clásica computadora de Von Neumann (Tanenbaum, 1997) a la cual se le han agregado procesadores adicionales. Todos los procesadores tienen acceso, para lectura y escritura de datos, a un mismo espacio de memoria al cual se accede a través de un bus de datos de uso común. Cada procesador opera asincrónicamente respecto al resto, y cualquier mecanismo de sincronización necesario debe realizarse de forma explícita. Toda comunicación, ya sea de datos o de control entre los procesadores se realiza a través del recurso compartido, la memoria común. La Figura 3.1 presenta gráficamente el concepto de arquitectura paralela de memoria compartida.

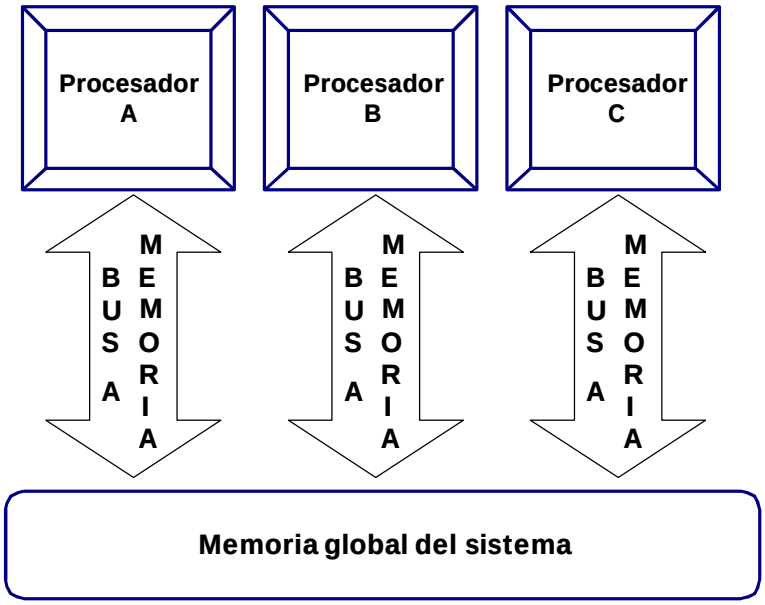


Figura 3.1: Arquitectura paralela de memoria compartida

La virtud más destacable de las arquitecturas de memoria compartida reside en el hecho de que la comunicación entre los procesadores es muy rápida, dado que se comunican a través del bus de datos. Sin embargo, ésta característica puede ser negativa para el funcionamiento del multiprocesador, ya que el bus de uso común puede convertirse en un cuello de botella en caso de requerirse un gran número de comunicaciones de datos. Además, la complejidad electrónica en el diseño de estos sistemas paralelos crece en forma exponencial con el número de procesadores. Por consideraciones de desempeño, para evitar el acceso continuo a la memoria global, en general cada procesador dispone de una memoria caché independiente. El problema de mantener cada caché local consistente con los cachés de los demás procesadores y con la memoria compartida hace que la electrónica de las placas madre sea muy compleja y poco escalable. Este problema, conocido como problema de coherencia del caché limita en la práctica la cantidad máxima de procesadores disponibles en este tipo de arquitecturas.

En los sistemas multiprocesadores con memoria distribuida, no se utiliza el concepto de una memoria global, accesible por todos los procesadores. El modelo introduce para cada procesador una memoria privada propia, no visible por el resto de los procesadores. La comunicación y sincronización entre procesadores se realiza por medio de pasaje de mensajes explícitos. En esta arquitectura no se dispone de un bus de datos compartidos que se pueda transformar en un cuello de botella, lo cual permite evitar el problema de coherencia del cache. Como contrapartida, los tiempos de comunicación son mayores que en el caso de una arquitectura con memoria compartida.

Esta arquitectura paralela es altamente escalable, admitiendo modelos con cientos y hasta miles de procesadores, y por ello es aplicable para problemas de gran envergadura, donde es posible definir un particionamiento adecuado para utilizar criteriosamente su gran poder de cómputo. Existen máquinas comerciales con esta arquitectura que llegan a los miles de procesadores. Como ejemplo puede citarse el sistema *Earth Link Simulator 5* (Top500, 2003), la máquina paralela más potente del mundo al momento de escribir esta tesis, en octubre de 2003. Este formidable equipo reúne 640 procesadores vectoriales (cada uno compuesto por 8 elementos de procesamiento) interconectados por una red de switches de alta velocidad. El poder teórico de procesamiento de este equipo está estimado en 40 Tflops.

La Figura 3.2 resume gráficamente las características de una arquitectura paralela basada en el concepto de memoria distribuida.

Un caso particular de multiprocesador de memoria distribuida lo constituyen los *clusters* o redes de computadores interconectados por redes de transmisión de datos de baja o mediana velocidad. Si bien la existencia de un recurso compartido como lo es la red de comunicación, introduce una desventaja con otros modelos desde el punto de vista de la eficiencia, esta arquitectura paralela se ha popularizado en la última década, como consecuencia de sus dos ventajas principales: puede diseñarse utilizando recursos de hardware de muy bajo costo ya existentes en cualquier organización (computadoras estándar y switches de redes de área local) y es altamente escalable, ya que para aumentar el poder computacional solo deben agregarse nuevas computadoras a la red. En los últimos años, importantes esfuerzos se han realizado para impulsar el modelo de clusters desde el contexto de las redes de área local a redes globales, posibilitando el uso de innumerables recursos computacionales conectados a través de Internet.

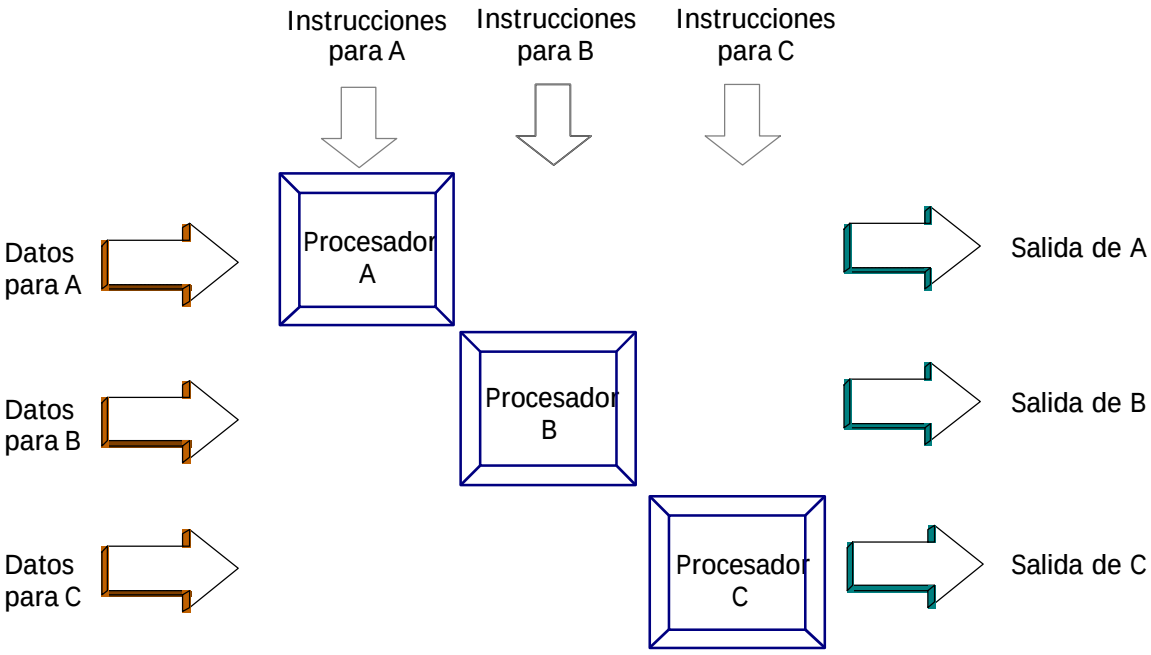


Figura 3.2: Arquitectura paralela de memoria distribuida

En nuestro entorno de trabajo (Centro de Cálculo e Instituto de Computación de la Facultad de Ingeniería, Universidad de la República, Uruguay) hemos empleado desde el año 1993 arquitecturas del tipo cluster de estaciones de trabajo para aprovechar las ventajas del procesamiento paralelo sobre una red de área local utilizando componentes de bajo costo. Los modelos de procesamiento utilizados en el marco de este trabajo se encuentran basados en esta infraestructura de hardware.

3.2.2. Clasificación de Arquitecturas Paralelas

Una taxonomía original de Flynn (1972) se ha aceptado como estándar para clasificar los tipos de arquitecturas paralelas por parte de la comunidad de investigadores en el área. El criterio utilizado en esta taxonomía se basa en dos características de los sistemas multiprocesadores, la ejecución de instrucciones y el flujo de datos sobre las cuales operan. Las combinaciones de estas características definen cuatro modelos, identificados por Flynn como SISD (*Single Instruction Single Data* – Una única instrucción y un único flujo de datos), SIMD (*Single Instruction Multiple Data* – Una única instrucción y un flujo múltiple de datos), MISD (*Multiple Instruction Single Data* – Múltiples instrucciones y un único flujo de datos) y MIMD (*Multiple Instruction Multiple Data* – Múltiples instrucciones y flujo múltiple de datos). La tabla de la Figura 3.3 presenta gráficamente la clasificación de Flynn.

	Instrucción única	***Instrucciones Múltiples***
Datos únicos	SISD	(MISD)
Datos Múltiples	SIMD	MIMD

Figura 3.3: Modelos de la clasificación de Flynn.

El modelo SISD identifica a una máquina con un solo procesador, y corresponde a la clásica arquitectura de Von Neumann donde en cada instante de tiempo se ejecuta una única instrucción sobre un único flujo de datos. Algunos computadores secuenciales modernos no corresponden estrictamente al modelo SISD. Como ejemplo, los procesadores RISC avanzados utilizan varios conceptos de las arquitecturas paralelas, como pipelining, ejecución paralela de instrucciones no dependientes, prefetching de los datos, etc. para lograr un incremento en la cantidad de operaciones por ciclo de instrucción. Pero la clasificación de Flynn no considera el paralelismo a nivel de instrucción, denominado paralelismo de grano fino, sino que se concentra en el paralelismo de grano grueso o a nivel de tareas.

El modelo SIMD identifica a los computadores paralelos conocidos como *procesadores vectoriales* o *arreglos de procesadores*, que disponen de una gran cantidad de elementos de procesamiento. En cada instante de tiempo cada procesador del arreglo ejecuta el mismo conjunto de instrucciones sobre diferentes conjuntos de datos asignados. Habitualmente este tipo de computadores paralelos se compone de un gran número de elementos de procesamiento capaces de realizar operaciones sencillas, por lo general operaciones aritmético-lógicas a nivel de bit, de byte o de palabra. Al conformarse de procesadores sencillos, es posible colocar una gran cantidad por chip. Los procesadores manejados por una única unidad de control que se encarga de la ejecución sincrónica de cada única instrucción en un instante de tiempo determinado. El modelo es útil para el diseño de aplicaciones paralelas uniformes, donde se aplica un mismo algoritmo de modo sincrónico sobre un gran volumen de datos. La aplicabilidad de este modelo queda limitada por las comunicaciones entre procesadores, usualmente predefinidas de acuerdo a una topología sencilla.

La categoría de computadores MIMD es la más difundida, debido a que es la que mejor permite implementar los modelos de memoria distribuida y explotar las ventajas de la escalabilidad de recursos de bajo costo. Los diferentes procesadores disponen de una autonomía de funcionamiento, ejecutando asincrónicamente en cada instante de tiempo diferentes instrucciones de su código sobre diferentes bloques de datos. El modelo incluye diferentes unidades de control que operan a los procesadores. Al no existir una unidad de control única, los mecanismos de sincronización deben realizarse explícitamente. Si bien existen máquinas MIMD de memoria compartida, en general los procesadores trabajan con una memoria local de modo que el sistema en su conjunto tiene memoria distribuida. La sincronización en este caso se realiza de modo explícito, mediante pasaje de mensajes.

El modelo MISD corresponde a la combinación de múltiples instrucciones ejecutando sobre un solo dato. Esta categoría es controvertida, e inclusive existen opiniones referentes a que no existen máquinas paralelas de este tipo. La clase MISD se compone de computadores de propósito específico que tienen buen desempeño para la resolución de problemas particulares. Como ejemplo de esta clase pueden citarse los *arrays sistólicos* utilizados en tareas de procesamiento de señales. De todos modos, la aplicabilidad de este modelo es muy poco significativa.

3.2.3. Paradigmas de Programación Paralela

Existen varios modelos de programación que permiten especificar, diseñar e implementar programas paralelos que ejecuten sobre los diferentes modelos de máquinas paralelas mencionados. En todos los casos, las tareas principales que enfrenta la programación paralela y distribuida consisten en *comunicar* y *sincronizar* un conjunto de procesos que se ejecutan concurrentemente en diferentes procesadores. Sin entrar en una descripción exhaustiva de todos los modelos y variantes aplicables a cada arquitectura de multiprocesador, se ofrece a continuación un panorama de las principales técnicas de programación distribuida. Se profundiza en el modelo basado en la técnica de pasaje de mensajes aplicado a multiprocesadores de memoria distribuida, que constituye la infraestructura informática sobre la que se han basado los algoritmos evolutivos paralelos desarrollados en este trabajo.

Los distintos modelos de programación paralela se diferencian en ciertos aspectos inherentes al procesamiento paralelo−distribuido. Las principales diferencias consisten en el nivel de *granularidad* −que evalúa la cantidad de trabajo realizada por cada procesador−, el modo de aplicación de las instrucciones y los mecanismos utilizados para la comunicación entre procesos. A continuación se presenta un resumen de los modelos más populares, basado en el texto de Foster (1995).

El modelo de paralelismo de datos propone tomar ventaja de la concurrencia existente cuando se debe aplicar una misma operación sobre un conjunto múltiple de datos. Este modelo considera el menor nivel de granularidad posible, al interpretar cada instrucción como una tarea. En general se parte de una distribución de datos sobre los procesadores disponibles y a partir de esta información se diseña un programa adecuado para la ejecución sobre multiprocesadores del tipo SIMD. Bibliotecas de paralelismo como HPF (High Performance Fortran) basan su propuesta en este modelo.

Los modelos de programación de memoria compartida concentran su interés en el acceso asincrónico a un recurso compartido común, el espacio de direccionamiento. Los mecanismos de sincronización pasan a tener una relevancia fundamental en este modelo, para impedir los problemas resultantes del acceso simultáneo o la apropiación del recurso compartido.

El modelo de tareas y canales permite especificar programas paralelos de granularidad alta, a nivel de tareas. Cada tarea encapsula un programa secuencial ejecutando sobre una memoria local e interactuando con su entorno. Además de acceder a su memoria local, las tareas son capaces de enviar y recibir mensajes, crear nuevas tareas y finalizar su ejecución. El medio de comunicación entre tareas consiste en colas de mensajes denominados canales, que pueden ser creados o eliminados dinámicamente. Este modelo de programación ofrece una independencia al modo en que las tareas se asignan a los procesadores disponibles. La abstracción de modelos permite especificar a alto nivel la semántica de programas paralelos.

El modelo de pasaje de mensajes se basa en la definición de tareas que se comunican utilizando primitivas explícitas a través de un medio de comunicación. El mecanismo de comunicación se basa en las operaciones básicas de envío y recepción mensajes, que permiten implementar la transferencia de datos y la sincronización en el funcionamiento de los procesos que ejecutan en diferentes procesadores. La diferencia con el modelo de tareas y canales consiste en que los canales de comunicación no son incluidos como objetos del modelo, y la creación de tareas no se incluye como actividad. Originalmente concebido para modelar el caso de copias de un único programa que ejecutan en varios procesadores, ha extendido su dominio de aplicación para especificar, diseñar e implementar cualquier programa paralelo para ejecución sobre arquitecturas MIMD.

3.2.4. Técnicas y herramientas para el diseño e implementación de aplicaciones paralelas

Se han propuesto varias técnicas de programación aplicables para implementar aplicaciones paralelas y distribuidas usando los modelos de programación presentados anteriormente, y en especial el paradigma de pasaje de mensajes. Las técnicas más difundidas son la técnica de *Descomposición Funcional* y la técnica de *Descomposición de Dominio*.

La técnica de Descomposición Funcional se basa en la identificación de módulos que realizan operaciones independientes que pueden realizarse en paralelo. Esto determina que en cada instante de tiempo se ejecuten simultáneamente diferentes secciones o programas en varios procesadores, reduciendo el tiempo total de ejecución respecto a la aplicación ejecutada de modo serial. Los conceptos de particionamiento de tareas y modularización tienen una importancia vital ara esta técnica de programación.

En la técnica de Descomposición de Dominio, la aplicación paralela ejecuta en los diferentes procesadores la misma porción de código, pero trabajando concurrentemente sobre diferentes conjuntos de datos. Las estrategias de particionamiento de datos, asignación entre datos y procesos y los mecanismos de comunicación entre procesos que trabajan con regiones de datos vecinas son las principales claves sobre la que se cimenta esta técnica.

Con respecto a las herramientas disponibles para el diseño de aplicaciones paralelas, es posible distinguir entre cuatro familias, que se describen a continuación.

El programador de aplicaciones paralelas y distribuidas dispone de mecanismos de comunicación entre procesos a partir de la introducción de la interface de sockets BSD en los sistemas operativos derivados del kernel UNIX original, como UNIX System V, Solaris de Sun, AIX de IBM, y HP-UX. Las primitivas de socket permiten la comunicación entre procesos sobre los protocolos UDP y TCP, brindando el soporte lógico para resolver los problemas básicos de la programación paralela y distribuida. Sin embargo, la programación con sockets ofrece una solución de bajo nivel que implica al programador conocer las características de la red de comunicación, definir las estructuras de datos de los mensajes y coordinar las interacciones entre los procesos que se comunican. Este tipo de programación de bajo nivel puede ser adecuado para sistemas de pequeño porte, pero sus características, complejidades y limitaciones lo hacen inadecuado para el desarrollo de grandes aplicaciones paralelas.

Las bibliotecas basadas en pasaje de mensajes ofrecen un mayor nivel de abstracción para implementar la comunicación entre procesos paralelos, resolviendo los problemas de manipulación de tipos de datos y representaciones, tolerancia a fallos, mecanismos de seguridad, entre otros. Las bibliotecas presentan de modo integrado variadas rutinas para la transferencia de datos y sincronización entre procesos que abstraen al programador de las complejidades inherentes a los mecanismos de bajo nivel involucrados (sockets, protocolos de transporte, etc.). Como ejemplo de este tipo de bibliotecas pueden mencionarse *PVM - Parallel Virtual Machine* (Geist et al, 1994) y *MPI - Message Passing Interface* (Gropp et al, 1999). En la siguiente subsección se ofrece una breve descripción de la biblioteca de pasaje de mensajes MPI, utilizada en la implementación de algoritmos evolutivos paralelos en el marco de este proyecto. En nuestro entorno de trabajo hemos utilizado ambas bibliotecas como herramientas para el diseño de programas paralelos y distribuidos en diferentes campos de aplicación por varios años. Como consecuencia, se ha acumulado una gran experiencia, tanto en el desarrollo específico de aplicaciones paralelas y distribuidas, como en la modificación de las propias bibliotecas para lograr objetivos específicos al utilizarlas sobre un conjunto de estaciones de trabajo interconectadas por una red de área local.

El mecanismo de invocación a procedimientos remotos (RPC) consistió en uno de los primeros prototipos de comunicación entre aplicaciones. Este mecanismo posibilita la transferencia sincrónica de datos y control entre dos procesos que no comparten un espacio de direccionamiento común. Los mecanismos basados en colas de mensajes ofrecen una implementación más robusta para comunicación sincrónica entre procesos.

En los últimos años, los modelos de invocación de métodos remotos (RMI), los modelos de objetos distribuidos (CORBA), y los modelos basado en servicios (webservices, XML, SOAP) han proporcionado nuevas alternativas para la comunicación entre procesos que ejecutan sobre diferentes procesadores, en especial en el caso de procesos que ejecutan en entornos de programación distribuidos. En la actualidad, un nuevo nivel en el desarrollo de aplicaciones paralelas se encuentra en desarrollo. El modelo de Grid Computing, propone el uso de protocolos de almacenamiento de datos, gestión de recursos, autenticación y seguridad, etc. posibilitando la computación a alto nivel entre entornos heterogéneos.

3.2.5. La biblioteca MPI

MPI es una biblioteca para la programación de aplicaciones paralelas utilizando el paradigma de comunicación de procesos mediante pasaje de mensajes. El estándar MPI fue definido en 1992 por un conjunto de desarrolladores de software y aplicaciones científicas en coordinación con importantes empresas (IBM, Intel, PVM, nCUBE) para proporcionar un modelo de biblioteca implementable sobre una variada gama de arquitecturas multiprocesador para el diseño de aplicaciones distribuidas portables y eficientes (MPI Forum, 1992). Las características principales de la biblioteca se presentan a continuación:

- Tiene como objetivo principal el trabajo sobre plataformas de memoria distribuida, aunque incluye primitivas para trabajo sobre plataformas de memoria compartida.
- Trabaja sobre la idea de que el paralelismo es explícito, en el sentido de que es definido y controlado en su totalidad por el programador de aplicaciones.
- Considera como único mecanismo de comunicación entre procesos el pasaje de mensajes explícito.
- Propone como mecanismo de diseño de programas el modelo SPMD (copias de un único programa ejecutando asincrónicamente sobre diferentes conjuntos de datos), aunque puede adaptarse para diseñar bajo un modelo MPMD (Multiple Program Multiple Data – múltiples programas ejecutando concurrentemente sobre diferentes conjuntos de datos).

MPI provee soporte para el diseño de programas paralelos y distribuidos cuyos componentes se comunican mediante invocaciones a funciones para recepción y envío de mensajes desde y hacia otros procesos. El conjunto de procesos (cuyo número es fijo y definido al momento de la creación) pueden utilizar una variada gama de primitivas de comunicación punto a punto o colectivas, sincrónicas o asincrónicas, bloqueantes o no bloqueantes de acuerdo a las necesidades de la lógica de la aplicación paralela. Las estructuras de comunicación se encapsulan en objetos denominados comunicadores, que abstraen al programador de las complejidades de los mecanismos de pasaje de mensajes. Desde su definición varias implementaciones del estándar MPI han sido desarrolladas por diversas organizaciones, a tales efectos pueden citarse las implementaciones LAM, MPICH, CHIMP, etc.

En el caso de los algoritmos desarrollados en este trabajo, tanto la propuesta de un algoritmo genético paralelo específico para el Problema de Steiner Generalizado que se presenta en el Capítulo 5 como la biblioteca utilizada para el diseño de los algoritmos evolutivos que se presentan en el Capítulo 6 se han implementado utilizando la implementación MPICH de la biblioteca MPI.

3.2.6. Medidas de performance

El principal objetivo para utilizar las técnicas de alta performance consiste en mejorar el *desempeño* de una aplicación determinada. Sin embargo, el desempeño de una aplicación es un concepto complejo y polifacético. Además de la medida tradicionalmente utilizada para evaluar la eficiencia computacional de un programa, su *tiempo de cómputo*, otros múltiples factores deben considerarse cuando se intenta medir el desempeño de una aplicación paralela. A modo de ejemplo es posible citar los mecanismos de almacenamiento de datos en dispositivos y las medidas relacionadas con la transmisión de datos entre procesos, dos factores que influyen directamente en el tiempo total de ejecución de una aplicación paralela.

Por otra parte, los aspectos de utilización de los recursos disponibles y en especial la capacidad de utilizar mayor poder de cómputo para resolver problemas más complejos o de mayor dimensión son dos de las características más deseables para las aplicaciones paralelas.

En el contexto de este trabajo se utilizará el tiempo total de ejecución como medida del desempeño de los algoritmos paralelos diseñados, tomando en cuenta que es una medida simple de evaluar, y resulta útil para evaluar el esfuerzo requerido para la resolución de un problema de optimización por parte de un algoritmo evolutivo. A continuación se presentan los aspectos principales del modelo de evaluación de desempeño y sus conceptos relacionados, basado en el texto de Foster (2003).

3.2.6.1. Modelo de performance

El modelo de performance propuesto considera a cada métrica, en este caso ejemplificaremos con el tiempo total de ejecución T, como función del tamaño del problema a resolver (N), el número de procesadores disponibles (P), el número de tareas o procesos (U), y otras variables dependientes del propio algoritmo considerado y de las características del hardware sobre el cual se ejecuta. Una expresión genérica se presenta en la Ecuación 3.1.

$$T = f(N, P, U, \mathbf{K})$$

Ecuación 3.1: Expresión genérica para la métrica tiempo de ejecución.

Formalmente, el tiempo de ejecución de un programa paralelo se define como el tiempo que ocurre entre que el primer proceso comienza su ejecución hasta que el último proceso finaliza su ejecución. Durante la ejecución, cada proceso realizará diversas actividades, pasando por diferentes estados, como el estado de procesamiento o cómputo efectivo, el estado de comunicación de información y el estado ocioso. Por lo tanto, el tiempo total de ejecución se compone de la suma del tiempo de procesamiento (T_{PROC}), más el tiempo destinado a comunicaciones (T_{COM}), más el tiempo en estado ocioso (T_{IDLE}), tal como se indica en la Ecuación 3.2.

$$T = T_{PROC} + T_{COM} + T_{IDLE}$$

Ecuación 3.2: Componentes del tiempo total de ejecución.

Considerando P tareas ejecutando en P procesadores, el tiempo total de ejecución puede descomponerse de acuerdo a la expresión de la Ecuación 3.3, donde T^{i}_{ETAPA} indica el tiempo insumido en el procesador i en la etapa correspondiente (valores que es posible obtener de las estadísticas y reportes sobre la utilización de recursos en los computadores utilizados).

$$T = \frac{1}{P}\left(\sum_{i=1}^{P} T^{i}_{PROC} + \sum_{i=1}^{P} T^{i}_{COM} + \sum_{i=1}^{P} T^{i}_{IDLE} \right)$$

Ecuación 3.3: Tiempo total de ejecución discriminado por tareas y etapas.

Sobre los componentes del tiempo total de ejecución es posible indicar que:

- El tiempo de cómputo necesario para completar una tarea obviamente depende directamente de la complejidad y de las dimensiones del problema a resolver, pero otros factores como el número de tareas utilizadas y la cantidad de elementos de procesamiento disponibles también lo afectan significativamente. Cuando se utilizan máquinas paralelas en entornos distribuidos, la heterogeneidad del equipamiento disponible afecta directamente los tiempos de cómputo. La importancia de este aspecto debe contemplarse especialmente al utilizar redes de computadores heterogéneos como plataforma de ejecución de aplicaciones paralelo–distribuidas. De este modo, las características de los procesadores, tales como su capacidad de cómputo, la memoria disponible, la posibilidad de uso de caché, etc., son elementos que deben tomarse en cuenta para no asumir un modelo simplificado para el tiempo de cómputo.
- El tiempo de comunicación depende de la localidad, es decir de la ubicación física, de los procesos o tareas que componen la aplicación distribuida. Usualmente se distingue entre comunicación *interprocesador*, cuando se comunican tareas que ejecutan en elementos de procesamiento diferentes y comunicación *intraprocesador*, cuando se comunican tareas que ejecutan en el mismo elemento de procesamiento. Este último tipo de comunicación se lleva a cabo de modo más veloz e inclusive es posible tomar ventajas de las características de la arquitectura en equipos multiprocesadores, optimizando de modo considerable los tiempos de comunicación. Por otra parte, los tiempos de comunicación entre tareas que ejecutan en elementos de procesamiento diferentes son generalmente más elevados, en especial si se trabaja sobre computadores paralelos virtuales de memoria distribuida, que utilizan medios poco eficientes para la comunicación, como lo son las redes de área local.

  En un modelo idealizado de arquitectura paralela de memoria distribuida, el costo de comunicar dos tareas ubicadas en diferentes computadores tiene dos componentes: el tiempo necesario para el establecimiento de la conexión, que es un parámetro de la red de comunicaciones denominado *latencia* y el tiempo de transferencia de información, usualmente determinado por el ancho de banda disponible en el canal físico de comunicación que conecta a los equipos origen y destino. La Ecuación 3.4 presenta el tiempo requerido para enviar un mensaje de L bits de largo, donde T_{TR} indica el tiempo necesario para transferir un bit.

$$T = latencia + L \,.\, T_{TR}$$

Ecuación 3.4: Tiempo de comunicación de un mensaje.

- El tiempo que una tarea permanece en estado ocioso es consecuencia del orden no determinístico en el cual se ejecutan las operaciones por parte de una aplicación paralela compuesta por varias tareas. Minimizar el tiempo ocioso de las tareas debe ser un objetivo directo del diseñador de algoritmos paralelos. Una tarea puede estar ociosa básicamente por uno de los siguientes motivos: la ausencia de recursos de cómputo disponibles o la ausencia de datos sobre los cuales operar. El primer caso puede intentar evitarse utilizando técnicas de balance de cargas para distribuir del modo más adecuado los requerimientos de cómputo entre los elementos de procesamiento disponibles. Por su parte, la no–disponibilidad de datos puede ser consecuencia de un mal diseño del programa o de que el equipamiento requiere elevados tiempos para la comunicación de datos. Técnicas conocidas como de *superposición de cómputo y comunicación* pueden aplicarse para rediseñar el programa de modo de tratar de evitar estas situaciones.

3.2.6.2. Speedup y Eficiencia de un algoritmo paralelo

De los argumentos mencionados anteriormente es posible concluir que el tiempo de ejecución, considerado aisladamente, no es siempre una buena métrica para evaluar el desempeño de aplicaciones paralelas y distribuidas. Dado que el tiempo de ejecución varía con el tamaño del problema y con los recursos disponibles, los tiempos de ejecución deben normalizarse de algún modo para comparar el desempeño obtenido al resolver problemas de tamaños diferentes y al utilizar diferentes números de procesadores. Por este motivo habitualmente se utilizan dos métricas relativas para evaluar el desempeño de un algoritmo paralelo, que utilizan los tiempos de ejecución como base: el ***speedup*** y la ***eficiencia***.

El speedup es una medida de la mejora de rendimiento obtenida al utilizar un algoritmo paralelo ejecutando sobre un número determinado de procesadores con respecto al mejor algoritmo serial –considerando su tiempo de ejecución, o sea el más rápido– conocido que resuelve el problema. La Ecuación 3.5 presenta la definición de la medida que se conoce como speedup ***exacto*** o ***absoluto***. T_{SERIAL} indica el tiempo del mejor algoritmo serial conocido y T_M el tiempo del algoritmo paralelo ejecutando sobre M procesadores.

$$S_M = \frac{T_{SERIAL}}{T_M}$$

Ecuación 3.5:Definición de speedup absoluto.

Dado que para muchos problemas no se conoce un "mejor algoritmo serial" que lo resuelva, es habitual trabajar con una definición alternativa de speedup, conocida como ***speedup algorítmico*** (conocido también como ***speedup relativo***) que evalúa el rendimiento de un algoritmo paralelo al aumentar la cantidad de procesadores utilizados, comparado con el rendimiento obtenido al utilizar un solo procesador. Esta definición considera al tiempo del propio algoritmo paralelo ejecutando en un único procesador (T_1) en lugar del tiempo del mejor algoritmo serial, su definición se presenta en la Ecuación 3.6.

$$S_M = \frac{T_1}{T_M}$$

Ecuación 3.6:Definición de speedup algorítmico.

La eficiencia de un algoritmo paralelo corresponde a un valor normalizado del speedup, relativo al número de procesadores utilizados. Se define por la Ecuación 3.7, y dependiendo del modo de calcular el speedup se tendrá un valor de eficiencia ***absoluta*** o de eficiencia ***algorítmica*** o ***relativa***. En general los valores de eficiencia estarán entre 0 y 1, y los valores de eficiencia cercanos a 1 identificarán situaciones casi ideales de mejora de performance.

$$E_M = \frac{S_M}{M}$$

Ecuación 3.6:Definición de eficiencia.

En la determinación del speedup deben tenerse en cuenta los argumentos presentados en la sección precedente sobre la evaluación de los tiempos de ejecución, considerándose los factores que lo afectan, en especial las características de configuración de la máquina paralela utilizada.

Para lograr una comparación justa de los tiempos de ejecución, debe tenerse en cuenta que:

- Para calcular el speedup exacto o absoluto debe utilizarse efectivamente el mejor algoritmo serial disponible, o en su defecto el mejor algoritmo serial conocido, en caso de no existir una cota inferior para la complejidad del problema.
- Debe analizarse el hardware disponible, asignando eventualmente coeficientes (*pesos*) a los diferentes computadores en máquinas paralelas heterogéneas o utilizando los algoritmos diseñados para la evaluación del hardware (*benchmarks*)
- Debe tomarse en cuenta el asincronismo de las aplicaciones paralelas, por lo cual es necesario obtener las medidas mediante análisis estadísticos, considerando un número significativo de experimentos.

Considerando los aspectos mencionados, la situación ideal al utilizar un algoritmo paralelo es la de lograr el speedup lineal: al utilizar *P* procesadores obtener una mejora de factor *P* en el tiempo de ejecución. Aunque en ciertos casos particulares es posible obtener speedup *superlineal*, la situación habitual es que utilizando un algoritmo paralelo sobre *P* procesadores se obtiene una mejora de factor menor que *P*, situación conocida como de speedup *sublineal*. El gráfico de la Figura 3.4 ejemplifica estos conceptos, presentando un análisis genérico de los diferentes valores de speedup obtenidos en la ejecución de una aplicación paralela al utilizar diferente número de procesadores.

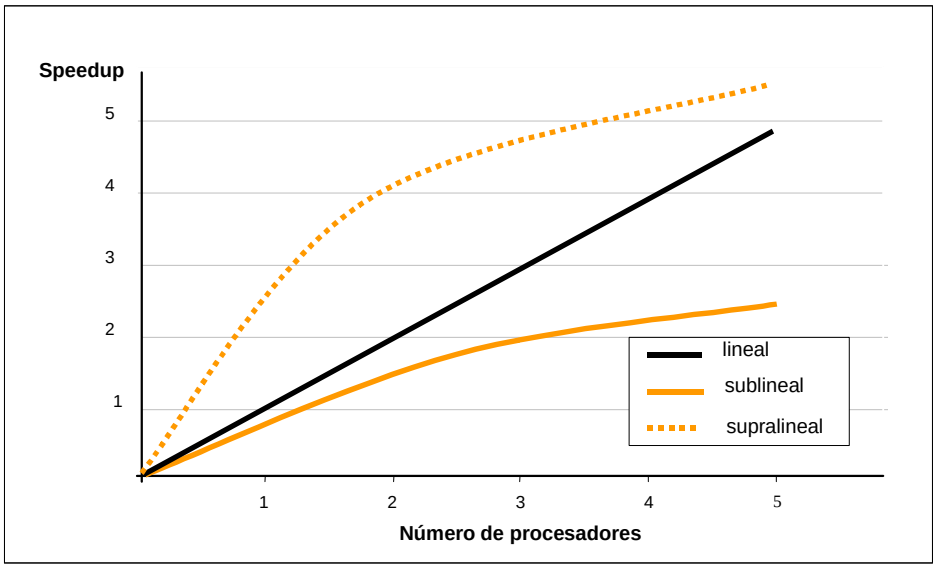


Figura 3.4: Speedup lineal, sublineal y supralineal.

Existen varios motivos que impiden el crecimiento lineal del speedup, entre los más notorios pueden citarse las demoras introducidas por las comunicaciones, el overhead en intercambio de datos, el overhead en trabajo de sincronización, los cuellos de botella en el acceso al hardware necesario, factores que degradan la performance al incrementar el número de tareas que componen a un sistema paralelo. Estos factores inclusive pueden producir que el uso de más procesadores sea contraproducente para el objetivo de lograr una mejora en el desempeño de la aplicación paralela.

Pero existe un factor clave, la existencia de tareas no paralelizables, que tiene un rol preponderante al momento de considerar las limitaciones a la aplicación de las técnicas de procesamiento paralelo. Este hecho fue reconocido por Amdahl (1967), quien observó que l*a parte serial de un programa determina una cota inferior para el tiempo de ejecución, aún cuando se utilicen al máximo técnicas de paralelismo.* Este importante resultado se conoce desde ese momento como *ley de Amdahl*.

Un análisis superficial de la ley de Amdahl parece proyectar una perspectiva sombría sobre la capacidad de mejorar el desempeño de aplicaciones aplicando las técnicas de paralelismo, considerando la existencia de una cota inferior para el tiempo de ejecución, cuyo valor fue estimado en $O(\log_2.N)$ por Minsky (1971). La visión de Minsky fue considerada muy pesimista por muchos investigadores y revisada por Hwang (1984) quien, utilizando análisis de modelos estadísticos, propuso una estimación de $O(N.\log_2.N)$ como cota superior posible para el speedup alcanzable. Una profundización en los conceptos vinculados con la ley de Amdahl proporciona una visión más optimista si se considera la complejidad de las tareas a resolver: ***la razón para agregar procesadores debe ser para resolver problemas más complicados o de dimensión mayor, y no para resolver más rápido un problema de tamaño fijo.*** La idea detrás del argumento anterior tiene el valor de tomar en cuenta la ***complejidad*** de las tareas realizadas. Considerando que habitualmente las tareas no paralelizables son de complejidad lineal ($O(N)$), como por ejemplo las lecturas de datos de entrada, mientras que los algoritmos paralelizables tienen una complejidad mayor, si se logra reducir el orden de complejidad del algoritmo mediante el uso de N procesadores, el speedup ideal puede alcanzar valores de $O(N)$. Este importante resultado fue presentado por Gustafson (1988), dando un nuevo impulso a la visión optimista del uso de las técnicas de procesamiento paralelo para la resolución de problemas complejos (Gustaffson et al, 1998).

El último argumento presentado nos conduce a un concepto fundamental en la aplicación de técnicas de procesamiento de alta performance: la ***escalabilidad*** de un algoritmo paralelo. La escalabilidad se define como la capacidad de utilizar los recursos computacionales disponibles para obtener un mejor rendimiento en la ejecución de aplicaciones paralelas y constituye una de las principales características deseables de los algoritmos paralelos y distribuidos.

3.2.6.3. Speedup y eficiencia en el contexto de los algoritmos evolutivos paralelos

Los conceptos presentados anteriormente han sido aplicados específicamente al contexto de los algoritmos evolutivos paralelos, explicando porqué las particularidades de estos algoritmos permiten obtener valores de speedup superlineal en ciertos casos. A continuación se presenta la definición de ***speedup*** en el contexto de algoritmos evolutivos paralelos que puede encontrarse en el artículo de Alba (2002) que se ha tomado como referencia.

El ***speedup*** mide la relación entre el tiempo medio de ejecución del algoritmo evolutivo ejecutando sobre un procesador y el tiempo medio de ejecución del algoritmo ejecutando sobre m procesadores. Corresponde a la expresión de la Ecuación 3.8, donde se ha notado por T_M el tiempo de ejecución de un algoritmo ejecutando sobre M procesadores. La introducción de los valores medios está motivada por la naturaleza no determinística de los tiempos de ejecución de programas paralelos en general, y el asincronismo que habitualmente caracteriza a los algoritmos evolutivos paralelos específicamente.

$$S_M = \frac{E[T_1]}{E[T_M]}$$

Ecuación 3.8: Definición de speedup para un algoritmo evolutivo paralelo.

Esta medida evalúa que tan eficiente resulta el algoritmo paralelo cuando se dispone de M recursos computacionales, referente a la eficiencia del algoritmo serial y tomando como medida el tiempo de ejecución de los procesos. Utilizando esta definición, se distingue entre los casos de ***speedup sublineal*** ($Speedup_M < M$) y ***speedup superlineal*** ($Speedup_M > M$).

La medida de ***eficiencia*** normaliza el valor del speedup a un porcentaje, correspondiendo el valor 100% al speedup lineal, de acuerdo a la expresión de la Ecuación 3.9.

$$E_M = \frac{S_M}{M}.100\%$$

Ecuación 3.9: Definición de eficiencia para un algoritmo evolutivo paralelo.

En problemas vinculados a otras áreas de investigación, uno de los objetivos principales al aplicar técnicas de procesamiento paralelo lo constituye el alcanzar valores cercanos al speedup lineal, propuesto generalmente como una cota superior para la eficiencia. En el área de los algoritmos evolutivos, la naturaleza de los modelos y las características de las plataformas de ejecución hacen frecuente que se pueda alcanzar valores de speedup superlineal, tal como se expone en Alba (2002).

Alba presenta una taxonomía de medidas de speedup aplicables específicamente para algoritmos evolutivos. A un primer nivel aplica la categorización de speedup ***fuerte*** y speedup ***débil***. El speedup ***fuerte*** corresponde a una versión del speedup absoluto tradicional, en la cual se compara el tiempo de ejecución del algoritmo evolutivo paralelo con el tiempo del mejor algoritmo conocido para la resolución del problema, aplique éste técnicas evolutivas o no. Como consecuencia de la dificultad práctica de encontrar el mejor algoritmo que resuelva un problema determinado, se introduce el speedup ***débil*** basándose en una analogía con el speedup algorítmico. Esta medida se define comparando el tiempo de la versión paralela del algoritmo con el tiempo de su versión secuencial, y es el criterio utilizado por la mayoría de los investigadores en el área. El autor distingue variantes dentro de la categoría de speedup ***débil***, el speedup con criterio de parada basado en la calidad de la solución obtenida, de acuerdo a sus argumentos el modo más justo de evaluar el speedup en algoritmos evolutivos paralelos, y el speedup con criterio de parada basado en esfuerzo prefijado.

Asimismo, se deja constancia de una diferencia importante al trabajar con algoritmos evolutivos, donde los modelos paralelos pueden tener un mecanismo de evolución diferente al de los modelos seriales. Mientras estos últimos no introducen restricciones a la interacción entre individuos (trabajan con el modelo conocido como de ***interacción panmíctica***), en un modelo paralelo pueden existir restricciones, siendo cada individuo capaz de interactuar solamente con un subconjunto distinguido de ***vecinos***. De este modo, debe diferenciarse el caso cuando se compara el tiempo de ejecución de un modelo paralelo con un modelo panmíctico, del caso donde se compara con el propio algoritmo paralelo ejecutando sobre un único procesador.

Considerando la clasificación presentada en el artículo de referencia mencionado anteriormente, en este trabajo se utiliza una versión de speedup algorítmico débil, dado que se comparan los tiempos de ejecución del algoritmo paralelo contra los de su versión panmíctica serial.

3.3. Técnicas de programación de alta performance aplicadas a los algoritmos genéticos.

Esta sección presenta los conceptos detrás de las propuestas de aplicación de las técnicas de programación de alta performance a los algoritmos genéticos. Se presentan los motivos de aplicación de las técnicas de paralelismo, los diferentes criterios utilizados por los investigadores para clasificar a los algoritmos genéticos paralelos y por último se ofrece una descripción de los principales modelos propuestos.

3.3.1. Motivos para la aplicación de las técnicas de procesamiento de alta performance a los algoritmos genéticos

A pesar de que el mecanismo de los algoritmos genéticos se basa en un proceso de evolución que es paralelo por naturaleza, el modelo clásico propuesto originalmente por Holland (1975) es secuencial, motivado por la falta de disponibilidad de arquitecturas paralelas al momento de su definición. Como consecuencia de esta restricción, algunos aspectos del funcionamiento del algoritmo genético clásico resultan difíciles o imposibles de paralelizar, como es el caso de la selección panmíctica utilizada para imitar el concepto de presión selectiva. Otros mecanismos no existentes en el algoritmo genético serial surgen al considerar los modelos paralelos, como el operador de migración entre poblaciones semi-independientes.

Las técnicas de procesamiento paralelo y distribuido se aplican al modelo clásico de los algoritmos genéticos con el objetivo de obtener mejoras tanto desde el punto de vista de la performance del mecanismo evolutivo como desde el punto de vista algorítmico, para mejorar la calidad de la búsqueda genética.

Analizando desde el punto de vista de la eficiencia, la paralelización permite afrontar la lentitud de la convergencia de los algoritmos genéticos cuando la dimensión del problema hace necesario utilizar poblaciones numerosas, o en los casos que requieren múltiples evaluaciones de funciones de fitness costosas. Conjuntamente a la posibilidad de obtener mejores tiempos de ejecución, mediante la aplicación de técnicas de procesamiento paralelo es posible obtener mejor precisión y eficiencia numérica en los resultados, tal como se describe en los trabajos de Gordon y Whitley (1993) y Hart et al. (1996), entre otros. Este hecho indica que la propia estructura algorítmica de los Algoritmos Genéticos Paralelos (AGP) constituye un tópico interesante, independientemente de su implementación sobre una arquitectura de alta performance específica.

Desde el punto de vista algorítmico, los algoritmos genéticos paralelos pueden explotar la naturaleza paralela intrínseca del mecanismo evolutivo, trabajando simultáneamente sobre varias poblaciones independientes o semi-independientes para resolver el mismo problema. Eventuales intercambios de soluciones (migraciones) permiten introducir la diversidad y variación necesaria para evitar los problemas de convergencia prematura en valores sub-óptimos. Complementariamente, los algoritmos genéticos paralelos son capaces de aprovechar las características de paralelismo implícitas en el propio problema de optimización, analizando concurrentemente diferentes secciones del espacio de búsqueda del problema.

3.3.2. Criterios de clasificación de Algoritmos Genéticos Paralelos

Un análisis de las ideas de aplicación de las técnicas de procesamiento paralelo y distribuido a los algoritmos genéticos, permite establecer diferentes criterios para clasificar los diferentes diseños propuestos de algoritmos genéticos paralelos. De acuerdo a la literatura especializada, pueden identificarse los criterios que se ofrecen a continuación:

- El modo de evaluación de la función de fitness, que puede ser concentrada o distribuirse en diferentes procesos.
- El número de poblaciones utilizadas por el algoritmo genético paralelo. Las alternativas consisten en utilizar una única población (modelo panmíctico) o utilizar poblaciones múltiples.
- El tamaño de las subpoblaciones y su organización, para aquellos modelos de poblaciones múltiples.
- El mecanismo de intercambio de individuos entre poblaciones múltiples.
- El modo de aplicación del mecanismo de cruzamiento, que puede ser centralizado o distribuido.
- El conjunto de individuos considerado para la aplicación del mecanismo de selección, puede ser un conjunto local o global.
- El mecanismo de sincronización entre los elementos de procesamiento.

El modo de organizar la población de individuos solución del problema durante el proceso evolutivo es considerado como el principal criterio de clasificación de algoritmos genéticos paralelos por parte de los investigadores en el área.

3.3.3. Modelos de Algoritmos Genéticos Paralelos

Análogamente a todo problema susceptible de resolución mediante las técnicas de procesamiento de alta performance, dos enfoques principales surgen intuitivamente para la paralelización del mecanismo de los algoritmos genéticos. Estos enfoques se corresponden con las técnicas usuales del procesamiento paralelo, las técnicas de descomposición funcional y descomposición de dominio, y han estado presentes en los trabajos de investigación desde los primeros intentos por paralelizar los algoritmos genéticos.

El modelo que surge de distribuir funcionalmente el algoritmo, asignando a diferentes procesadores distintas etapas del mecanismo evolutivo, se conoce como *modelo maestro-esclavo*. La funcionalidad candidata a paralelizar en un algoritmo genético es la evaluación de la función de fitness, la cual frecuentemente involucra un tiempo de ejecución mayor que el de los sencillos operadores evolutivos. La Figura 3.5 presenta gráficamente a un algoritmo genético paralelo modelo maestro-esclavo.

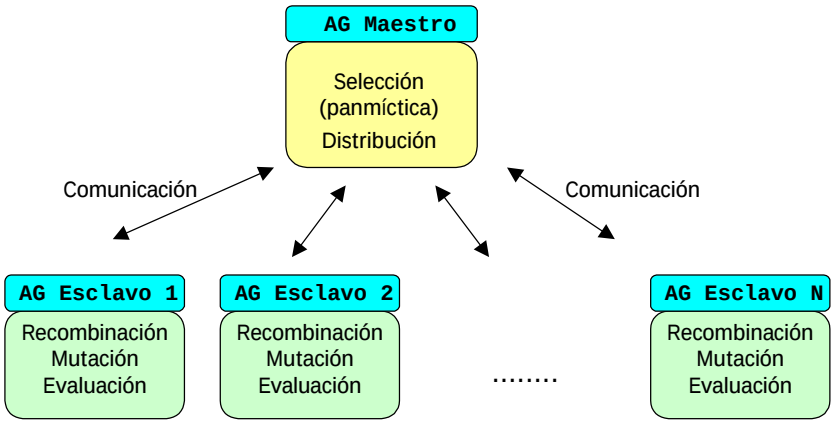


Figura 3.5: Algoritmo genético paralelo modelo maestro-esclavo.

Por contrapartida, un enfoque orientado a la distribución de datos da lugar a los denominados *modelos de poblaciones distribuidas*, en donde se organizan subpoblaciones que evolucionan de modo semi-independiente. Un operador de intercambio de individuos, denominado operador de *migración* permite a las subpoblaciones interactuar, introduciendo una nueva fuente de diversidad en el mecanismo evolutivo. La Figura 3.6 presenta gráficamente a un algoritmo genético paralelo modelo de poblaciones distribuidas.

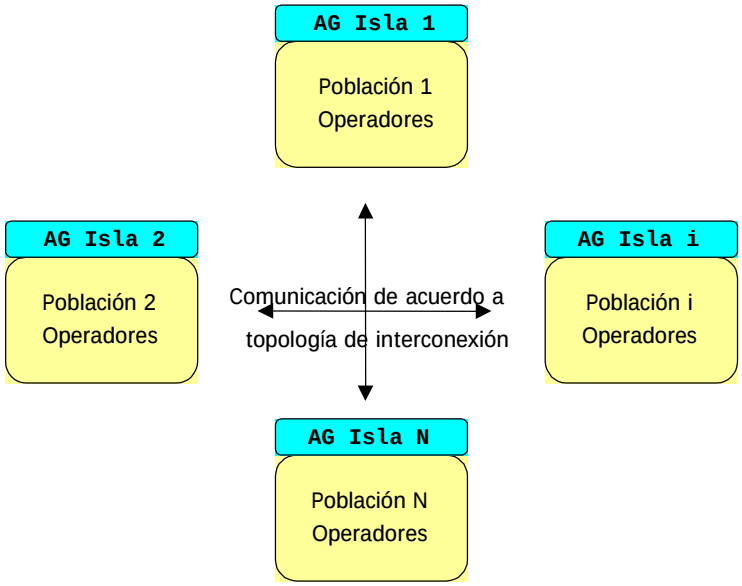


Figura 3.6: Algoritmo genético paralelo modelo de poblaciones distribuidas.

Un tercer modelo de algoritmo genético paralelo surge al considerar una estructura espacial subyacente que determina la vecindad entre los individuos de la población. El *modelo celular* o *masivamente paralelo* limita las interacciones a aquellos individuos definidos como adyacentes por la estructura, y el modelo de evolución reúne características particulares que lo diferencian de los dos modelos anteriormente presentados. La Figura 3.7 presenta gráficamente a un algoritmo genético paralelo modelo celular. El desarrollo de los multiprocesadores con arquitectura de memoria compartida popularizó este modelo de algoritmo genético paralelo en la década de 1990.

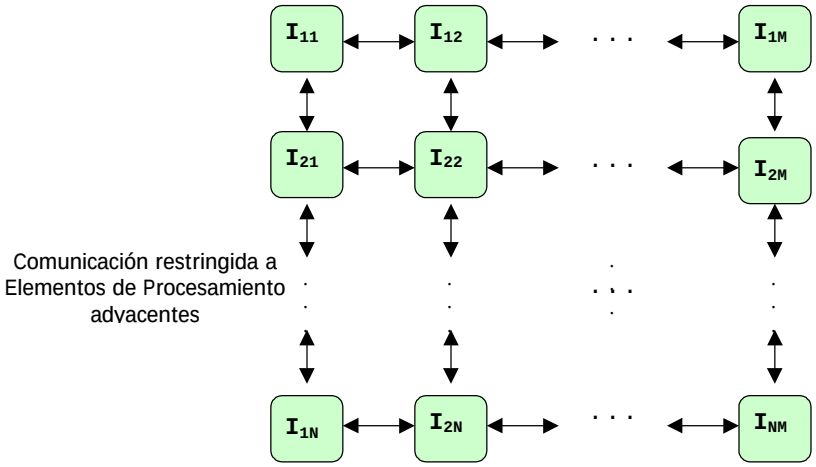


Figura 3.7: Algoritmo Genético Paralelo Modelo Celular o Masivamente Paralelo.

Estos tres modelos, reconocidos desde las primeras etapas de la investigación de los algoritmos genéticos paralelos, han evolucionado y se han diversificado. De modo particular, el enfoque de población distribuida ha dado lugar a varios modelos de paralelismo aplicado a los algoritmos genéticos. Otros enfoques no tradicionales de paralelización han sido propuestos, y se comentan en detalle en la reseña histórica de la Sección 3.4.

Usualmente al aplicar una técnica de procesamiento paralelo a un algoritmo serial, se obtiene un algoritmo paralelo, el cual provisto con la misma entrada que el algoritmo serial original, produce exactamente la misma salida que este. Excepto para el modelo maestro-esclavo, en el cual sólo se distribuye la evaluación de la función de fitness, los modelos de algoritmos genéticos paralelos violan la regla de producir la misma salida con la misma entrada que el algoritmo serial. Dado que el modelo de interacción panmíctica se reemplaza por un mecanismo de poblaciones semi-independientes, el comportamiento del algoritmo distribuido es diferente. En una población panmíctica cada individuo tiene la posibilidad de interactuar –competir o cruzarse– con cualquier otro individuo, mientras que en un modelo de poblaciones distribuidas las interacciones están limitadas a individuos de la misma población o vecindad.

Esta característica peculiar de los algoritmos genéticos paralelos de poblaciones distribuidas proporciona nuevos matices de interés sobre el problema paralelizado, dependiendo de los diferentes criterios utilizados al dividir el problema.

3.4. Etapas en el desarrollo de Algoritmos Genéticos Paralelos

En los últimos años, significativos avances en el ámbito del procesamiento paralelo y de la programación evolutiva han consolidado a los algoritmos genéticos paralelos como una técnica de optimización robusta y eficiente para la resolución de problemas NP-difíciles.

Desde sus orígenes a mediados de la década de 1970, una gran cantidad de propuestas de implementación y modelos teóricos han sido presentadas por los investigadores en el área de la computación evolutiva y el procesamiento paralelo. Analizando su contenido, es posible distinguir tres etapas en la investigación, relevantes para comprender los conceptos teóricos, de diseño e implementación de los algoritmos genéticos paralelos. Conjuntamente, con el paso de los años las propuestas de clasificación se han extendido para considerar la totalidad de los modelos de aplicación de paralelismo a los algoritmos genéticos. Un esquema cronológico de las etapas identificadas se ofrece en la Figura 3.8.

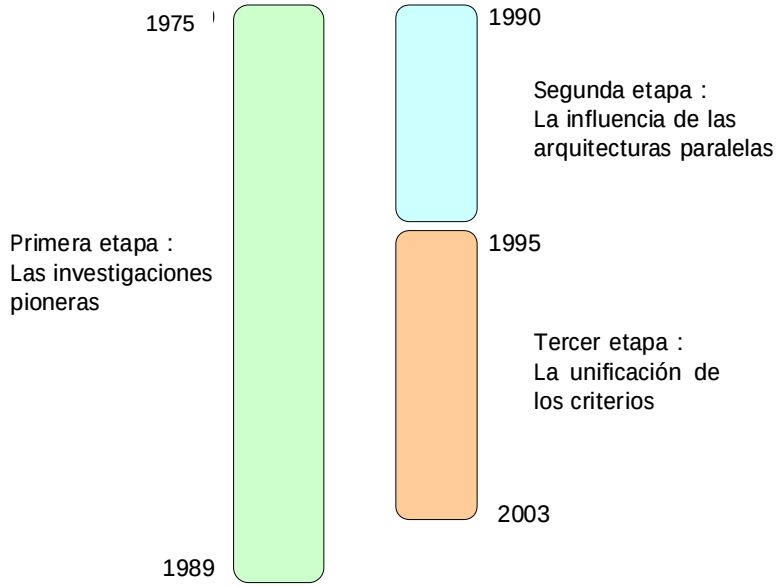


Figura 3.8: Etapas en el desarrollo de los algoritmos genéticos paralelos.

La primera etapa corresponde a los orígenes del modelo, en la cual fueron realizados análisis teóricos, propuestas de diseño y las primeras implementaciones de algoritmos genéticos paralelos. Cronológicamente, esta etapa se extiende desde los primeros estudios sobre algoritmos genéticos realizados a mediados de la década de 1970 hasta el fin de la década de 1980.

La segunda etapa se caracteriza por una prolífica gama de implementaciones de algoritmos genéticos paralelos aplicados a problemas particulares, la mayoría implementadas sobre plataformas específicas de hardware de alta performance. Esta etapa se extendió a lo largo de la primera mitad de la década de 1990 y estuvo claramente influenciada por la difusión de las técnicas de procesamiento paralelo, en particular sobre multicomputadores de memoria compartida y por la popularización de las técnicas de procesamiento distribuido sobre redes de computadoras.

La etapa que se extiende desde la segunda mitad de la década de 1990 hasta la actualidad se distingue por la preocupación por parte de los investigadores por la formalización de los conceptos teóricos y el modelado matemático del mecanismo de los algoritmos genéticos paralelos como herramienta de optimización. Complementariamente, se ha avanzado en la comprensión de los modos de aplicación de las técnicas de procesamiento paralelo, hasta proponerse taxonomías completas, que constituyen un intento de unificar la clasificación de los algoritmos genéticos paralelos.

En las secciones siguientes se resumen las características de cada una de estas etapas y se analizan los modelos propuestos, comentando especialmente los trabajos considerados representativos del avance en cada período.

3.4.1. 1970-1989: Las investigaciones pioneras

La naturaleza paralela del mecanismo de los sistemas adaptativos y de los algoritmos evolutivos fue reconocida en los trabajos pioneros en el área por parte de Holland (1963) y Bossert (1967) en la primera mitad de la década de 1960. De acuerdo a Goldberg (1989a), Holland inclusive discutió el funcionamiento de mecanismos reproductivos sobre un modelo primitivo de máquina paralela, la *computadora de circuitos iterativos*; por su parte, Bossert introdujo un modelo de poblaciones múltiples, con competición por supervivencia entre poblaciones, y eventos ocasionales para explorar la totalidad del espacio de soluciones, eliminando la peor población a intervalos aleatorios o modificando la función de fitness con el tiempo.

En sus trabajos, ni Holland ni Bossert llegaron a proponer implementaciones paralelas ni distribuidas de los algoritmos genéticos. Sus análisis se concentraron en los modelos seriales, cuyo enfoque permitía el estudio de sus propiedades matemáticas. Estos estudios pioneros facilitaron la comprensión de las características del mecanismo evolutivo y su aplicabilidad como herramienta de optimización.

Desde la citada idea de Holland, deben recorrerse casi 15 años para encontrar una nueva referencia a la conjunción de la programación evolutiva, algoritmos genéticos en particular, y las técnicas de procesamiento paralelo. En 1975, De Jong analizó teóricamente el modelo de paralelismo maestro-esclavo sincrónico, y propuso su variante con *salto generacional*, en los que se define un valor de salto que determina la fracción de la población que el algoritmo genético reemplaza en cada generación (De Jong, 1975)

En 1976, Bethke realizó el estudio teórico de un caso particular de algoritmo genético paralelo sobre una máquina paralela SIMD, analizando la eficiencia de uso de la capacidad de procesamiento (Bethke, 1976). El autor concluye que la eficiencia se acerca al máximo posible en los casos en los cuales las evaluaciones de la función de fitness son mucho más costosas que las operaciones de evolución, un caso típico para muchas aplicaciones. Sin embargo, no se incluyó en el análisis la influencia del overhead de comunicación. Si bien Bethke fue el primero en describir abstractamente la codificación de un algoritmo genético paralelo según el modelo maestro-esclavo, tampoco él llegó a proponer específicamente una implementación de un algoritmo paralelo, ni a realizar su simulación mediante algoritmos seriales, limitándose al análisis teórico mencionado.

La primera referencia a una taxonomía de los algoritmos genéticos paralelos corresponde a Grefenstette en su trabajo de 1981. En la clasificación elaborada por este autor se reconocen tres categorías, el modelo maestro-esclavo basado en la distribución de la función de fitness (del cual describe una versión sincrónica y una asincrónica), un modelo para máquinas paralelas de memoria compartida y un modelo distribuido de poblaciones múltiples (Grefenstette, 1981). A grandes rasgos, estas categorías primordiales han evolucionado a lo largo del tiempo, refinándose para aplicarse a diseños específicos y/o generalizándose al plantear taxonomías de amplio espectro.

En su trabajo de 1985, Grosso observó que la evolución de la función de fitness era más rápida en una población dividida en subpoblaciones o demes que en una única población distribuida y propuso técnicas de migración "retrasadas", en las cuales los demes evolucionan aisladamente hasta llegar a un punto cercano a la convergencia y luego comienzan las migraciones con alta frecuencia. Esta propuesta minimiza el costo debido a las comunicaciones, permitiendo obtener resultados de calidad similar a los obtenidos con las técnicas de migración estándar (Grosso, 1985).

De acuerdo a Davis et al. (1991), en 1987 Schwefel realizó una serie de experimentos para estudiar los efectos de la parametrización autoadaptativa en un modelo de Estrategia de Evolución (μ,λ). Schwefel utilizó una de las primeras computadoras paralelas de arquitectura SIMD, el modelo CRAY-1 y propuso el primer modelo paralelo para una Estrategia de Evolución.

En 1987 se presenta la primera implementación de un algoritmo genético paralelo, designado con el genérico nombre de Parallel Genetic Algorithm (PGA). Este algoritmo se basa en un modelo de subpoblaciones homogéneas cooperativas ("islas") con topología de migración dinámica, y fue propuesto por Pettey et al. (1987). El estudio preliminar de esta implementación fue empírico, siendo analizado teóricamente dos años más tarde por dos de sus creadores, Pettey y Leuze (1989). En este último trabajo, los autores amplían los análisis de De Jong (1975) sobre los algoritmos genéticos secuenciales con el fin de considerar los efectos de las estrategias de selección, cruzamiento y mutación sobre la eficiencia del algoritmo PGA. Análisis posteriores del funcionamiento del algoritmo y su éxito probabilístico fueron realizados por Mühlenbein (1991) quien lo aplica a la resolución del *Traveling Salesman Problem* (TSP).

Los algoritmos genéticos paralelos de grano fino fueron introducidos como una implementación de algoritmos genéticos sobre computadores masivamente paralelos por Robertson en 1987. La implementación se realizó sobre un equipo Connection Machine 1, paralelizando todas las fases del algoritmo (selección de progenitores, selección de población reemplazada, cruzamiento y mutación). El algoritmo seleccionaba y reemplazaba una pequeña fracción de la población en cada generación, y reportó muy buenos resultados, con un tiempo de ejecución independiente del tamaño de la población (Robertson , 1987).

Simultáneamente, los algoritmos genéticos paralelos de grano fino fueron estudiados por Manderick y Spiessen (1989), Mühlenbein et al (1988), Mühlenbein (1989) y Gorges-Schleuter (1989), proponiendo fijar los individuos en una grilla planar y restringiendo la selección y el cruzamiento a pequeñas vecindades en la grilla. El modelo de Mühlenbein y Gorges-Schleuter, de nombre ASPARAGOS, fue implementado en su primera versión sobre una red de transputers, utilizando para la población una estructura de escalera cíclica. Luego evolucionó incluyendo diferentes estructuras y mecanismos de pareo (Gorges-Schleuter, 1992) hasta constituirse en una herramienta efectiva de optimización (Gorges-Schleuter, 1997).

Variantes al modelo genérico de islas propuesto por Pettey et al. fueron introducidas por Cohoon et al. (1989) y Tanese (1989a). Cohoon et al. investigaron las similitudes entre la teoría de los equilibrios puntuados y el mecanismo evolutivos de los algoritmos genéticos paralelos multi-población. Por su parte, Tanese inició el estudio de los efectos de la migración en la eficiencia y calidad de resultados de los algoritmos genéticos paralelos, mostrando que tasas medianas de migración son necesarias para hallar soluciones de la misma calidad que un algoritmo genético serial. En términos de eficiencia del modelo paralelo, concluyó que pueden obtenerse speedups casi lineales cuando la migración es infrecuente y de tasa baja.

Complementariamente, Tanese (1989b) estudió el modo en que los parámetros de migración influyen en la pérdida de diversidad de alelos. Para prevenirla, propuso fijar diferentes parámetros y/o mecanismos de selección y cruzamiento para cada deme, causando que algunos converjan más lentamente que otros, en uno de las primeras referencias de un algoritmo genético paralelo heterogéneo encontradas en la literatura. En los experimentos comparativos realizados por Tanese está implícita la idea de que es posible realizar una comparación justa entre un modelo paralelo que utilice múltiples pequeñas poblaciones y uno que utilice una única población global.

En su texto de 1989, Goldberg sugiere un diseño "basado en objetos" para los algoritmos genéticos paralelos, y describe dos modelos: el ***modelo de comunidad***, un híbrido con multipoblaciones y migración en el cual cada población utiliza un modelo distribuido de evaluación de la función de fitness y el ***modelo de polinización***, un modelo distribuido, con estructura asignada a la población que determina la difusión de material genético (Goldberg, 1989a). Los modelos de Goldberg resultaron bastante complejos y no fueron implementados en la práctica. Complementariamente, Goldberg estudió el problema de determinar el tamaño adecuado para las subpoblaciones paralelas (Goldberg, 1989b).

En los trabajos de Grefenstette y Baker (1989) y Grefenstette (1991) se analizan propiedades invariantes para diferentes modelos de algoritmos genéticos. Se define la categoría de algoritmos genéticos admisibles, como aquellos que cumplen ciertos requerimientos mínimos en la manera de mapear la función objetivo en una medida de fitness y en el modo de utilizar la medida de fitness para crear descendientes de un par de individuos. Basándose en esta categorización, Grefenstette muestra que todo algoritmo genético admisible exhibe alguna forma de paralelismo implícito en el sentido de que el esfuerzo de búsqueda se concentra en varias combinaciones efectivas de genes (esquemas) simultáneamente.

En su trabajo de 1990, Gorges-Schleuter ofrece una clasificación primaria de los métodos de paralelización que utilizan poblaciones no panmícticas, reconociendo tres modelos:

- El modelo de islas, con demes separados, cada uno de los cuales constituye un algoritmo genético con población panmíctica.
- El modelo stepping stone, que cuenta con demes organizados de acuerdo a una estructura de vecindades no todas contiguas, lo cual limita el intercambio de material genético entre subpoblaciones.
- El modelo de vecindades, en el cual se define una relación de adyacencia sobre los individuos, los cuales pueden interactuar exclusivamente con sus vecinos.

La división entre los dos primeros modelos no es hoy considerada por los investigadores en el área, que se refieren a ambas como modelo de islas o migración (Gorges-Schleuter, 1990).

Un resumen de los principales trabajos desarrollados en la etapa de investigaciones pioneras se ofrece en la Figura 3.9.

Autor	*Año*	*Trabajo*
Holland	1963	Estudio de propiedades teóricas de los AGP.
Bossert	1967	Estudio de propiedades teóricas de los AGP.
De Jong	1975	AGP Modelo maestro-esclavo.
Bethke	1976	AGP sobre una máquina paralela SIMD.
Grefenstette	1981	Primera clasificación de AGP.
Grosso	1985	Estudio de modelos de migración.
Pettey et al.	1987	Implementación Parallel Genetic Algorithm (PGA)
Robertson	1987	AGP de grano fino.
Cohoon et al.	1989	Equilibrios puntados y modelos multipoblación.
Tanese	1989	Estudios de efectos de parámetros de migración.
Manderick, Spiessen	1989	AGP de grano fino.
Mühlenbein	1989	AGP de grano fino.
Gorges-Schleuter	1989	AGP de grano fino.
Goldberg	1989	Modelos de AGP basados en objetos.
Grefenstette	1989-91	Paralelismo implícito en AG.
Gorges-Schleuter	1990	Clasificación de AGP con poblaciones no panmícticas

Figura 3.9: Principales trabajos de la etapa de investigaciones pioneras.

3.4.2. 1990-1995: La influencia del desarrollo de las arquitecturas paralelas

La proliferación de diseños de arquitecturas paralelas y de alta performance sobre finales de la década de 1980 y comienzo de la década de 1990 influyó positivamente en la investigación y desarrollo de versiones paralelas de los algoritmos genéticos. El trabajo de investigación se movió desde el terreno teórico al ámbito de las implementaciones y sus aplicaciones a la resolución de complejos problemas de optimización. Simultáneamente, el análisis comparativo de las estrategias de paralelización aplicadas se manifestó como un importante tema de estudio.

En esta etapa, que es posible considerar que se extiende hasta mediados de la década de 1990, es frecuente encontrar clasificaciones específicas y subclasificaciones para cada uno de los modelos generales identificados. En general, los investigadores centraron su trabajo en alguna arquitectura multiprocesador específica, la cual condicionaba el modelo de paralelismo a aplicar, de modo de obtener las mayores ventajas del hardware disponible.

La tendencia a las implementaciones específicas comienza a revertirse hacia el final del lustro, surgiendo trabajos con el objetivo de condensar los resultados de las investigaciones y ofrecer modelos independientes de las implementaciones y arquitecturas.

Continuando el mencionado estudio de Grosso (1985), Braun exploró las técnicas de migración "retrasadas" o activadas por eventos y la "degeneración" (migración que se produce luego de la convergencia absoluta) en su trabajo de 1990 (Braun, 1990)

En su trabajo de 1991, Starkweather et al. estudian si es posible, y en caso de serlo bajo que condiciones, que un algoritmo genético paralelo proporcione una solución de mejor calidad que un modelo serial. Los autores especulan que los algoritmos genéticos paralelos se desempeñarán adecuadamente para funciones de fitness separables, en las cuales combinando soluciones parciales es posible obtener una mejor solución, que en aquellos problemas con funciones de fitness no separables (Starkweather et al, 1991). Esta línea de trabajo será extendida más adelante por Whitley et al. (1997a).

Como consecuencia de su simplicidad de implementación y análisis, el modelo maestro-esclavo se popularizó como el primero en implementarse sobre arquitecturas paralelas.

Fogarty y Huang (1991) trabajaron implementando el modelo maestro-esclavo sobre una red de transputers, concluyendo que el importante crecimiento del overhead de comunicación constituía un impedimento para futuras mejoras de sus resultados. Varios investigadores chocaron contra este problema de la escalabilidad del modelo maestro-esclavo. Al agregar procesadores, la eficiencia de su algoritmo genético paralelo decrecía como consecuencia del overhead introducido para posibilitar la comunicación.

Abramson y Abela (1992) implementaron un algoritmo genético paralelo panmíctico sobre una arquitectura de memoria compartida (Encore Multimax de 16 procesadores) para resolver el problema de asignación de horarios escolares, reportando speedups limitados. Más adelante, trabajando sobre esta implementación, Abramson, Mills y Perkins (1993) agregaron una máquina de memoria distribuida (Fujitsu AP1000 con 128 procesadores) y lograron buenos speedups utilizando ambos equipos, al trabajar con hasta 16 procesadores, degradándose la performance al incrementar el número de procesadores por encima de ese valor.

En su trabajo de 1992, Baluja resume las ideas detrás del desarrollo contemporáneo de la teoría de equilibrios puntuados, original de Cohoon (1987). Esta teoría se basa en los principios de especiación alopátrica –que involucra la rápida evolución de nuevas especies en un entorno separado– y de estabilidad –ausencia de cambio. Considera que la reintroducción de viejos especímenes constituye un poderoso mecanismo para recomponer la diversidad en un ambiente que alcanzó un equilibrio, pero donde el cambio es beneficioso. Este concepto es la base de las técnicas de migración. Hasta la propuesta de Baluja, los investigadores asumían que aunque las poblaciones separadas introducen un beneficio, los efectos de la introducción súbita de nuevas características podrían ser contraprocentes para el mecanismo evolutivo.

En su trabajo, Baluja modifica la idea de la arquitectura básica de los algoritmos genéticos masivamente paralelos existente hasta el momento, planteando que mediante una reducción de la severidad de los bordes entre poblaciones puede ser posible sobrellevar los problemas asociados con la introducción súbita de nuevo material genético. Un modo de conceptualizar esta idea es introducir solapamiento en las poblaciones. Esta estructura permite la transferencia gradual de información genética sin la introducción súbita de cromosomas. La porción de intercambio del algoritmo genético paralelo, formada por las secciones solapadas entre subpoblaciones, es inherente a la propia estructura de las subpoblaciones (Baluja, 1992).

Para el diseño de su modelo, denominado ***massively distributed parallel genetic algorithm*** (mpdGA), Baluja analiza los aspectos de estructura y localidad de las subpoblaciones y los tamaños de las secciones solapadas entre poblaciones, mencionando que debe llegarse a un compromiso entre la lentitud de propagación de buenos cromosomas para áreas de solapamiento pequeñas y la desviación de la idea de los equilibrios puntuados al crecer las áreas y acercarse a un modelo panmíctico.

La propuesta de implementación de Baluja se basa en la arquitectura paralela MassPar MP-1, modelo SIMD, compuesta de un array de elementos de procesamiento. Su trabajo reporta la obtención de un modo más eficiente de resultados de similar calidad para el conjunto de problemas de prueba –problemas de De Jong, funciones deceptivas y la maximización de la función gap– que los obtenidos mediante un algoritmo genético paralelo de poblaciones distribuidas, de implementación basada en Genitor, un algoritmo propuesto por Whitley (1989).

Como extensión a su trabajo sobre los algoritmos genéticos masivamente paralelos, en un trabajo posterior de 1993, Baluja estudia diferentes topologías para la estructura de población y su performance sobre un conjunto de problemas estándar de prueba y el problema TSP, sin llegar a conclusiones definitivas sobre la existencia de una configuración óptima (Baluja, 1993).

El problema de la convergencia prematura y la propuesta de una implementación para evitarlo mediante el modelo de migración sobre una arquitectura nCube es estudiado por Balakrishnan en su trabajo de 1993. Los resultados de este trabajo indican la efectividad del mecanismo de paralelismo implementado para la impedir la convergencia anticipada aumentando la diversidad de la población (Balakrishnan, 1993).

El modelo denominado forkingGA es introducido por Tsutsui y Fujimoto, en su trabajo de 1993, en base a la idea de utilizar multipoblaciones independientes para la solución de problemas multimodales. El mecanismo de migración utilizado considera una población padre con modo bloqueante y poblaciones hijas con un modo de encogimiento como resultado de una creación de poblaciones. Cada población toma un rol diferente en la tarea de optimización, siendo responsable de diferentes sectores no solapados del espacio de búsqueda (Tsutsui y Fujimoto, 1993). El modelo es extendido por sus creadores en trabajos posteriores (Tsutsui y Fujimoto, 1994), (Tsutsui et al., 1997).

En el trabajo de Gordon y Whitley (1993) se propone una clasificación de algoritmos genéticos paralelos que reconoce tres categorías, basadas en la división originalmente presentada por Goldberg (1989a):

- modelo de población global, una versión simplificada del modelo maestro-esclavo que utiliza selección por torneo para simplificar el paralelismo, y sus versiones elitistas.
- modelo de islas y migración.
- modelo masivamente paralelo o celular, que asigna un número bajo de individuos por procesador o elemento de procesamiento (EP), llegando al caso extremo de asignar un individuo por EP, requiriéndose tantos EP como individuos en la población. En este modelo, cada individuo se procesa en paralelo en cada generación y el cruzamiento está limitado a un deme -o vecindad- del individuo. Usualmente la topología de conexión y la estructura de los demes es fija, y se corresponde con la topología de conexión de los EP en el supercomputador. Una descripción del modelo celular se presenta en el trabajo de Whitley (1993), en el cual se justifica esta denominación por tratarse de un tipo particular de autómata celular.

Los autores realizan un estudio comparativo de diferentes modelos de algoritmos genéticos paralelos , incluyendo a su propia implementación, denominada Genitor. Genitor nació en 1989 como un algoritmo genético paralelo de población global y selección por rango (Whitley y Kauth, 1988), (Whitley et al. 1989), y evolucionó a un modelo de islas con migración en su versión Genitor II (Whitley y Starkweather, 1990).

Un modelo de paralelismo trivial, asignando varios procesadores a una misma instancia del problema que se resuelve concurrentemente mediante algoritmos genéticos seriales, es propuesto por Shonkwiler, quien lo denomina IIP (independent, identical and parallel) en su trabajo de 1993. Su método permite obtener interesantes resultados estadísticos, trabajar con diferentes condiciones iniciales y combinaciones de los parámetros, etc. (Shonkwiler, 1993).

Uno de los primeros modelos de grano fino implementados sobre una red de workstations interconectadas por una LAN se describe en el trabajo de Maruyama, Hirose y Konagaya (1993). En este trabajo se analizan los efectos del sincronismo entre poblaciones distribuidas como influencia sobre la calidad de la solución obtenida y la mejora de performance comparada con el algoritmo genético serial.

Las estrategias de distribución de datos para el modelo de algoritmos genéticos desordenados (*messy genetic algorithms*) son estudiadas en el trabajo de Merkle y Lamont (1993). El modelo de algoritmos genéticos desordenados es original de Goldberg (1990), ideado para tratar con las limitaciones de optimización de ciertas funciones deceptivas. El algoritmo genético desordenado consta de una *fase primordial* en la cual se crea la población utilizando una estrategia de enumeración parcial, distribuyendo los bloques de construcción de soluciones y una *fase de yuxtaposición* en la cual se mezclan las soluciones encontradas en la fase primordial. La fase primordial usualmente domina en cuanto al tiempo de ejecución, por lo cual técnicas de paralelismo pueden aplicarse para mejorar la performance.

En el trabajo citado, Merkle y Lamont proponen diferentes estrategias de distribución para explotar el paralelismo de datos en el algoritmo de selección por torneo de la fase primordial. Conjuntamente con las estrategias propuestas (indexado de soluciones, indexado modificado, ordenamiento e intercalado de bloques de construcción y ordenamiento y asignación por bloques), los autores ofrecen una interesante descripción del algoritmo en un enfoque independiente de la arquitectura, y un análisis de los efectos del tamaño de subpoblaciones y su compatibilidad. Como conclusión de su trabajo, prueban el efecto significativo de las estrategias de distribución de datos sobre el tiempo de ejecución de la fase primordial, aunque dejan constancia de que no se mejora la calidad de la solución respecto al modelo secuencial.

En su trabajo de 1993, Munetomo, Takai y Sato estudian los diferentes esquemas de migración que permiten reducir comunicaciones innecesarias en los modelos de poblaciones distribuidas. El esquema definido, denominado algoritmo de intercambio sigma toma en cuenta la desviación estándar en la distribución del fitness de cada deme para determinar la frecuencia de intercambio de material genético (Munetomo, Takai y Sato, 1993).

El trabajo de Masakazu et al. (1993) propone un modelo paralelo con estructuras jerárquicas de subpoblaciones que incorpora un mecanismo de optimización de los parámetros de cada algoritmo genético utilizando un meta-algoritmo genético de nivel superior. Los parámetros de los operadores genéticos se codifican y se optimizan mediante el proceso evolutivo del meta-algoritmo genético. El modelo se implementó sobre una red de workstations resultando un efectivo algoritmo de búsqueda distribuido.

En el trabajo de Bianchini y Brown (1993) se analizan los aspectos de implementación de algoritmos genéticos paralelos sobre arquitecturas paralelas de memoria distribuida. Sobre estas plataformas, las consideraciones de performance tradicionalmente condicionaban el diseño de los algoritmos genéticos paralelos, haciendo que las implementaciones centralizadas fueron dejadas de lado, superadas por una variada gama de implementaciones distribuidas, las cuales involucran menor número de cuellos de botella en las comunicaciones, y granularidad mayor en las subpoblaciones.

Sin embargo, los autores mencionan ciertos casos en los cuales la calidad de la búsqueda mejora al trabajar con una única población global respecto al resultado obtenido trabajando con varias subpoblaciones, en particular para hallar buenas soluciones iniciales aproximadas en pocas generaciones. Trabajando sobre los modelos de Grefenstette (1981), los autores proponen una extensión a los prototipos sobre memoria distribuida, introduciendo aspectos de centralización en los algoritmos. El objetivo detrás de este enfoque, llamado "semi-distribuido" por los autores, consiste en resolver adecuadamente el delicado equilibrio entre calidad de la búsqueda y performance del algoritmo.

Se propone la organización de los procesadores en clusters, en un nivel por encima de la asignación de poblaciones a procesadores, logrando un cierto grado de centralización. De este modo se mantiene el enfoque de poblaciones distribuidas sin generar un aumento excesivo en las comunicaciones. Esta implementación comparte algunas de las ventajas de los modelos distribuidos y otras de los centralizados.

Basados en las ideas expuestas, Bianchini y Brown proponen una taxonomía de algoritmos genéticos paralelos de acuerdo con el modo de distribución de la población entre los múltiples procesadores. Las estrategias de implementación expuestas son:

- Implementación centralizada: Considerado como un caso extremo, consta de una única población y la implementación sigue el modelo maestro-esclavo, sobre una topología de interconexión en estrella, con el proceso maestro en el centro. La estrategia maestro-esclavo no es práctica en la mayoría de las implementaciones sobre memoria distribuida, como consecuencia del cuello de botella en las comunicaciones hacia el maestro, la secuencialidad necesaria del proceso de replicación y la granularidad demasiado fina de los procesos que genera comunicaciones excesivas. Para aminorar el impacto de estos problemas, los autores asignan en su implementación un gran número de individuos a cada proceso esclavo, seleccionando adecuadamente los grupos de modo de que a diferentes esclavos sean asignados aproximadamente el mismo número de individuos. Complementariamente, los procesos esclavos incorporan la lógica necesaria para realizar la operativa de evolución, trabajando aislados por un cierto número de generaciones antes de enviar los resultados al maestro y sincronizar con el resto de la población. El maestro continúa siendo responsable de la ejecución del mecanismo de evolución sobre la población en su conjunto, pero ésta etapa, que involucra comunicaciones, ocurre con menos frecuencia. Para reducir el tiempo ocioso del proceso maestro mientras aguarda el resultado del trabajo de los esclavos, se le asigna una subpoblación a dicho proceso maestro, cuyo tamaño se selecciona adecuadamente de modo que el proceso maestro finalice el procesamiento de su población antes de que los esclavos comiencen los envíos de sus resultados.
- Implementación semi-distribuida: Constituye un modelo intermedio entre una población global y una totalmente distribuida, que intenta evitar el cuello de botella en las comunicaciones hacia un proceso maestro. La idea consiste en organizar clusters de procesadores que trabajen como lo hace el modelo centralizado. Los autores proponen la conexión de los procesos maestros de cada cluster en una topología toroidal, basando esta elección en cuestiones de simplicidad del algoritmo, ya que el diseño cíclico permite que cada proceso tenga un vecino en las cuatro direcciones espaciales. Cada cierto tiempo se realizan comunicaciones sincrónicas entre los procesos maestros, intercambiando los mejores individuos. Los esclavos se comportan del mismo modo que en la implementación centralizada. Esta organización reduce el efecto del problema de contención causada por el acceso a un único maestro, aunque incluye cierto overhead como consecuencia de la replicación semi-distribuida y las nuevas comunicaciones entre procesos maestros.
- Implementación distribuida: Este modelo elimina las poblaciones compartidas. Cada procesador mantiene una porción de la población y aplica un algoritmo genético secuencial sobre ella, con migración ocasional de los mejores individuos. Nuevamente se propone una topología toroidal de conexión para simplificar el algoritmo y reducir las comunicaciones. Como se reconoce en el trabajo, en las implementaciones de este estilo se asigna un esfuerzo computacional en el procesamiento de individuos poco aptos, pero este problema puede reducirse adoptando estrategias de migración adecuadas, que introduzcan la diversificación necesaria en las subpoblaciones.
- Implementación totalmente distribuida: Un caso extremo, que corresponde al modelo anterior, sin migración entre subpoblaciones. Es el modelo con menor cantidad de comunicaciones.

Complementando la clasificación anterior, los autores ofrecen una breve introducción a las implementaciones de modelos paralelos sobre arquitecturas de memoria compartida. Generalmente, los algoritmos genéticos paralelos centralizados se consideran más adecuados para arquitecturas de memoria compartida, ya que el mecanismo para compartir la población es proporcionado por el hardware y el sistema operativo del multiprocesador correspondiente. Disponiendo de este hardware especializado, simplemente se debe recargar el caché de los procesadores en cada generación, un mecanismo que no involucra pasaje de mensajes de ningún tipo y que se resuelve a nivel del bus de datos del equipo. Pero las limitaciones en el ancho de banda del mencionado canal y las latencias en las memorias son muy grandes comparadas con las velocidades de los procesadores, lo cual ocasiona que la performance se degrade cuando se utilizan muchos procesadores y una población numerosa.

Los autores indican que los niveles de performance alcanzados por sus versiones centralizadas y semi-distribuidas sobre arquitecturas de memoria distribuida mejoran la preformance obtenida por los enfoques centralizados sobre memoria compartida. En una posterior extensión al trabajo anterior, Bianchini et al. (1995) analizan cuantitativamente las ventajas del modelo semi-distribuido sobre las restantes implementaciones. En este nuevo artículo, los autores extienden el conjunto de problemas de testeo, originalmente compuesto exclusivamente por el problema de Programación Lineal Entera 0-1, contemplando funciones bien conocidas en los benchmarks de optimización, los cinco problemas de minimización de DeJong y el TSP. La conclusión de su trabajo indica que realizar selecciones centralizadas cada cierto tiempo consiste en una mejora sobre los enfoques tradicionales de distribución de población. Califican la estrategia de centralizaciones como un mecanismo extremadamente robusto que permite lograr mejoras en la calidad de los resultados obtenidos, manteniendo los tiempos de ejecución y las comunicaciones bajo control.

En su reseña de 1994, Adamidis plantea una categorización de algoritmos genéticos paralelos basada en dos enfoques distintivos respecto a la operativa del mecanismo evolutivo, el modelo estándar y el modelo de descomposición (Adamidis, 1994):

- El modelo estándar de Adamidis consiste en la implementación del mecanismo de los algoritmos genéticos secuenciales sobre máquinas paralelas, correspondiendo al modelo de paralelización global o modelo maestro-esclavo. El mecanismo algorítmico del modelo serial no se modifica al distribuir la evaluación de la función de fitness.
- El modelo de descomposición corresponde a una operativa diferente a la de los algoritmos genéticos seriales, donde la población existe en una forma distribuida, ya sea en forma de una única población efectivamente distribuida entre procesos o poblaciones múltiples independientes. El autor refina este modelo, considerando los algoritmos genéticos paralelos de grano fino (población única) y los de grano grueso (modelo de islas y stepping stone).
- Complementariamente, menciona la posibilidad de crear algoritmos híbridos que combinen los enfoques, los cuales cuentan con un grado de complejidad mayor que los modelos paralelos mencionados. En esta línea de investigación, el autor propuso posteriormente un modelo cooperativo (CoPDEB – *Co-operating Populations with Different Evolution Behavior*) en su trabajo con Petridis (Adamidis y Petridis, 1996) y en su tesis de doctorado (Adamidis 1997). El comportamiento diferenciado de las subpoblaciones en el este modelo se consigue utilizando diferentes mecanismos de selección, operadores y métodos de comunicación.

Los trabajos de Levine (1993,1994) presentan una implementación paralela para los algoritmos genéticos de estado estacionario –aquellos que reemplazan solamente una pequeña parte de la población, usualmente un conjunto muy pequeño de individuos en cada generación. El diseño de Levine consiste en un modelo de islas con poblaciones múltiples independientes que evolucionan de acuerdo a un algoritmo genético de estado estacionario y pequeñas migraciones ocasionales de los individuos más aptos. La propuesta del modelo paralelo para el algoritmo genético de estado estacionario fue implementada en un IBM SP-1 con 128 nodos y aplicada al problema de particionamiento de conjuntos para modelar el control de tráfico aéreo, mostrando significativos resultados en términos de mejora de performance y calidad de la solución hallada. Otro resultado destacable lo constituye la capacidad de escalabilidad de la implementación para resolver problemas más complejos.

En la clasificación de implementaciones de algoritmos genéticos paralelos ofrecida en el trabajo de Lin, Punch y Goodman (1994) se describen las tres categorías clásicas ya comentadas. Su trabajo se especializa en los algoritmos genéticos paralelos de grano grueso, analizando diferentes estrategias de migración y esquemas de conectividad, como principal mecanismo para resolver el problema de la convergencia prematura de los algoritmos genéticos seriales o de población única. Los autores resaltan la capacidad de los algoritmos genéticos paralelos de mantener poblaciones múltiples, capaces de evolucionar independientemente, permitiendo una mejor y más exhaustiva búsqueda en el espacio de soluciones. Las categorías de la clasificación de Lin, Punch y Goodman comprenden:

- Modelo micrograno: Utiliza una población única, con distribución tipo maestro-esclavo de evaluación de la función de fitness. No resuelve el problema de convergencia prematura, pero logra mejoras significativas de performance respecto al algoritmo genético serial para problemas complejos.
- Modelo de grano fino: Asigna un individuo a cada elemento de procesamiento. Cada individuo forma parte de varias subpoblaciones, solapadas entre sí. La adyacencia entre individuos está determinada por la topología de interconexión entre los elementos de procesamiento. La alta conectividad entre vecinos incrementa la difusión de individuos con mejor fitness, pero hace a la población susceptible al dominio de algún esquema de codificación y podría llevar a convergencia prematura. Limitar las interacciones, por su parte, reduce la performance de la implementación. Los autores plantean que en las implementaciones de este modelo debe lograrse un compromiso entre estos factores para obtener una solución de calidad, con buena performance.
- Modelo de grano grueso: Caracterizado por poseer una gran cantidad de individuos asignados a poblaciones semi-independientes.

Conjuntamente, los autores estudian los diferentes criterios que permiten clasificar a los algoritmos genéticos paralelos de grano grueso. Distinguen entre los tres criterios que se presentan a continuación:

- El modelo de migración, que determina cómo y cuando se realiza el intercambio de individuos, diferenciando entre modelo aislado, donde las poblaciones evolucionan independientemente sin existir comunicación, el modelo de islas sincrónico, y el modelo de islas asincrónico, en los cuales la migración debe ocurrir en un punto común para las subpoblaciones (sincrónico) o puede ocurrir en eventos no relacionados con la evolución de las subpoblaciones (asincrónico)
- El esquema de conectividad entre poblaciones. El esquema puede ser estático, basado en las topologías predefinidas entre los elementos de procesamiento de los multicomputadores de alta performance, o dinámico, un enfoque en el cual la topología de conexión entre subpoblaciones no está prefijada, sino que se permite que varíe durante la evolución.

- El criterio de homogeneidad de las poblaciones. Los modelos homogéneos utilizan en sus subpoblaciones componentes los mismos mecanismos de codificación, selección, cruzamiento y reemplazo, e inclusive los mismos criterios para la selección de sus parámetros. Por su parte, los modelos no homogéneos permiten a las subpoblaciones utilizar diferentes mecanismos evolutivos, con criterios propios para la selección de parámetros, e inclusive utilizando codificaciones diferentes para las soluciones del problema. La variación de representaciones y esquemas evolutivos de un modelo heterogéneo puede resultar beneficiosa para la resolución de problemas de optimización multiobjetivo o con espacios de búsqueda extensos.

Finalmente, los autores proponen un modelo de algoritmo genético paralelo asincrónico, estático (característica que no es limitante del modelo propuesto, que podría ser dinámico) y heterogéneo, denominado *Injection Island*. Este algoritmo tiene como característica distintiva su heterogeneidad, utilizando diferentes resoluciones (largos de strings de representación) para las codificaciones de los individuos en cada población. El mecanismo de migración es solo de "una vía", el intercambio se da exclusivamente desde poblaciones de baja resolución a poblaciones de alta resolución. Este modelo de migración requiere de un mecanismo de ***traducción*** entre representaciones, la cual se hace sin pérdida de información al considerar la jerarquía de intercambio indicada. Dicha jerarquía establece un mecanismo de ***refinamiento*** de soluciones, basado en las reglas de migración, que comprenden cuatro modelos (simple, completa, estricta, pobre) diferenciados por los diferentes destinos de los individuos migrados.

El modelo *Injection Island* realiza la búsqueda en múltiples codificaciones, cada una concentrada en diferentes áreas del espacio de soluciones. El espacio de búsqueda para nodos de baja resolución es proporcionalmente menor, lo cual permite hallar soluciones "adecuadas" rápidamente, que son inyectadas a los nodos de alta resolución para su posterior refinamiento. La jerarquía define nodos padre e hijo, que comparten porciones del mismo espacio de búsqueda. La búsqueda de rápidos candidatos en baja resolución por parte del padre puede potencialmente orientar al hijo en su búsqueda. Este enfoque permite una aplicación sistemática de una técnica Divide & Conquer que no puede ser garantizada por los algoritmos genéticos paralelos homogéneos, los cuales en general producen soluciones pertenecientes simultáneamente a varios subespacios de búsqueda.

Simultáneamente, los nodos con el tamaño de bloque más pequeño pueden hallar soluciones con la mayor resolución. El modelo *Injection Island* procesa los diferentes niveles de resolución en paralelo y reduce el riesgo de búsqueda en intervalos incorrectos evitando el fenómeno de convergencia prematura.

En su completo trabajo de 1996, Schwehm discute los tipos de algoritmos genéticos paralelos independizándose de aquellas implementaciones diseñadas sobre hardware especial. El resultado es una categorización precisa, que describe claramente las características relevantes de cada modelo (Schwehm, 1996).

Analizando el modelo de poblaciones paralelas, Schwehm propone una clasificación basada en la organización estructural de la población. Las categorías propuestas, de menor a mayor grado de estructuración, corresponden a ***modelo global*** (población no estructurada y mecanismo de selección global), ***modelo regional*** (subpoblaciones y modelo de selección local) y ***modelo local*** (población provista con una estructura de vecindad, que define el mecanismo de selección de individuos para el cruzamiento). Luego de exponer su clasificación, explica la correspondencia existente con los modelos de otras taxonomías, algunas de ellas estudiadas en esta reseña (Grefenstette, 1981), (Manderick y Spiessens, 1989), (Gorges-Schleuter, 1989).

Las características de las categorías en la clasificación de Schwehm se comentan a continuación:

- El *modelo global* implementa el algoritmo genético clásico, con su única población panmíctica, sobre una plataforma de hardware paralelo-distribuida. Se identifica al mecanismo de selección proporcional como el cuello de botella de la implementación, mencionando como referencia el citado análisis de Bethke (1976). Dentro del modelo global, es posible identificar variantes de acuerdo al mecanismo para lidiar con dicho problema. El autor menciona a los modelos globales con selección secuencial, que omiten la paralelización del mecanismo de selección. Este modelo es implementable como un típico modelo maestro-esclavo con la población almacenada en memoria compartida o en el proceso maestro, y son útiles cuando la evaluación de la función de fitness es costosa. Asimismo se mencionan los modelos globales con selección paralela, que distribuyen el cálculo del fitness absoluto para el mecanismo de selección proporcional, o el mecanismo de ordenación para mecanismo de selección por rangos, ambos con costo logarítmico en el número de individuos; o el mecanismo de selección por torneo, más simple de paralelizar. En estos enfoques subsiste el problema de obtener los individuos seleccionados para el cruzamiento, tratando de evitar los habituales conflictos de memoria o cuellos de botella en las comunicaciones.
- El *modelo regional* constituye una extensión al modelo clásico de los algoritmos genéticos. La población se divide en subpoblaciones (regiones o demes) cada una de las cuales ejecuta independientemente un algoritmo genético. Los mecanismos globales de selección, cruzamiento y reemplazo se reemplazan por mecanismos locales y se agrega un operador de intercambio de material genético entre las poblaciones (migración). Dentro de este modelo, diversas variantes pueden establecerse de acuerdo a la estructura y número de población proporcionada a las regiones, así como las topologías de migración y los parámetros que la controlan - estrategia de migración, tasa de migración, frecuencia de migración-. Mediante la modificación de estos parámetros, una determinada instancia del modelo regional puede acercarse más al modelo global -utilizando una topología de migración completa, alta tasa y frecuencia de migración- o al modelo local - utilizando una topología de migración rígida.
- El *modelo local* consiste en una modificación de la estructura de los algoritmos genéticos, en la cual se provee a la población de una estructura espacial. Los mecanismos de selección, reproducción y reemplazo se llevan a cabo localmente, dentro de una vecindad definida por proximidad espacial entre los individuos. La estructura de la vecindad determina la ubicación espacial de los individuos y los caminos de propagación de material genético entre la población. Los principales parámetros espaciales de este modelo son la topología de la vecindad y su tamaño (radio). Las estrategias de selección local y reemplazo local determinan la presión de selección, usualmente menor en este modelo que en los modelos regional y global, como consecuencia de las limitaciones impuestas por la estructura espacial. Las estrategias más comunes de cruzamiento local utilizan el valor de fitness proporcional local o un rango proporcional, e inclusive la selección por torneo local. Pueden seleccionarse dos individuos de la vecindad para reproducción o seleccionarse un único individuo que forzosamente se apareará con el individuo central de la vecindad. La estrategia de reemplazo estándar para este modelo es el reemplazo generacional sustituyendo el individuo que ocupa el lugar central de la vecindad, ya que el reemplazo elitista no puede utilizarse como técnica local, al necesitar conocimiento de datos de la población global.

Es posible considerar el trabajo de Schwehm como la culminación de la segunda etapa en la evolución del diseño y las clasificaciones de algoritmos genéticos paralelos, por ofrecer una taxonomía completa, independiente de las implementaciones particulares y del hardware subyacente, y la correspondencia existente con los modelos de otras clasificaciones.

A modo de resumen, los principales trabajos de la segunda etapa en la evolución del diseño y las clasificaciones de algoritmos genéticos paralelos se ofrecen en la Figura 3.10.

Autor	*Año*	*Línea de trabajo*
Braun	1990	Técnicas de migración "retrasadas"
Starkweather et al.	1991	Condiciones para que un AGP supere calidad de un AG.
Fogarty y Huang	1991	Modelo maestro-esclavo sobre una red de transputers.
Abramson y Abela	1992	AGP panmíctico sobre memoria compartida.
Baluja	1992	AGP masivamente paralelos.
Shonkwiler	1993	Modelo de paralelismo trivial, sin comunicación.
Gordon y Whitley	1993	Clasificación de AGP sobre modelos de Goldberg.
Maruyama et al.	1993	AGP de grano fino implementado sobre una red de workstations interconectadas por una LAN.
Merkle y Lamont.	1993	Estrategias de distribución de datos para el modelo de AG desordenados (Messy GA).
Tsutsui y Fujimoto	1993	ForkingGA – multipoblaciones independientes para la solución de problemas multimodales.
Masakazu et al.	1993	modelo de AGP con estructuras jerárquicas.
Munetomo, Takai y Sato	1993	Mecanismos de migración para reducir comunicaciones.
Bianchini y Brown	1993	AGP sobre arquitecturas de memoria distribuida.
Adamidis	1994	Categorización de AGP basada en la operativa del mecanismo evolutivo.
Levine	1994	Modelos de AGP de estado estacionario.
Lin, Punch y Goodman	1994	Clasificación de AGP, especializada en grano grueso.
Schwehm	1996	Clasificación de AGP independiente del hardware.

Figura 3.10: Principales trabajos de la etapa influida por la difusión de arquitecturas paralelas.

3.4.3. 1995-2003: La generalización y unificación de los modelos

La tercera etapa en la evolución de las clasificaciones de algoritmos genéticos paralelos se extiende desde mediados de la década de 1990 hasta nuestros días. Los trabajos de investigación en esta etapa se caracterizan por tratar de independizarse totalmente del hardware subyacente, frecuentemente analizando implementaciones de diversos modelos y tratando de ofrecer taxonomías genéricas, que contemplen la mayoría de los modelos existentes.

El estudio de los modelos distribuidos domina esta etapa, motivado por de la difusión de las arquitecturas paralelas distribuidas de bajo costo (clusters de PCs o workstations), aunque continúan desarrollándose implementaciones para supercomputadores de memoria compartida.

Tomassini, en su reseña sobre algoritmos genéticos (Tomassini, 1995), y en su reseña posterior, específica sobre algoritmos genéticos paralelos (Tomassini, 1999), hace referencia a varios modelos de programación paralela aplicada a los algoritmos genéticos. Siguiendo el modelo de Schwehm (1996) ya comentado, el autor reconoce la existencia de jerarquías de aplicación del paralelismo:

- La aplicación del paralelismo a nivel de la función de fitness conduce al modelo maestro-esclavo, que Tomassini denomina modelo de paralelismo global.
- La aplicación del paralelismo a nivel de la población permite implementar el modelo de islas, constituido por poblaciones semi-independientes que implementan

el algoritmo genético serial, con el agregado de migraciones de individuos. Tomassini aún considera la versión stepping stone, donde la migración solamente ocurre entre subpoblaciones vecinas.

- Cuando el paralelismo se aplica a nivel de individuos, se obtienen los algoritmos genéticos celulares o de grano fino. La terminología celular es adoptada por su similitud con los autómatas celulares con reglas de transición estocásticas, de acuerdo al trabajo de Whitley (1993) y a trabajos propios (Tomassini, 1993a, 1993b). Menciona la adecuación de este modelo a las arquitecturas SIMD MasPar y CM200, que proporcionan comunicación eficiente implementada por hardware.

Tomassini reconoce para cada modelo las versiones sincrónicas y asincrónicas, dependiendo del mecanismo de coordinación entre poblaciones mediante las comunicaciones, ofreciendo una clasificación dependiendo del comportamiento espacial y temporal del algoritmo genético paralelo , la cual se resume en la Figura 3.11.

	Grano Grueso		***Grano Fino***
Sincronismo/Nivel	Población	Individuo	Fitness
Sincrónico	Islas Sincrónicas	Celular	Maestro-esclavo
Asincrónico	Islas Asincrónica	Estocástico	Maestro-esclavo asincrónico.

Figura 3.11: Modelos de la clasificación de Tomassini

Fuera de su clasificación, Tomassini coloca a los algoritmos genéticos paralelos híbridos –que combinan enfoques de paralelismo–, y a los algoritmos genéticos paralelos no uniformes –cuyas poblaciones utilizan diferentes codificaciones, mecanismos evolutivos y/o parámetros–, englobados dentro de una categoría de modelos de algoritmos genéticos paralelos no estándar.

Venkateswaran, Obradovic y Raghavendra (1996) proponen independientemente un modelo de algoritmos genéticos cooperativos, que corresponde a un enfoque intermedio entre el modelo global y el modelo de islas. Múltiples procesos ejecutan un algoritmo genético global en paralelo, cada uno con la totalidad de la población, trabajando sobre diferentes áreas del espacio de soluciones, lo cual mejora las probabilidades de obtener mejores soluciones. El modelo es heterogéneo, en el sentido de que cada proceso cuenta con sus propios mecanismos de selección y reemplazo, operadores de cruzamiento y mutación. Esta característica hace al algoritmo menos sensitivo a una determinada elección de estos parámetros. Adicionalmente, existe cooperación entre los procesos, que intercambian individuos periódicamente con el objetivo de resolver el problema eficientemente.

El modelo fue implementado de acuerdo a un modelo maestro-esclavo sobre una plataforma distribuida (red de computadores) y aplicado a la resolución de problemas de asignación, reportándose resultados óptimos para pequeños problemas y mejoras significativas de performance respecto al modelo serial para problemas donde la relación entre tiempo de comunicación y tiempo de ejecución es pequeña.

En los trabajos de Whitley et al. (1997a, 1997b, 1998, 1999), se realizan exhaustivos análisis del modelo de islas con subpoblaciones y migración. En estos trabajos se estudian detalladamente los parámetros que controlan la operación de migración, el intervalo de migración y el tamaño de la población migrada. Conjuntamente, se analizan diferentes representaciones para problemas de optimización y las mejoras obtenidas con respecto a la calidad de la solución al utilizar el modelo de islas especialmente para problemas linealmente separables. Sus resultados muestran la capacidad de los algoritmos con multipoblaciones semi-independientes para resolver este tipo de problemas. Aún para problemas con funciones no separables, los resultados indican que el uso de migración introduce una mejora de performance significativa respecto al modelo panmíctico.

Los trabajos en el área por parte de Cantú-Paz (1998a, 1998b) y Cantú-Paz y Goldberg (1999), culminaron con la publicación de un completo texto que reúne análisis teóricos y aspectos de implementación y estudios de eficiencia para algoritmos genéticos paralelos (Cantú-Paz, 2000a).

La clasificación de algoritmos genéticos paralelos de Cantú-Paz extiende los modelos clásicos de Grefenstette (1981), y coincide con las propuestas de Adamidis (1994), Lin, Punch y Goodman (1994), Alba, Cotta y Troya (1999a) y Tomassini (1999). Tomando en cuenta la distribución de la población y de las operaciones aplicadas durante el mecanismo evolutivo, Cantú-Paz reconoce las siguientes categorías:

- modelo maestro-esclavo
- modelo de población múltiple.
- modelo de grano fino.
- modelos paralelos híbridos.

Cantú-Paz estudia exhaustivamente los algoritmos genéticos paralelos modelo maestro-esclavo de población única en su trabajo de 1998. El modelo se basa en una sencilla aplicación del paralelismo a los algoritmos genéticos, que consiste en distribuir la evaluación de la función de fitness entre un conjunto de procesos esclavos, manteniendo las operaciones (selección, cruzamiento y mutación) en un único proceso maestro (Cantú-Paz, 1998b).

El autor destaca que el enfoque cuenta con ciertas ventajas respecto a otros modelos de algoritmos genéticos paralelos: utilizan el mismo mecanismo de exploración del espacio de soluciones que los algoritmos genéticos seriales, lo que hace aplicables directamente las consideraciones de diseño deducidas teórica y experimentalmente para éstos; la sencillez de su implementación, que los popularizó especialmente durante las primeras etapas del desarrollo de algoritmos genéticos paralelos y las mejoras de performance significativas al resolver problemas con funciones de fitness complejas.

En una primera etapa, Cantú-Paz estudia el modelo maestro-esclavo sincrónico, para obtener cotas inferiores simples para la eficiencia del modelo, ya que este enfoque introduce puntos de sincronización antes de comenzar a procesar la siguiente generación. De este modo, analiza el tiempo de ejecución, identificando sus componentes de cálculo y de comunicación de información entre procesos, obteniendo resultados para el número óptimo de procesadores, el speedup y la eficiencia para el modelo sincrónico. El tiempo de cálculo requerido por el proceso evolutivo depende del tamaño de la población, que no puede reducirse arbitrariamente sin una pérdida potencial de la calidad de la solución. El tiempo de evaluación de la función de fitness puede disminuirse aumentando el número de procesos esclavos, de modo de asignar menor cantidad de individuos a cada uno de ellos. El tiempo de comunicación depende del número de procesos esclavos utilizado, y el overhead de comunicación será mayor al aumentar la cantidad de procesos.

Cantú-Paz argumenta que la eficiencia de las implementaciones maestro-esclavo crece cuando las evaluaciones son más costosas, logrando alcanzar speedups casi lineales para problemas complejos. Cita como ejemplos los trabajos de Grefenstette (1995) sobre entrenamiento de robots, y de Punch et al. (1993) en el área de selección y clasificación de características. De acuerdo a esta idea, es posible identificar un problema adecuado para resolver con este modelo midiendo el tiempo de evaluación serial de la función de fitness y comparándolo con el tiempo requerido por las operaciones de evolución, cuya paralelización de las operaciones de evolución no es usual ni útil, ya que necesitan información global y la mejora de performance se pierde en tareas de comunicación y sincronización.

En su análisis del modelo maestro-esclavo sincrónico, Cantú-Paz muestra que la eficiencia llega al 50% en la configuración óptima, lo cual indica que aproximadamente la mitad del tiempo los procesadores se encuentran ociosos, esperando que el maestro envíe datos para procesar. Sin embargo, en el modelo sincrónico el maestro no puede enviar nuevos datos hasta que todos los esclavos hayan finalizado su tarea. La sincronización es necesaria si se desea que el comportamiento del algoritmo paralelo sea exactamente igual a la del serial, pero es posible evitarlo si se está dispuesto a aceptar que el modelo paralelo se comporte de modo diferente.

En el modelo maestro-esclavo asincrónico, tan pronto como un esclavo finaliza su tarea envía su resultado al maestro, quien inserta los individuos cuya función fue evaluada en la población. Luego, el maestro genera nuevos individuos, y los envía a los esclavos disponibles. Este modelo tiene el potencial de ser más eficiente que el sincrónico, en especial si los tiempos de evaluación de la función de fitness no son constantes para todos los individuos. Como contrapartida, la complejidad algorítmica de este modelo es mayor, ya que se introducen nuevos parámetros que controlan el comportamiento del algoritmo genético.

Cantú-Paz vincula los modelos maestro-esclavo asincrónicos con el modelo sincrónico con salto generacional de De Jong, ya descrito (De Jong, 1975). La diferencia entre estos modelos consiste en la manera en la cual se incorporan los nuevos individuos a la población, asunto que debe resolverse en cada uno de los modelos maestro-esclavo. Algunas de las alternativas consisten en insertar los nuevos individuos aleatoriamente, reemplazar a los peores individuos de la generación anterior, o competir con otros individuos para ganar un lugar en la población.

Si la inserción de individuos se basa en sus valores de fitness, la presión de selección se incrementará, y el algoritmo genético podría necesitar mayores poblaciones para obtener una solución de calidad similar a la del algoritmo con menor presión. El paralelismo asincrónico puede tener otros efectos sobre el algoritmo genético, además de incrementar la presión de la selección. El más notable es que los individuos pueden retornar de los esclavos en un orden diferente al que fueron creados, al finalizar ciertos esclavos sus evaluaciones antes que otros. Aguardar por los individuos ordenadamente es posible, pero en este caso decrece la performance. Cantú-Paz cita el estudio empírico de Davison y Rasheed (1999), que indica que el efecto de este mecanismo aleatorio no es significante en el funcionamiento del algoritmo genético, aún para muchos procesos esclavos, del orden del tamaño de la población.

Cantú-Paz concluye que si bien los algoritmos maestro-esclavo asincrónicos son algo más complejos de implementar, en general la ganancia en performance supera el pequeño esfuerzo adicional de implementación. Sin embargo, debe tenerse en cuenta que el no determinismo de los procesos asincrónicos hace a este tipo de algoritmos genéticos paralelos más difíciles de analizar teóricamente que su modelo sincrónico.

El modelo maestro-esclavo no parece ser intuitivamente la mejor opción de diseño para un modelo de población distribuida, pero Cantú-Paz lo analiza para obtener una cota inferior a la performance de los algoritmos paralelos con poblaciones múltiples distribuidas.

En su línea de trabajo de aplicación del paralelismo a los algoritmos genéticos desordenados, Lamont, Gates y Merkle (1997) propusieron una implementación utilizando la biblioteca de desarrollo de aplicaciones paralelo-distribuidas MPI. El modelo, basado en el algoritmo ***fast messy genetic algorithm*** original de Goldberg et al. (1993), está concebido para explotar las características de cálculo local y global de los algoritmos genéticos desordenados. La escalabilidad del modelo propuesto se estudia en un trabajo posterior por parte de los mismos autores (Lamont et al, 1998).

En su trabajo de 1999, Alba y Troya analizan la importancia del sincronismo en las migraciones para algoritmos genéticos paralelos con poblaciones distribuidas. Para su trabajo, identifican a los modelos de grano fino y de grano grueso como subclases de un único modelo de algoritmos genéticos paralelos formado por un conjunto de subalgoritmos comunicados. Proponen la denominación de modelos distribuidos y celulares para los modelos de grano grueso y fino respectivamente, considerando que la granularidad de un problema paralelo refiere a la relación entre trabajo computacional y comunicación, mientras que en el caso de los modelos de algoritmos genéticos paralelos se diferencian en el modo que estructuran la población. Los modelos distribuidos cuentan con subpoblaciones numerosas y subalgoritmos poco relacionados, mientras que los modelos celulares cuentan con pocos subalgoritmos fuertemente relacionados, que operan con pocos individuos (Alba y Troya, 1999).

Como resultado de numerosos trabajos donde efectúan comparaciones entre modelos distribuidos y celulares sincrónicos y asincrónicos, (Alba et al, 1999b, 1999c), (Alba y Troya, 1999b, 2001) los autores confirman que en general los modelos asincrónicos superan en término de performance y calidad de resultados a los modelos sincrónicos, confirmando resultados similares obtenidos por otros autores.

La investigación y el desarrollo de implementaciones de algoritmos genéticos paralelos para plataformas de hardware específicas disminuyen en comparación con el período anterior, pero no desaparecen por completo.

En este sentido, Baraglia y Perego (1999) estudian la implementación de modelos paralelos de grano fino sobre multiprocesadores con conexión de hipercubo. En su breve taxonomía discriminan entre el modelo centralizado con población panmíctica y los modelos distribuidos de grano fino y de grano grueso. Los autores ofrecen un análisis comparativo entre algoritmos genéticos paralelos de grano fino y de grano grueso implementados sobre un multiprocesador nCUBE 2 de 128 nodos para la resolución del TSP, mostrando la potencial ventaja de escalabilidad de la implementación de grano fino al crecer el número de procesadores.

Sprave (1999) definió un modelo formal unificado para las estructuras de población no panmícticas en algoritmos evolutivos. El objetivo de su trabajo consiste en definir una propuesta para la terminología y el modelo matemático de las estructuras multipoblacionales, hasta el momento dominadas por términos y modelos análogos a los del modelo secuencial o vinculados con la computación de alta performance. Sprave propone modelar las estructuras de población mediante hipergrafos, estudiando los modelos de migración y de grano fino. El entorno definido por Sprave se presenta como una poderosa herramienta para el cálculo de medidas de la presión de selección, entre las cuales el trabajo de Sprave incluye el cálculo del diámetro de la estructura de población y de la probabilidad de posesión (takeover probability), vinculada con el concepto de tiempo de posesión (takeover time), que mide el número de generaciones necesarias para que el mejor individuo de la población inicial saturar la totalidad de la población, asumiendo que el único operador aplicado es el de selección.

Davidson y Rasheed (1999) estudian los efectos del paralelismo global aplicado a los algoritmos genéticos de estado estacionario. El resultado principal de la investigación empírica realizada con el algoritmo GADO (Genetic *Algorithm for Design Optimization* (Rasheed, 1998)) prueba que el modelo de paralelismo maestro-esclavo permite obtener mejoras de performance significativas sin afectar la convergencia del mecanismo evolutivo del algoritmo genético serial.

Un original modelo híbrido de poblaciones distribuidas con secciones compartidas es presentado en el trabajo de Hiroyasu, Miki y Watanabe (1999). El diseño peculiar del modelo permite lograr diversidad y alta precisión en las soluciones, características necesarias para la resolución de problemas de optimización multiobjetivo.

Berger y Barkaoui (2003) presentaron un modelo híbrido con poblaciones solapadas denominado PHGA (*Parallel Hybrid Genetic Algorithm*) aplicado al problema de ruteo de vehículos con ventanas de tiempo. El modelo combina la potencialidad de un mecanismo evolutivo paralelo y con un operador de relajación parcial con constantes temporales para lograr una exploración exhaustiva del espacio de búsqueda. El trabajo de estos investigadores incluye innovaciones a los operadores de evolución del algoritmo genético estándar, para incorporar conceptos emergentes de técnicas de optimización promisorias como las heurísticas de inserción o las colonias de hormigas. De acuerdo a los resultados presentados, el modelo ha alcanzado su objetivo de lograr una mejora de la intensidad y de la diversificación de la búsqueda. El modelo PHGA se reporta como un algoritmo competitivo y de mejor performance que las metaheurísticas conocidas para los problemas de ruteo de vehículos con ventanas de tiempo.

En su trabajo de 1999, Nowostawski y Poli presentan una taxonomía unificada que reúne un amplio número de modelos propuestos en la literatura. Su trabajo contempla varias categorías, siguiendo un enfoque globalizador, para tomar en cuenta la mayoría de los modelos descritos en este trabajo, en un intento de que su propuesta comprenda a la totalidad de las clasificaciones elaboradas hasta ese momento (Nowostawski y Poli, 1999a). Ocho categorías son identificadas por Nowostawski y Poli, quienes distinguen entre los siguientes modelos:

- Modelo maestro-esclavo, con sus versiones sincrónicas y asincrónicas.
- Modelo distribuido de subpoblaciones con migración.
- Modelo de poblaciones estáticas solapadas, sin migración.
- Modelo masivamente paralelo.
- Modelo de poblaciones dinámicas, con solapamiento.
- Modelo de estado estacionario.
- Modelo desordenado paralelo.
- Modelos híbridos.

En el modelo monopoblación maestro-esclavo, se distingue entre versiones sincrónicas y asincrónicas, de acuerdo al criterio adoptado para coordinar las poblaciones antes de iniciar una nueva generación.

El modelo distribuido de subpoblaciones con migración es considerado uno de los enfoques más difundidos de aplicación de paralelismo a los algoritmos genéticos. El nuevo operador de migración caracteriza a este modelo, y permite discriminar entre sus diferentes variantes de acuerdo a sus parámetros, topología de migración, tasa de migración, frecuencia de migración y esquema de selección y reemplazo de individuos migrados. En su clasificación, los autores mencionan al ya extinto modelo stepping stone como un caso particular de este modelo. En los extremos de este enfoque se posicionan los algoritmos genéticos paralelos de grano fino y de grano grueso, de acuerdo a la cantidad de subpoblaciones y los individuos asignados a ellas. Los autores indican que los algoritmos genéticos paralelos de grano grueso son generalmente implementados en modelos de máquinas paralelas MIMD distribuidas, mientras que los de grano fino son adecuados para arquitecturas SIMD de memoria compartida. El principal inconveniente de este modelo consiste en la dificultad de implementar soluciones escalables para grandes problemas, lo cual lleva generalmente a diseñar modelos híbridos que permitan un uso más inteligente del hardware paralelo disponible, introduciendo jerarquías de procesamiento.

El modelo de poblaciones estáticas solapadas es similar al anterior, pero no cuenta con un operador de migración. El intercambio de información entre las subpoblaciones se realiza a través de áreas de solapamiento, formadas por individuos que pertenecen a más de un deme. La ausencia de comunicaciones explícitas hace a este modelo adecuado para su implementación en arquitecturas paralelas SIMD, realizando los intercambios a través de memoria compartida.

Los algoritmos genéticos masivamente paralelos son una extensión del modelo anterior, con un número muy elevado de demes y pocos individuos –idealmente solo uno– asignado a cada deme. El mecanismo de intercambio de material genético en este modelo se conoce como difusión, y se encuentra limitado por la estructura subyacente a la población, usualmente en forma de grilla. Este modelo es adecuado para computadores masivamente paralelos que proporcionen mecanismos eficientes de comunicación entre elementos de procesamiento.

El modelo de poblaciones dinámicas con solapamiento es un aporte de los propios Nowostawski y Poli (1999b), cuyos orígenes se remontan a los trabajos previos de Nowostawski (1997, 1998a y 1998b). Surge como una propuesta híbrida que combina la evaluación distribuida de fitness del modelo maestro-esclavo y el modelo de subpoblaciones estáticas con solapamiento. En cada generación, los demes se crean dinámicamente, siendo este mecanismo de reorganización el encargado del intercambio de información entre subpoblaciones. La principal ventaja de la introducción de la reorganización de los demes es eliminar el efecto de los individuos más lentos en el modelo maestro-esclavo. Cada deme se crea tan pronto como existan suficientes individuos para ello, y en ese momento se inicia la nueva generación del algoritmo genético. Desde el punto de vista del procesamiento paralelo, corresponde a un algoritmo MIMD maestro-esclavo asincrónico totalmente escalable, adecuado para la ejecución en máquinas distribuidas o masivamente paralelas.

El mecanismo de actualización continua de la población de los algoritmos genéticos de estado estacionario posibilita una aplicación sencilla de las técnicas de paralelismo. La operadores de selección y reemplazo deben aplicarse secuencialmente sobre los individuos en la sección crítica del algoritmo, mientras los restantes operadores pueden realizarse en paralelo.

Los autores incluyen una categoría para los algoritmos genéticos paralelos desordenados. Las características de este modelo de algoritmos genéticos permiten aplicar técnicas específicas de paralelismo, tal como se comentaron al analizar el trabajo de Merkle y Lamont (1993).

Los modelos híbridos combinan diferentes enfoques para paralelizar los algoritmos genéticos. Principalmente definen jerarquías de ordenamiento entre poblaciones, aplicándose en cada una de las jerarquías enfoques de paralelización diferentes, con el objetivo de combinar las ventajas de dos o más modelos de paralelismo. Como ejemplo, es común combinar múltiples poblaciones con el modelo maestro-esclavo o con el modelo de grano fino. Su estructura los hace adecuados para implementar sobre multiprocesadores. Estos modelos son estudiados en detalle en el texto de Cantú-Paz (2000c), bajo el nombre de modelos jerárquicos.

En el año 2002, Alba y Tomassini presentaron una completa reseña sobre paralelismo y algoritmos evolutivos (Alba y Tomassini, 2002) donde analizan los modelos de algoritmos evolutivos paralelos, aspectos de implementación y herramientas conjuntamente con análisis teóricos de las medidas de eficiencia computacional. En su clasificación los autores distinguen entre modelos *panmícticos, estructurados estándares* que incluyen a los modelos distribuidos y celulares y *estructurados no estándares* que corresponden a los modelos no homogéneos, jerárquicos e híbridos variados. Complementariamente, se presentan clasificaciones para implementaciones de algoritmos genéticos paralelos basadas en tres criterios: el *modelo*, discriminando entre implementaciones de grano fino, de grano grueso y masivamente paralelos; el *tipo*, identificado entre implementaciones orientadas a la aplicación, al algoritmo y *toolkits;* y en la *aplicación*, en donde se agrupan las implementaciones de acuerdo al campo en el cual fueron utilizadas como herramientas para la resolución de problemas, entre los que figuran la optimización combinatoria e investigación operativa, diseño de redes de comunicaciones, aplicaciones financieras, diseño de circuitos electrónicos y VLSI y diseño en ingeniería.

La reseña de Alba y Tomassini reúne varios aspectos de originalidad que merecen ser destacados. El estudio conjunto de modelos teóricos y el análisis de implementaciones específicas es de gran ayuda como material introductorio al área de investigación en algoritmos evolutivos paralelos. El estudio de las áreas de aplicación constituye por sí mismo una novedad en el campo de las taxonomías. Por último, la extensa lista de referencias bibliográficas es un material muy útil para conocer los principales trabajos de los investigadores en el área.

Los trabajos relevantes de la tercer etapa en la evolución del diseño y las clasificaciones de algoritmos genéticos paralelos se ofrecen en la tabla de la Figura 3.12.

Autor	*Año*	*Línea de trabajo*
Tomassini	1995 1999	Clasificación de AGP
Adamidis. Venkateswaran, Obradovic, Raghavendra	1996	Modelo de AG cooperativos.
Whitley et al.	1997, 1998, 1999	Análisis del modelo de islas con subpoblaciones y migración
Cantú-Paz	1998, 1999, 2000	Diseño e implementación de AGP.
Lamont, Gates, Merkle	1997	Paralelismo para AG desordenados, usando MPI.
Alba y Troya	1999	Importancia del sincronismo en las migraciones para AGP con poblaciones distribuidas.
Baraglia y Perego	1999	AGP de grano fino sobre multiprocesadores con conexión de hipercubo
Hiroyasu,Miki,Watanabe	1999	AGP híbrido de poblaciones distribuidas con secciones compartidas
Davidson y Rasheed	1999	Paralelismo global aplicado a AG de estado estacionario
Sprave	1999	Modelo formal unificado para estructuras de población no panmícticas.
Nowostawski y Poli	1999	Taxonomía unificada de AGP.
Berger, Barkaoui,Bräysy	2001	Modelo con poblaciones solapadas para AG de estado estacionario
Alba y Tomassini	2002	Paralelismo y algoritmos evolutivos.

Figura 3.12: Principales trabajos de la etapa de generalización y unificación de los modelos.

3.4.4. Panorama del estado actual

El trabajo desarrollado en el área en los últimos años es extenso, y el esfuerzo por lograr un modelo unificado para categorizar las diferentes maneras de aplicar las técnicas de procesamiento paralelo a los algoritmos genéticos puede notarse en los comentarios al final de la sección precedente.

Sin duda, la referencia inmediata al estado actual de investigación en el área de los algoritmos genéticos paralelos lo constituye el texto de Cantú-Paz (2000c). Sin embargo, la clasificación utilizada por el mencionado autor no contempla algunos enfoques de paralelismo que poseen características intrínsecas que los diferencian de los modelos estándar discriminados en el referido texto. Complementariamente, la categorización de modelos híbridos, aunque extensamente analizada por Cantú-Paz, reúne una amplia gama de enfoques, y es susceptible de un refinamiento.

Por los motivos expuestos, la clasificación de Nowostawski y Poli (1999a) emerge como una alternativa más comprehensiva y específica. A su vez, se presenta como una clasificación más flexible y útil, al incluir categorías para modelos no tradicionales, no existentes en la taxonomía clásica, como los algoritmos genéticos desordenados paralelos o los modelos de estado estacionario.

Es posible vincular la categorización de Nowostawski y Poli con los modelos desarrollados a lo largo de la historia de investigación de los algoritmos genéticos paralelos.

El modelo maestro-esclavo generalmente ha conservado esta denominación a lo largo de la evolución de las categorizaciones. De todos modos ciertos autores han designado con otros términos al modelo, a saber:

- han sido denominados modelos de paralelismo global por autores como Gordon y Whitley (1993), Schwehm (1996), y Tomassini (1995, 1999).
- como modelo de paralelismo estándar por Adamidis (1994a).
- como modelo micrograno por Lin et al. (1994).
- como modelo de implementación centralizada por Bianchini y Brown (1993) y por Baraglia y Perego (1999).

El modelo distribuido de subpoblaciones con migración es el que ha recibido mayor variedad de denominaciones por parte de los investigadores. Este modelo incluye a los siguientes enfoques:

- el modelo PGA de Petty y Leuze (1987).
- los modelos de islas de Gorges-Schleuter (1989), Gordon y Whitley (1993), Whitley et al. (1990, 1997, 1998a, 1999) y Tomassini (1995, 1999).
- el modelo forkingGA de Tsutsui y Fujimoto (1993).
- el modelo sin migración de Shonkwiler (1993).
- el modelo heterogéneo de Tanese (1989b).
- el modelo distribuido de grano fino de Maruyama, Hirose y Konagaya (1993).
- el modelo de implementación distribuida de Bianchini y Brown (1993).
- el modelo de descomposición de Adamidis (1994a).
- el modelo de grano grueso de Lin et al. (1994) y de Baraglia y Perego (1999).
- el modelo regional de Schwehm (1996).
- el modelo de algoritmos genéticos distribuidos de Alba y Troya (1999a).

El modelo denominado de poblaciones estáticas solapadas sin migración por Nowostawski corresponde al modelo de grano fino original de Robertson (1987) e incluye las siguientes propuestas:

- el modelo de grano fino de Mühlenbein (1989), Manderick y Spiessen (1989) y Baraglia y Perego (1999).
- el modelo de vecindades de Gorges-Schleuter (1989).
- el modelo de polinización de Goldberg (1989a).
- el modelo de grano fino de Lin et al. (1994).
- el modelo local de Schwehm (1996).
- el modelo estocástico de Tomassini (1995, 1999).

La categoría de algoritmos genéticos masivamente paralelos incluye a los modelos :

- modelo mpdGA de Baluja (1992, 1993a).
- el modelo de algoritmos genéticos paralelos celulares de Gordon y Whitley (1993), Tomassini (1998) y Alba y Troya (1999a).

El modelo de poblaciones dinámicas, con solapamiento es reciente y original de Nowostawski y Poli (Kwasnicka y Nowostawski, 1997), (Nowostawski 1998a, 1998b), (Nowostawski y Poli 1999b), por lo cual no existen referencias directas sobre otros modelos similares.

La categoría de algoritmos genéticos paralelos de estado estacionario incluye las siguientes propuestas:

- el modelo de islas de Levine (1993, 1994).
- el modelo maestro-esclavo de Davidson y Rasheed (1999).
- el modelo híbrido multipoblación de Berger, Barkaoui y Bräysy (2001).

Los esfuerzos por diseñar AG desordenados paralelos han sido limitados, en esta línea de trabajo pueden mencionarse :

- el modelo de distribución de la evaluación de la función de fitness propuesto por Goldberg (1990).
- el modelo de distribución de datos en la fase primordial Merke y Lamont (1993).
- el modelo fmGA (fast messsy genetic Algorithm) propuesto por Lamont, Gates y Merkle (1997).

Los modelos híbridos engloban a aquellos que combinan enfoques y estrategias, conjuntamente con los que cuentan con originales propuestas que aún no han constituido un tópico de investigación por sí mismos. Dentro de ellos es posible incluir a :

- el modelo de comunidad de Goldberg (1989a).
- los modelos de algoritmos genéticos cooperativos de Adamidis (Adamidis y Petridis, 1996), (Adamidis, 1997) y Venkateswaran et al. (1996).
- el modelo distribuido jerárquico con meta-optimización de parámetros de Masakazu et al. (1993).
- el modelo de organización en clusters denominado modelo de implementación semidistribuido de Bianchini y Brown (1993).
- el modelo distribuido con secciones compartidas de Hiroyasu, Miki y Watanabe (1999).
- los denominados modelos jerárquicos por Cantú-Paz (2000c).

De acuerdo a la enumeración precedente, es claro que la propuesta de unificación de terminologías y taxonomías de modelos de algoritmos genéticos paralelos constituye una necesidad importante para la comunidad científica interesada en este tema de investigación.

La alternativa de enfoques es variada, y determinar el modelo adecuado para la resolución de un determinado problema de búsqueda u optimización no será tarea sencilla sin tener un panorama de las características de los modelos, sus particularidades de implementación y las ventajas y desventajas respecto a los restantes enfoques.

En tal sentido, la intención de esta sección ha sido presentar las diferentes propuestas realizadas a lo largo del período de investigación en el área de los algoritmos genéticos paralelos, resumiendo los detalles principales de los artículos, textos y reseñas disponibles.

Complementando el trabajo de categorización de Nowostawski y Poli, este capítulo reúne los modelos más relevantes de algoritmos genéticos paralelos y las enmarca dentro de la propuesta de taxonomía unificada.

3.5. Conclusiones

La aplicación de las técnicas de programación de alta performance se ha difundido extensamente en la última década como un mecanismo para mejorar la eficiencia computacional de los algoritmos genéticos. Dividiendo la población entre un conjunto de elementos de procesamiento, los algoritmos genéticos paralelos permiten abordar problemas difíciles de resolver o de grandes dimensiones, hallando resultados de buena calidad en tiempos razonables.

Adicionalmente, los modelos paralelos de algoritmos genéticos permiten introducir un mecanismo de evolución diferente al de los modelos secuenciales, explorando simultáneamente distintas secciones del espacio de búsqueda. Este mecanismo tiene la potencialidad de aportar la diversidad necesaria en la población para mejorar los resultados de los algoritmos seriales en problemas complejos.

Este capítulo ha presentado una descripción de las técnicas de programación de alta performance y las propuestas de su aplicación con el objetivo de mejorar el desempeño y calidad de resultados de los algoritmos genéticos. Complementariamente, se presentó una reseña de la evolución histórica en el diseño y clasificación de algoritmos genéticos paralelos, culminando con un panorama del estado actual en el área, presentando una visión unificadora de las diferentes propuestas de algoritmos genéticos paralelos.

CAPÍTULO 4

EL PROBLEMA DE STEINER GENERALIZADO

"Man muss immer generalisieren."
(" Uno siempre debería generalizar ")

CARL JACOBI

The Mathematical Experience,
P. Davis, R. Hersh, 1981.

4.1. **Introducción**

Una red de comunicaciones se compone de un conjunto de nodos emisores/receptores de información, conectados por enlaces que permiten la transmisión de la información de modo unidireccional o bidireccional. Habitualmente se suele diferenciar entre ciertos nodos distinguidos, denominados ***nodos terminales*** de la red de comunicaciones, sobre los que se plantean requisitos de conectividad, y otros nodos que se utilizan exclusivamente como conectores intermedios en los caminos de comunicación de información entre nodos terminales.

Uno de los principales problemas de construcción de redes de comunicaciones plantea el diseño de una topología de interconexión de sus nodos de modo que la red verifique ciertas características de confiabilidad. La confiabilidad de una red es una medida que evalúa la probabilidad de éxito en la comunicación entre pares de nodos, y constituye un factor influyente en la calidad del servicio ofrecido a los usuarios. La evaluación de los parámetros exactos que determinan la confiabilidad de una red de comunicaciones es un problema NP-difícil (Ball, 1979) (Provan y Ball, 1983); un enfoque alternativo consiste en utilizar los denominados ***parámetros de vulnerabilidad***, que se encuentran relacionados con la medida de confiabilidad de la red de comunicaciones, pero su evaluación no es tan compleja. El número de caminos diferentes que permiten comunicar a nodos terminales de la red constituye un importante parámetro para determinar la vulnerabilidad de la red, ya que la existencia de múltiples caminos posibilita disponer de varias vías alternativas para el transporte de datos entre nodos. En general, el número mínimo de caminos requeridos no es el mismo para todo par de nodos terminales de la red, sino que diferentes pares de nodos pueden requerir distintos números de caminos alternativos que permitan la comunicación entre ellos.

El rápido desarrollo de la infraestructura de redes de comunicaciones, diseño de software y servicios de Internet ha sido propulsado por la gran demanda de comunicaciones de datos en los últimos veinte años. Por este motivo, se ha renovado el interés en los problemas de diseño de redes de comunicaciones, comprendiendo entre otros problemas de ubicación óptima de antenas, la asignación de frecuencias en telefonía celular, y variados problemas de diseño estructural que refieren al ruteo de información a través de la red (Corne, Oates y Smith, 2000) (Pedrycz y Vasilakos 2001).

Como consecuencia del continuo crecimiento en el tamaño de las redes de comunicaciones, las instancias subyacentes de problemas de optimización vinculados con el diseño de las redes frecuentemente plantean un desafío a los algoritmos de resolución existentes. Por este motivo la comunidad de investigadores se encuentra hoy en día en la búsqueda de nuevos algoritmos, capaces de sustituir a los métodos exactos tradicionales cuya pobre eficiencia los hace frecuentemente inaplicables para resolver en tiempos razonables problemas de la vida real, de grandes dimensiones.

En este sentido, las técnicas heurísticas se han propuesto como métodos aplicables a la resolución de los problemas de diseño de redes de comunicaciones confiables. Aunque los métodos heurísticos no garantizan la obtención de un valor óptimo verdadero para un problema de optimización, permiten encontrar valores casi óptimos, cuya calidad puede ser suficiente para satisfacer las demandas de los diseñadores. Entre una amplia gama de heurísticas y técnicas modernas de optimización, las técnicas de computación evolutiva han emergido como métodos flexibles y robustos para la resolución de los complejos problemas de optimización subyacentes al diseño de redes de comunicaciones, así como otras áreas de aplicación en la industria, matemática, economía, telecomunicaciones y bioinformática entre otros.

Esta Tesis se enfoca en una amplia clase de problemas de diseño de redes de comunicaciones que pueden modelarse bajo el denominado *Problema de Steiner Generalizado*. Considerando una red de comunicaciones con ciertos nodos distinguidos denominados terminales, el Problema de Steiner Generalizado consiste en diseñar una subred de mínimo costo, que verifique un conjunto de requerimientos prefijados de conexión entre pares de nodos terminales.

Usualmente la minimización de los costos de las conexiones en una red de comunicaciones y la maximización de su confiabilidad son objetivos contrapuestos. Como ejemplo, un modelo que minimice el costo total de la red sin contemplar restricciones adicionales de conectividad (como los introduce la exigencia de que exista un mínimo número de caminos de conectividad entre pares de nodos terminales) conduciría a una topología de red en forma de árbol. Esta clase de soluciones, que no introducen redundancia de caminos, no son útiles en la mayoría de los escenarios de la vida real, ya que no son capaces de tolerar ni siquiera una simple falla en uno de los componentes de la red. Cualquier deficiencia en uno de los nodos o en algún enlace afectaría la operatividad de la red, impidiendo la comunicación al menos entre un par de nodos terminales.

El Problema de Steiner Generalizado incorpora requisitos adicionales de conectividad sobre pares de nodos terminales, contemplando las exigencias de muchos problemas de la vida real. El problema se aplica entonces al diseño de redes de comunicaciones en las cuales la alta confiabilidad de la red queda garantizada por la existencia de diferentes caminos alternativos entre pares de nodos terminales. De este modo se logra que el funcionamiento de la red sea más robusto, capaz de resistir a fallas en sus componentes.

El capítulo se organiza del modo que se describe a continuación. En la sección siguiente se presenta formalmente el Problema de Steiner Generalizado, su modelo matemático, así como sus variantes y métodos aproximados y heurísticos propuestos para su resolución. A continuación se presenta una reseña de las propuestas de aplicación de técnicas de computación evolutiva para la resolución del problema. Por último, la sección final del capítulo ofrece las conclusiones sobre el estudio del problema considerado.

4.2. El Problema de Steiner Generalizado

Esta sección presenta una caracterización formal del Problema de Steiner Generalizado, incluyendo un caso de estudio que ejemplifica la complejidad intrínseca del problema. A continuación se discute una formulación matemática general que puede utilizarse como base para la solución del problema, y se presentan variantes que simplifican la especificación general del problema y antecedentes de aplicación de técnicas heurísticas para la resolución del problema generalizado y sus variantes.

4.2.1. Formulación del Problema de Steiner Generalizado

A continuación se presenta una formulación del Problema de Steiner Generalizado basada en la incluida en el compendio de problemas de optimización para los cuales no existe algoritmo conocido de resolución en tiempo polinomial de Kahn y Crescenzi (2003).

Considérense los siguientes elementos:

- Un grafo no dirigido $G = (V,E)$, siendo V el conjunto de nodos y E el conjunto de aristas que representan a los enlaces de comunicación bidireccionales (full-duplex).
- Una matriz C de costos no negativos asociados a las aristas del grafo G.
- Un subconjunto fijo del conjunto de nodos $T \subseteq V$, llamado conjunto de *nodos terminales*, de cardinalidad $n_T = |T|$, tal que $2 \leq n_T \leq n$, siendo n = $|V|$ la cardinalidad del conjunto de nodos V.
- Una matriz simétrica $R = r_{ij}$ con $i,j \in T$, de dimensión n_T x n_T, cuyos elementos son enteros no negativos que indican los requerimientos de conectividad – cantidad de caminos disjuntos requeridos– entre todo par de nodos terminales (i,j).

El Problema de Steiner Generalizado propone encontrar G_T, un subgrafo de G de costo mínimo, que sea un grafo de cubrimiento del conjunto de nodos terminales. Todo par de nodos terminales $i,j \in T$, $i \neq j$ deberán ser localmente r_{ij} arista−conexos en G_T. Esto significa que deben existir r_{ij} caminos disjuntos, que no comparten aristas, entre los nodos terminales i y j en G_T.

Sobre los nodos no pertenecientes al conjunto de nodos terminales no se plantean requisitos de conectividad. Estos nodos, conocidos como *nodos de Steiner*, pueden formar parte o no de la solución óptima, dependiendo de la conveniencia de utilizarlos.

La descripción previa corresponde a la versión arista−conexa del Problema de Steiner Generalizado, utilizada para modelar problemas sobre redes en las cuales se supone a los enlaces (aristas) sujetos a fallas, pero los nodos se suponen perfectos. El problema admite una formulación análoga para el caso en que sean los nodos los que pueden fallar. En este caso la especificación del problema exige la existencia de caminos nodo−disjuntos, es decir caminos que no comparten nodos, entre pares de nodos terminales.

4.2.2. Un ejemplo de Problema de Steiner Generalizado

El siguiente caso de estudio del Problema de Steiner Generalizado ejemplifica la complejidad intrínseca del problema.

Considérese el grafo G de la Figura 4.1, donde los nodos terminales se presentan en color oscuro y se han etiquetado por las letras A – E, mientras que los nodos de Steiner se presentan con color claro (para simplificar el diagrama, no han sido etiquetados). Los costos asociados a cada arista se especifican en la figura, mientras que los requerimientos de conectividad se presentan en la Tabla 4.1.

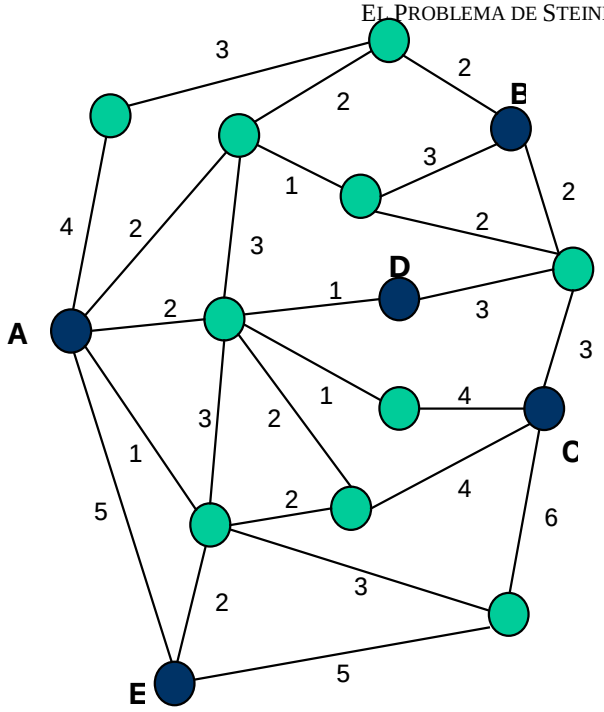


Figura 4.1: Grafo original G para el ejemplo de Problema de Steiner Generalizado

Nodos terminales	***Número de Caminos Requeridos***
A ↔ B	3
B ↔ D	2
A ↔ E	1
A ↔ C	3
B ↔ C	3
A ↔ D	2

Tabla 4.1: Requerimientos de conectividad sobre los nodos terminales del grafo G.

La figura 4.2 presenta una solución de costo mínimo para la instancia del Problema de Steiner Generalizado planteado sobre el grafo *G* de la Figura 4.1 Para conectar los 5 nodos terminales y verificar los requerimientos de conectividad, la solución incluye 8 de los 10 nodos de Steiner originales, seleccionados como los óptimos en términos de número y posición considerando los costos de los enlaces correspondientes. Asimismo, el grafo solución incluye 16 de las 26 aristas del grafo original, disminuyendo el costo total de un valor original de 71 a un valor de 36 en el diseño óptimo. La solución presentada no es única, ya que otras soluciones óptimas pueden existir para el problema.

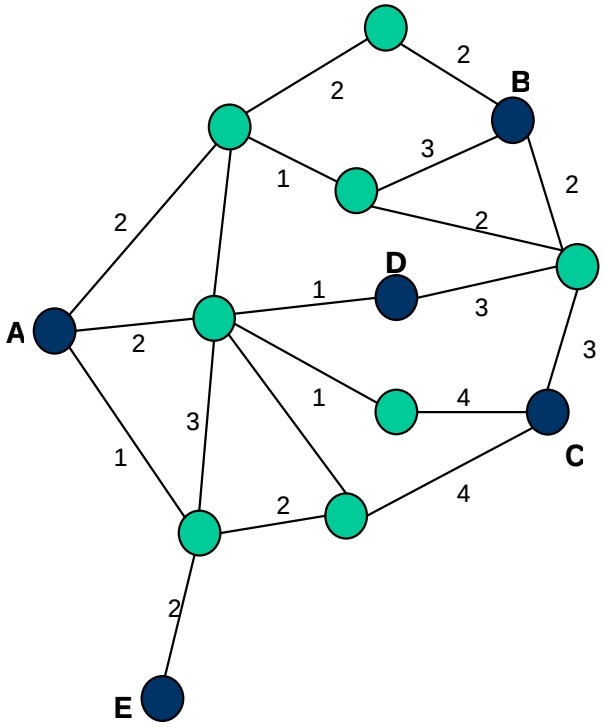


Figura 4.2: Solución al Problema de Steiner Generalizado sobre el grafo G.

Un problema como el presentado en el caso de ejemplo modela el diseño de redes de comunicaciones en situaciones como la que se presenta en la Figura 4.3, donde se esquematiza el diseño de una red de comunicaciones en un escenario de la vida real. Este escenario heterogéneo incluye varios elementos de transmisión y recepción de datos: computadores conectados en redes de área local (LAN) y redes de área global (WAN), enrutadores, antenas y dispositivos móviles comunicados por una red celular.

El esquema ilustra los beneficios de contar con múltiples caminos alternativos para garantizar la confiabilidad de las comunicaciones entre terminales de la red. Como ejemplo, considerando el diseño de la red presentada en la Figura 4.3, puede apreciarse que existen dos caminos disjuntos en enlaces que conectan a los nodos terminales A y B y tres caminos disjuntos en enlaces que conectan a los nodos terminales B y C, implicando un diseño robusto capaz de tolerar fallas en los enlaces sin afectar la operatividad de la red. La Figura 4.3 ejemplifica un diseño de alta confiabilidad para la red de comunicaciones, pero obviamente no constituye una solución de costo mínimo, ya que incluye múltiples enlaces superfluos que sería posible eliminar, optimizando el diseño de la red presentada.

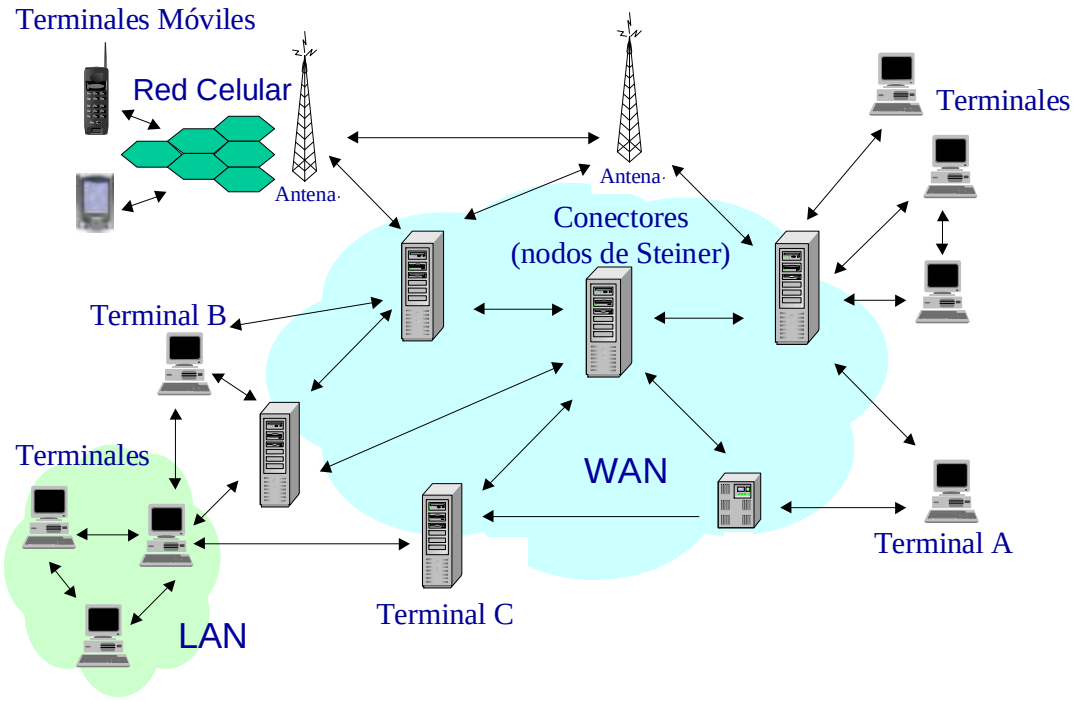


Figura 4.3: Una red de comunicaciones confiable en un escenario de la vida real.

4.2.3. Modelo Matemático del Problema de Steiner Generalizado

Winter (1987) introdujo un modelo matemático como un problema de Programación Lineal Entera Mixta para la versión arista−conexa del Problema de Steiner Generalizado. La formulación se presenta en las Ecuaciones 4.1 – 4.8 de la Figura 4.4.

El modelo se basa en asignar múltiples variables a cada arista $(i,j) \in E$. Se utiliza una variable binaria x_{ij} que indica la presencia o ausencia de cada arista (i,j) en una solución, y un conjunto de variables reales y_{ij}^{kl} que indican la cantidad de flujo enviado a través de la arista (i,j), en el sentido de i a j, en un camino que conecta a los nodos terminales k y l. La variable y_{ij}^{kl} tomará un valor nulo sólo en caso que la arista (i,j) no sea utilizada en ningún camino que conecte a dos nodos terminales k y l, y tomará un valor mayor que cero en caso de ser utilizada al menos en un camino en la solución del problema.

Las soluciones del Problema de Steiner Generalizado quedan determinadas por las soluciones óptimas del problema de Programación Lineal Entera Mixta presentado en la Figura 4.4.

$$Min \sum_{(i,j)\in E} C_{ij} \cdot x_{ij} \quad (4.1)$$

sujeto a:

$$x_{ij} \geq y_{ij}^{kl} + y_{ji}^{kl} \qquad \forall (i,j) \in E, \forall k,l \in T, k \neq l \quad (4.2)$$

$$\sum_{(k,j)\in E} y_{kj}^{kl} \geq r_{kl} \qquad \forall k,l \in T, k \neq l \quad (4.3)$$

$$\sum_{(i,l)\in E} y_{il}^{kl} \geq r_{kl} \qquad \forall k,l \in T, k \neq l \quad (4.4)$$

$$\sum_{(p,j)\in E} y_{pj}^{kl} - \sum_{(i,p)\in E} y_{ip}^{kl} \geq 0 \qquad \forall k,l \in T, \forall p \in V \setminus \{k,l\} \quad (4.5)$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in E \quad (4.6)$$

$$y_{ij}^{kl} \in \Re, y_{ij}^{kl} \geq 0 \quad \forall i,j : (i,j) \in E, \forall k,l \in T, k \neq l \quad (4.7)$$

$$r_{kl} \geq 0 \qquad \forall k,l \in T, k \neq l \quad (4.8)$$

Figura 4.4: Modelo matemático del GSP como problema de Programación Lineal Entera

En el modelo de la Figura 4.4, la Ecuación 4.1 presenta la función objetivo del problema, correspondiente a la minimización de la suma de los costos de las aristas presentes en la solución. Las restricciones de la Ecuación 4.2 acoplan las variables binarias x_{ij} con las variables reales y_{ij}^{kl} y y_{ji}^{kl}, indicando que si la arista (i,j) ha sido utilizada (en cualquier sentido) en algún camino que conecta a algún par de nodos terminales (k,l), entonces debe estar presente en la solución final. Las restricciones dadas por las Ecuaciones 4.3 y 4.4 aseguran que el flujo desde un nodo terminal k hacia otro nodo terminal l sea al menos r_{kl},, el número de caminos impuesto como requerimiento de conectividad para ese par de nodos terminales del problema. La restricción de la Ecuación 4.5 corresponde a las ecuaciones clásicas de conservación del flujo. Por último, las Ecuaciones 4.6, 4.7 y 4.8 indican las restricciones de integralidad y no-negatividad para las variables involucradas en el modelo.

Llamando $U = \{u_{ij}\}$ a la solución óptima del problema de Programación Lineal Entera Mixta presentado en la Figura 4.4, el conjunto de aristas determinado por $\{(i,j) \in E \,/\, u_{ij} = 1\}$ define al subgrafo G_T, solución del Problema de Steiner Generalizado.

Una formulación matemática análoga puede obtenerse para la versión nodo-conexa del Problema de Steiner Generalizado, utilizando una técnica de partición de nodos para transformar el grafo G.

En el trabajo que se reporta en esta Tesis no se utilizó directamente el modelo matemático presentado para el diseño de los algoritmos propuestos para el Problema de Steiner Generalizado. El subproblema de flujo máximo inducido por las Ecuaciones 4.3, 4.4 y 4.5 es equivalente al problema de hallar los caminos entre pares de nodos terminales sobre un grafo G, Este subproblema de flujo, cuyas soluciones son siempre enteras, fue utilizado como base de un procedimiento para determinar la factibilidad de las soluciones encontradas por los algoritmos evolutivos diseñados.

4.2.4. Variantes de problemas de Steiner

La complejidad del Problema de Steiner Generalizado obedece precisamente a la generalidad de su planteo. El hecho de permitir requerimientos variables de conectividad implica la existencia de un número variable de caminos disjuntos entre pares de nodos terminales en la solución del problema.

Ciertas variantes del problema simplifican este requerimiento, exigiendo un número fijo de caminos disjuntos entre pares de nodos terminales. Esta clase de problemas se conoce como *problemas de k conexión*. El número fijo *k*, entero no negativo, indica la cantidad de caminos disjuntos requeridos entre todo par de nodos terminales.

El caso más simple de problema de Steiner es aquel que solo exige un camino entre todo par de nodos. La solución de este problema particular tiene topología de árbol, ya que corresponde a un árbol de cubrimiento de costo mínimo de los nodos terminales del grafo. Por este motivo, esta variante es conocida por el nombre de *problema del árbol de Steiner*,

A continuación se comentan estas variantes del Problema de Steiner Generalizado. En lo que resta del documento se referirá genéricamente al conjunto compuesto por estas variantes simplificadas y por el problema generalizado como la *clase de problemas de Steiner*. A esta clase pertenecen también otras variantes de problemas de diseño de redes de comunicaciones confiables relacionados con los anteriores, que no serán estudiadas en detalle en este trabajo.

Problemas de k conexión

Los problemas de *k* conexión pueden considerarse como casos particulares del Problema de Steiner Generalizado, en los cuales se ha eliminado la posibilidad de exigir números variables de caminos disjuntos entre nodos terminales. En los problemas de *k* conexión los requerimientos de cantidad de caminos son constantes, y tienen un valor igual al número entero *k* para todo par de nodos terminales.

Es usual encontrar este tipo de problemas en aplicaciones donde se desea reducir los costos de incorporar enlaces alternativos entre nodos de la red, manteniendo un número bajo de caminos disjuntos entre todo par de nodos terminales. Como ejemplo, en el diseño de redes de comunicación de fibra óptica, una solución 2 arista−conexa permite introducir el nivel mínimo de redundancia de caminos entre todo par de nodos terminales, limitando la cantidad de enlaces adicionales necesarios, los cuales usualmente tienen un costo elevado.

La resolución de un problema de *k* conexión involucra hallar un subgrafo *k* arista−conexo que cubra al conjunto de nodos terminales de un grafo dado, es decir un grafo que no puede transformarse en desconexo removiendo menos de *k* aristas. Como consecuencia, una red de comunicaciones *k* arista−conexa es capaz de continuar funcionando correctamente, garantizando la comunicación entre todo par de nodos, aún luego de *k*−1 fallos en sus enlaces.

Los problemas de *k* conexión han sido estudiados ampliamente. Grötschel, Monma y Stoer (1995a) presentan una formulación de los problemas de *k* conexión como problemas de Programación Lineal. En particular, el *problema de biconexión* (caso en que $k = 2$), que propone el diseño de un grafo biconexo que cubre un conjunto de nodos terminales con costo mínimo, ha sido considerado frecuentemente como modelo para redes de comunicaciones y transporte. Monma, Munson y Pulleyblank (1990) estudiaron el caso particular del problema de biconexión donde la función de costo cumple la desigualdad triangular entre aristas.

Un caso genérico de problemas de *k* conexión ($k \geq 3$) para la variante en que los costos cumplen la desigualdad triangular es estudiado por Bienstock, Brickell y Monma (1990). Los autores analizan los casos de arista−conectividad, presentando teoremas que restringen la clase de subgrafos de cubrimiento *k*−conexos al conjunto de subproblemas estudiados, y el caso de nodo−conectividad, presentando un teorema análogo con la hipótesis adicional $|V| \geq 2.k$.

Problema del árbol de Steiner

El problema del árbol de Steiner es un caso particular del problema de k conexión, que corresponde al caso en que se exige un único camino entre pares de nodos terminales de la red de comunicaciones (caso en que $k = 1$ en un problema de k conexión).

Este problema ha sido extensamente estudiado como uno de los problemas clásicos de optimización combinatoria en el área de cubrimiento de grafos. El requerimiento de conexión unitario entre pares de nodos terminales se soluciona encontrando un árbol de cubrimiento de costo mínimo del conjunto de nodos terminales.

Como consecuencia de su amplia aplicabilidad, existe una gran diversidad de algoritmos exactos y heurísticas que resuelven el problema del árbol de Steiner, al respecto pueden consultarse las reseñas incluidas en las tesis de Ravi (1993) y Robledo (2001). En los trabajos citados se mencionan y estudian algoritmos exactos como los de enumeración de árboles de cubrimiento, los de enumeración de topologías, algoritmos basados en programación dinámica, y en relajación lagrangeana, entre otros. La mayoría de las heurísticas empleadas usualmente se basan en la utilización de técnicas *greedy* siguiendo caminos de costo mínimo o costo promedio, aunque también se ha propuesto utilizar estrategias heurísticas basadas en técnicas de contracción o de cubrimiento.

4.2.5. Complejidad de los problemas de Steiner

El Problema de Steiner Generalizado es NP−difícil, tal como se indica en el compendio de problemas de Kahn y Crescenzi (2003), tanto en su versión con requerimientos de caminos disjuntos en aristas como en su versión para caminos disjuntos en nodos.

Los problemas de k conexión son asimismo NP−difíciles. Luebke y Provan (2000) probaron la posibilidad de reducir una forma especial del problema de determinar un ciclo hamiltoniano a la versión planar del problema de biconexión.

El propio problema del árbol de Steiner, que plantea la restricción menos general en cuanto al número de caminos exigidos, es NP−completo (Karp, 1972). Este problema fue uno de los primeros siete problemas para los cuales Karp probó su completitud no polinomial.

Inclusive versiones simplificadas de estos problemas, como los casos con grafo planar o bipartito en conjunto de nodos terminales y conjunto de nodos no terminales, continúan siendo problemas NP−difíciles.

4.2.6. Enfoques para la resolución de la clase de problemas de Steiner

Como consecuencia de la complejidad de la clase de problemas de Steiner, su resolución utilizando algoritmos exactos se hace cada vez menos tratable al aumentar el tamaño de los problemas. Por tal motivo, se buscan soluciones alternativas utilizando heurísticas que permitan encontrar soluciones de calidad aceptable en tiempos razonables.

Una variada gama de heurísticas específicas y algoritmos de aproximación se han propuesto para la resolución de la clase de problemas de Steiner, en especial para el problema del árbol de Steiner y los problemas de k conexión. A manera de ejemplo, es posible citar las heurísticas específicas más populares que se han desarrollado para la resolución del problema del árbol de Steiner: Minimum Cost Paths (MCP) propuesta por Takahashi y Matsuyama (1980), Distance Network Heuristic (DNH) propuesta por Kou et al. (1981), Average Distance Heuristic (ADH) propuesta por Rayward-Smith (1983) y revisada por Rayward-Smith y Clare (1986). Otras propuestas se basan en utilizar estrategias greedy de inserción de nodos de Steiner como métodos de búsqueda local, como en los métodos presentados por Minoux (1990) y Voss (1990, 1992). El trabajo de Plesnik (1992) presenta una reseña de las heurísticas propuestas para el problema del árbol de Steiner.

Respecto al Problema de Steiner Generalizado, es posible indicar que no se ha estudiado en su formulación más genérica del mismo modo que han sido estudiados el problema del árbol de Steiner y, hasta cierto nivel, los problemas de *k* conexión. En general los investigadores han concentrado sus esfuerzos en resolver casos particulares o simplificados. En tal sentido, cabe mencionar desde el trabajo pionero de Steiglitz, Weiner y Kleitman (1969) quienes resolvieron una variante del problema aplicada al diseño de circuitos lógicos, pasando por la completa reseña de Winter (1987), que contiene una referencia al problema generalizado y su modelo matemático, culminando con los varios trabajos de Grötschel, Monma y Stoer (1990, 1991, 1995b) sobre el diseño de redes confiables. Más recientemente, conviene destacar los trabajos de Ravi (1993) y de Agrawal, Klein y Ravi (1994) quienes propusieron un algoritmo aproximado para un problema equivalente al Problema de Steiner Generalizado, así como los trabajos de Williamson, Goemans, Mihail, y Vazirani (1995); Jain (1998) y Vazirani (1998) que presentan un algoritmo de aproximación para el caso generalizado y sus variantes. Los resultados teóricos más recientes sobre algoritmos de aproximación para generalizaciones del problema del árbol de Steiner pueden encontrarse en la reseña de Vazirani (2000).

Con respecto a la utilización de metaheurísticas para la resolución de la clase de problemas de Steiner, nuevamente debe mencionarse que las referencias existentes se enfocan principalmente en la resolución de los casos simplificados. Existen varios antecedentes de aplicación de algoritmos evolutivos al problema del árbol de Steiner y a los problemas de *k* conexión, los cuales se reseñan y comentan en detalle en la sección siguiente. Con relación a la utilización de otras metaheurísticas para el problema del árbol de Steiner, es posible mencionar la aplicación de Tabú Search por parte de Xu, Chiu y Glover (1995, 1996, 1997, 1998), Tabú Search con Path Relinking propuesto por Bastos y Ribeiro (1999), GRASP paralelo hibridizado con búsqueda local propuesto por Martins (2000), búsqueda local hibridizada utilizada por Poggi de Aragão, Ribeiro, Uchoa y Werneck (2001) y GRASP estándar e híbridizado utilizado por Ribeiro, Uchoa y Werneck (2002), Asimismo, en varias ocasiones se ha propuesto utilizar Simulated Annealing para la resolución del problema del árbol de Steiner (Dowsland, 1991), (Osborne y Gillet, 1991) (Bravo y Candia, 1997).

Ninguno de los trabajos mencionados en el párrafo precendente abordó la resolución del caso generalizado del problema de Steiner. En nuestro entorno de trabajo se cuenta con el antecedente inmediato de aplicación de la metaheurística Ant Systems al Problema de Steiner Generalizado en su versión de arista conexión (Robledo, 2001) y a su versión de nodo conexión (Cancela et al., 2003). De acuerdo al relevamiento realizado, estos antecedentes constituyen los únicos intentos de abordar el caso generalizado del problema de Steiner mediante una técnica genérica, sin hacer uso de heurísticas específicas.

4.2.7. Casos Particulares del Problema de Steiner Generalizado

Ciertos casos particulares del Problema de Steiner Generalizado son de interés por reducir su complejidad a un orden polinomial, existiendo inclusive algoritmos de complejidad lineal en algunos casos específicos (Winter, 1987). Por este motivo, estos casos particulares han sido abordados por al comunidad de investigadores utilizando técnicas exactas.

Los casos particulares surgen de reducir la complejidad del problema generalizado, asumiendo propiedades sobre los elementos del grafo en cuestión. Los casos particulares más sencillos corresponden a instancias del problema donde los costos de las aristas se suponen idénticos o booleanos. Estos casos particulares se comentan brevemente a continuación.

El Problema de Steiner Generalizado con costos uniformes se plantea cuando los costos asociados a las aristas del grafo que representa a la red de comunicaciones son idénticos. Esta variante del problema se resuelve hallando un subgrafo con mínimo número de aristas que satisfaga los requerimientos r_{ij} de conectividad para los nodos terminales. Un algoritmo de orden polinomial fue presentado por Couch y Frank para resolver esta versión reducida del problema con costos idénticos en los enlaces, suponiendo permitida la existencia de aristas paralelas entre nodos. El caso en que no se permiten aristas paralelas en la construcción de la solución del problema ha sido atacado solo en una versión del problema de k conexión, donde además se asumen requerimientos de conectividad uniformes entre nodos terminales, por parte de Harara (según menciona Stoer (1996)). Un caso particular del Problema de Steiner Generalizado con costos uniformes se plantea cuando todos los nodos de la red son considerados terminales ($T = V$). Este problema reducido ha sido estudiado por Christofides y Whitlock (1981) quienes presentaron múltiples aplicaciones prácticas del problema.

Otro caso especial del Problema de Steiner Generalizado se presenta cuando se considera una matriz de costos booleana, resultando el problema conocido como *Augmentation Problem*, cuya formulación propone hallar el mínimo número de aristas tal que al agregarlas al grafo en cuestión se satisfacen los requerimientos de conexión dados por la matriz R. Este problema ha sido ampliamente estudiado en el pasado. Es posible encontrar referencias al estudio de la versión 2 arista−conexa del problema por parte de Eswaran y Tarjan (1976) y de Goldner y Rosenthal (1979). El caso genérico de k conexión fue estudiado por Watanabe y Nakamura (1987), mientras que un caso más genérico de Augmentation Problem con requerimientos de conexión variables es estudiado por Kajitani, Ueno y Wada (1988), Cai y Sun (1989) y Frank (1990), quien presentó algoritmos de resolución de orden polinomial. El algoritmo de menor complejidad conocido hasta el momento, que incrementa la conectividad de aristas de un grafo a un nivel k en tiempo polinomial fue propuesto por Gusfield, Martel y Naor (1990). Salvo este último algoritmo, el resto permiten soluciones que utilicen aristas paralelas.

Asimismo, existen casos particulares que reducen la complejidad del Problema de Steiner generalizado tomando ventaja de la topología del grafo considerado. Como ejemplo, Winter (1987) probó que existen algoritmos capaces de resolver el problema en tiempo lineal en casos especiales, como aquellos en los que el grafo considerado es serie−paralelo o de Halin.

4.3. Técnicas Evolutivas Aplicadas a la Resolución de los Problemas de Steiner

Esta sección presenta el relevamiento realizado sobre trabajos que han propuesto utilizar algoritmos evolutivos para la resolución de la clase de problemas de Steiner. Se presentan breves reseñas de los trabajos estudiados, comentando las principales características de las propuestas y detalles de los resultados obtenidos en cada caso.

Luego del estudio del estado del arte realizado, fue posible concluir que algunas variantes simples del problema de Steiner han sido abordadas, aunque no existen antecedentes de aplicación de técnicas de computación evolutiva al Problema de Steiner Generalizado.

A continuación se presentan referencias que reportan trabajos donde se aplican técnicas de programación evolutiva al problema del árbol de Steiner y a los problemas de k conexión.

El primer trabajo en el área data de 1989, fecha en que Hesser, Manner y Stucky, propusieron un algoritmo genético para la optimización de árboles de Steiner a partir de árboles de cubrimiento (Hesser et al, 1989). El problema resuelto en este caso utiliza un algoritmo exacto para hallar árboles de cubrimiento con $O(n \log n)$ operaciones y luego utiliza un algoritmo genético para optimizar el costo de los diferentes árboles obtenidos al agregar puntos de Steiner. El algoritmo codifica árboles de Steiner utilizando cromosomas con la información geográfica de las posiciones de los puntos de Steiner y considera como función de fitness el costo asociado al árbol de Steiner que es posible construir considerando los puntos originales y los puntos de Steiner codificados en el individuo. El algoritmo es configurado de acuerdo a los valores de los parámetros sugeridos por Grefenstette (1986). Los resultados se comparan con los obtenidos aplicando la técnica de optimización Simulated Annealing y el algoritmo ADH de Rayward-Smith y Clare (1986) una de las mejores heurísticas desarrolladas específicamente para el problema considerado. Los resultados del trabajo no fueron concluyentes, ya que el algoritmo genético propuesto por los autores no mostró diferencias significativas con los obtenidos mediante el algoritmo ADH, mostrando asimismo una similar eficiencia computacional. De todos modos, tomando en cuenta que el algoritmo ADH es una heurística especializa para la resolución del problema, los resultados de la aplicación del algoritmo evolutivo fueron considerados promisorios. En trabajos posteriores, los autores continuaron su trabajo en el área de los algoritmos genéticos aplicados al problema del árbol de Steiner (Hesser et al, 1991).

La propuesta de Hesser et al. no consistió literalmente en la resolución del problema del árbol de Steiner, ya que el algoritmo genético se utilizaba exclusivamente para la optimización de árboles previamente construidos en base a algoritmos exactos. En 1993, Kapsalis, Rayward-Smith y Smith realizaron la primer propuesta de resolución del problema del árbol de Steiner utilizando algoritmos genéticos (Kapsalis et al., 1993). La propuesta utiliza una codificación basada en representar los nodos de Steiner y el algoritmo genético trabaja sobre un espacio de individuos que incluye soluciones no factibles al problema, sobre los cuales se aplica un término de penalización al evaluar la función de fitness. El tamaño de los problemas abordados por Kapsalis et al. es reducido y por ello se utilizan poblaciones muy poco numerosas, de solo diez individuos. El artículo presenta un estudio exhaustivo de los parámetros de configuración del algoritmo genético y concluye aplicándolo a la resolución del problema sobre grafos con un grado de conectividad bajo (grafos *dispersos*). Los autores reportan resultados promisorios tanto en término de calidad de resultados como de eficiencia computacional para las instancias de problemas estudiados, comparativamente con los resultados obtenidos utilizando Simulated Annealing por parte de Dowsland (1991).

Davis, Orvosh, Cox y Qiu propusieron un algoritmo genético para un ***Problema de Diseño de Redes Confiables*** consistente en la asignación de anchos de banda a canales sobre los cuales se imponen restricciones de tráfico y restricciones de tolerancia a fallas en los enlaces. El algoritmo genético propuesto utiliza cromosomas que codifican los diversos parámetros del problema, como las capacidades de los enlaces, conjuntamente con permutaciones de la lista de requerimientos de tráfico. El procedimiento de evaluación es utilizado para determinar si se satisfacen las restricciones de ruteo y los requerimientos de tolerancia a fallas, y es complementado con una técnica *greedy* para reparar soluciones que violan las restricciones del problema. El modelo de evolución propuesto es del tipo de estado estacionario, como consecuencia del considerable esfuerzo computacional requerido para la evaluación de la función de fitness y la aplicación del mecanismo de corrección de individuos. El problema abordado por los autores no se corresponde exactamente con el de *diseño* de una red de comunicaciones ya que la topología de la red se considera prefijada, y el algoritmo genético se utiliza para optimizar la asignación de recursos (ancho de banda de los canales de comunicación) de modo de cumplir los requisitos de operatividad y confiabilidad. Los autores reportan la obtención de buenos resultados en términos de calidad y eficiencia computacional, al compararlos con otros dos métodos de resolución basados en programación entera y técnicas *greedy* incrementales (Davis et al., 1993).

En sus trabajos de 1993 y 1994, Julstrom trabajó sobre una variante del problema del árbol de Steiner, conocida como Problema de Steiner Rectilíneo. En este problema el grafo subyacente es planar, en el sentido de que los nodos terminales son puntos que se suponen ubicados en el plano euclídeo y deben conectarse mediante un árbol de Steiner. Adicionalmente, las conexiones entre nodos terminales y/o puntos de Steiner deben ser exclusivamente horizontales o verticales. El problema, que consiste en hallar el árbol de Steiner *rectilíneo* de costo mínimo, tiene aplicación directa en las áreas de diseño de redes de comunicaciones, sistemas mecánicos en edificaciones, tableros de circuitos y paquetes VLSI. Julstrom propuso un algoritmo genético de estado estacionario que utiliza una codificación mixta (binaria y no-binaria) basada en un mapeo de árboles de cubrimiento de n nodos con strings de largo $n-2$ derivado de una conocida prueba original de Prufer (1918) de la fórmula de Cayley (1889) para la enumeración de árboles de cubrimiento en un grafo completo. Un operador de cruzamiento modificado se introduce para evitar la excesiva pérdida de material genético en la descendencia. En el trabajo inicial de Julstrom (1993) el algoritmo genético no permitía alcanzar la misma calidad de soluciones que otras técnicas heurísticas. En su revisión de 1994, el autor presenta un mecanismo de inicialización mejorado que permite alcanzar soluciones apenas inferiores a las de otras técnicas. Posteriormente, el autor continuó sus investigaciones sobre el Problema de Steiner Rectilíneo. En sus publicaciones más recientes (Julstrom, 2001, 2002, 2003) propuso algoritmos genéticos híbridos que utilizan heurísticas específicas para la construcción de la población inicial y operadores de búsqueda local. Estas mejoras permiten obtener resultados de mayor calidad y más eficientemente que el algoritmo genético estándar, comparables con los resultados de técnicas heurísticas específicas para el problema.

En su tesis de doctorado de 1994, Esbensen propuso alternativas a la anterior propuesta de Kapsalis et al, extendiendo sus ideas al implementar un algoritmo genético para resolver el problema del árbol de Steiner. Trabajando con una codificación binaria y un procedimiento de decodificación basado en la heurística determinística Distance Network Heuristic propuesta por Kou et al. (1981), Esbensen introdujo una diferencia significativa al enfoque previo de Kapsalis et al, proponiendo trabajar exclusivamente con soluciones factibles, evitando introducir términos de penalización en la función de fitness. Adicionalmente, Esbensen comprendió que utilizando la codificación binaria mencionada, pueden existir diferentes genotipos para el mismo árbol de Steiner. Por tal motivo, para mejorar la eficiencia del operador de cruzamiento y para independizarlo de las diferentes representaciones genotípicas, introdujo un operador probabilístico de inversión, que reordena las componentes de un vector genotipo, sin modificar su expresión fenotípica (Esbensen, 1994).

Sobre la idea de la propuesta anterior, Esbensen y Mazumder presentaron un algoritmo genético para el problema del árbol de Steiner y su aplicación al diseño de circuitos VLSI, que mostró una calidad superior en las soluciones obtenidas respecto a dos de las mejores heurísticas conocidas para el problema del árbol de Steiner: Minimum Cost Paths (MCP) propuesta por Takahashi y Matsuyama (1980) y Distance Network Heuristic (DNH) propuesta por Kou et al. (1981), siendo el algoritmo genético competitivo en términos de performance (Esbensen y Mazumder, 1994).

En su trabajo de 1997, Huang, Ma y Hsu propusieron la implementación de un algoritmo genético para la resolución del problema de 3 conexión (Huang et al, 1997a). El modelo de resolución se basa en resolver sucesivamente los problemas de encontrar los mejores tres caminos entre todo par de nodos terminales, utilizando el algoritmo genético propuesto. La codificación utilizada representa tres caminos entre cada nodo origen y destino, agregando información adicional sobre el diámetro de la red o restricciones de conectividad existentes. Para mantener disjuntos a los tres caminos codificados, los autores proponen utilizar un operador de cruzamiento de dos puntos especialmente diseñado, que incorpora el intercambio de nodos duplicados. Utilizando este operador modificado se trabaja con soluciones factibles al problema y no es necesario aplicar mecanismos de corrección sobre soluciones inválidas.

Los autores reportan mejoras en la eficiencia y calidad de soluciones obtenidas por su algoritmo, como consecuencia de no utilizar procesamiento adicional para el chequeo de las restricciones ni para corrección de soluciones no factibles. Comparan su implementación con el algoritmo genético propuesto por Davis et al. (1993), con métodos de dos fases basados en heurísticas locales como el algoritmo de Dijkstra y con otras heurísticas. Es de destacar la aplicabilidad del enfoque utilizado en este trabajo para abordar problemas de mayor complejidad topológica, como es el caso del Problema de Steiner Generalizado.

Una versión paralela del algoritmo comentado anteriormente se presenta en el trabajo de Huang et al. (1997b), estudiados, paralelismo en los requerimientos y paralelismo sobre la población del algoritmo genético. El primer enfoque corresponde a un caso de descomposición de dominio en programación paralela, mientras que el segundo corresponde a una paralelización del algoritmo genétcio de acuerdo al modelo de poblaciones múltiples, implementada para ejecutar sobre una máquina paralela compuesta por una red de transputers.

Desde un punto de vista diferente, varios artículos recientes han abordado diversos problemas de diseño y de asignación de recursos en redes de comunicaciones que involucran el trabajo con árboles de Steiner, y sus autores aplicado las ideas existentes sobre el uso de algoritmos genéticos para su resolución.

En esta línea de trabajo, Zhu, Wainwrigth y Schoenefeld utilizaron un algoritmo híbrido para hallar árboles de Steiner, combinando las ideas de Esbensen con técnicas heurísticas propias para la optimización de requerimientos de ruteo multipunto sobre una red de comunicaciones (Zhu et al, 1998).

Premkumar, Chu y Chou estudiaron la aplicación de un algoritmo genético para la resolución de un caso especial del problema del árbol de Steiner, denominado ***problema del árbol de Steiner estrella***. En esta variante del problema los nodos terminales deben conectarse haciendo uso de un único nodo de Steiner, mediante una topología en forma de estrella (Premkumar, Chu y Chou, 1999a, 1999b, 2000). El algoritmo genético propuesto utiliza representación binaria basada en representar la ubicación de los nodos de Steiner candidatos a funcionar como ***hub*** de la red de comunicaciones. Adicionalemente, se utiliza selección estocástica ($\mu+\lambda$), cruzamiento uniforme y el algoritmo de Prim (1957) para la evaluación de la función de fitness. Los autores comparan los resultados de su algoritmo genético con los obtenidos mediante la técnica Tabú Search utilizada por Xu, Chiu y Glover (1996), reportando la obtención de resultados de calidad similar pero mejorando la eficiencia computacional.

Complementando el trabajo anterior, los autores presentaron un estudio detallado de la eficiencia computacional de un algoritmo genético aplicado al problema de hallar un árbol de cubrimiento con restricciones en los grados de los nodos (Premkumar, Chu y Chou, 1999c). En este trabajo se analizan los factores que afectan el desempeño del algoritmo genético, estudiando el impacto de diferentes mecanismos de codificación y de los operadores evolutivos.

Voss y Gutenschwager investigaron sobre la hibridación entre algoritmos genéticos y la técnica de aprendizaje por troceado, originada en el campo de la inteligencia artificial. Las técnicas de aprendizaje por troceado se basan en plantear un conjunto de hipótesis específicas que se utilizan para hallar relaciones de alto nivel en el espacio de búsqueda de un problema. En el contexto de los algoritmos genéticos, Voss y Gutenschwager plantean definir los esquemas por troceado, combinando trozos superiores de forma tal que sean separables con una probabilidad baja por el operador de recombinación. Las investigaciones sobre el mecanismo híbrido fueron realizadas utilizando como caso de estudio el problema del árbol de Steiner (Voss y Gutenschwager, 1998).

En su tesis de maestría, Do estudió estrategias de ruteo multicasting utilizando algoritmos genéticos (Do, 1999). Posteriormente, en conjunto con Hwang, y Yang desarrollaron un algoritmo genético para resolver el problema del árbol de Steiner, aplicado al diseño de una solución para el problema de ruteo multicasting (Hwang et al, 2000). En su propuesta, cada cromosoma codifica las rutas posibles entre un nodo origen y un conjunto de nodos destino, de acuerdo a un método de codificación propuesto en un trabajo previo por parte de Hiramatsu, Shimamoto y Yamasaki (1993). Los autores reportan la obtención de resultados de calidad superior que los obtenidos utilizando la heurística RSR de Rayward-Smith y Clare (1986).

Con respecto al problema del ruteo multicasting, Xianwei, Changjia y Gang propusieron un algoritmo genético híbrido en su trabajo de 2000. El algoritmo genético implementado utiliza una representación binaria que indica el conjunto presente de nodos de Steiner en la solución que representa el individuo. La propuesta combina una heurística para decodificar el genotipo binario y construir el árbol de Steiner asociado a la representación, diversas técnicas estándar de reducción de redes y un mecanismo evolutivo para minimizar el costo del árbol de Steiner. Los autores reportan la obtención de soluciones de mejor calidad que las obtenidas con heurísticas deterministas y con Simulated Annealing, en tiempos de ejecución moderados, pero no presentan detalles numéricos (Xianwei et al, 2000).

Ljubic, Raidl y Kratica presentaron un algoritmo genético híbrido aplicado a la solución del Augmentation Problem con requerimientos de biconectividad sobre aristas (Ljubic et al, 2001). La propuesta combina una técnica de reducción y un algoritmo genético de representación binaria para las aristas candidatas a aumentar el grafo original. Los autores presentan dos estrategias para tratar las soluciones no factibles en el proceso evolutivo: el enfoque más sencillo detecta y descarta las soluciones no factibles, mientras que el segundo método introduce una estrategia de corrección de soluciones no factibles que propone agregar aristas de bajo costo hasta obtener un grafo biconectado. El algoritmo genético presentado constituye una variante del utilizado para el Augmentation Problem con requerimientos de biconectividad sobre nodos en un trabajo previo por dos de los autores (Ljubic y Kratica, 2000)

Más recientemente, Galiasso y Wainwright estudiaron el problema de ruteo multipunto, extendiendo el trabajo de Zhu et al. (1998) y su propuesta de algoritmo genético híbrido. El trabajo de Galiasso y Wainwright incluye ideas originales de Esbensen (1994) para hallar árboles de Steiner (Galiasso y Wainwright, 2001). Complementariamente, el algoritmo genético híbrido que proponen los autores tiene la capacidad de determinar el ordenamiento óptimo de procesamiento de requerimientos y de optimizar los porcentajes de ancho de banda asignados a diferentes caminos de comunicación. Los autores reportan mejores resultados que los obtenidos por Zhu et al. (1998) en términos de calidad de soluciones obtenidas.

Aplicado al problema de diseño de circuitos VLSI, Wakabayashi presentó recientemente un algoritmo genético para la resolución del problema del árbol de Steiner rectilíneo. En la propuesta de Wakabayashi, el cromosoma codifica la información topológica del árbol de Steiner recursivamente, indicando la relación padre–hijo entre los nodos. El árbol de Steiner se construye utilizando información geométrica determinada durante la evaluación de la función de fitness, aplicando un método constructivo basado en el tradicional algoritmo de Kruskal para obtener árboles de cubrimiento. La función de fitness cuantifica el costo total de la red y la máxima demora entre nodos origen y destino de comunicación. El algoritmo genético utiliza un mecanismo especial de cruzamiento, denominado intercambio de subárboles. La propuesta se presenta como efectiva en términos de calidad de soluciones de los árboles rectilíneos de Steiner obtenidos, aunque se menciona que la performance es mala comparada con algoritmos estándar, y se propone investigar sobre su mejora en el futuro (Wakabayashi, 2002).

Ninguno de los trabajos mencionados han abordado el problema de Steiner Generalizado utilizando técnicas evolutivas. En su lugar, se han limitado a estudiar las variantes más simples, presentadas en la sección precedente: los problemas de *k* conexión y el problema del árbol de Steiner.

4.4. Conclusiones

Este capítulo ha presentado el Problema de Steiner Generalizado, un problema que modela el diseño de redes de comunicaciones de alta confiabilidad topológica que constituyen el objetivo de esta Tesis.

Se ofreció una definición formal del problema, una formulación de su modelo matemático como problema de programación lineal entera y casos de estudio que ejemplifican la complejidad intrínseca del problema y su aplicabilidad en escenarios de redes de comunicaciones reales. Asimismo se presentaron las variantes simplificadas del problema generalizado: el Problema del Arbol de Steiner y los problemas de *k* conexión.

Conjuntamente, se reseñaron los antecedentes de aplicación de métodos aproximados y heurísticos propuestos para la resolución del caso generalizado del problema, y se presentó una reseña de las propuestas de aplicación de técnicas de computación evolutiva para la resolución de las variantes simplificadas del problema.

El estudio de trabajos relacionados con la utilización de heurísticas permitió concluir que no existen referencias de aplicación de técnicas evolutivas para la resolución del caso generalizado del problema estudiado, comprobándose que los investigadores se han limitado a estudiar las variantes simplificadas. Este hecho resalta la originalidad de la propuesta del trabajo que se reporta en esta Tesis.

CAPÍTULO 5

UN ALGORITMO GENÉTICO PARALELO PARA EL PROBLEMA DE STEINER GENERALIZADO

"You say you got a real solution
Well you know, We'd all love to see the plan.
……
You tell me that is evolution
Well you know, We all want to change the world."

J. LENNON, P. MC CARTNEY
Revolution, *The Beatles* (*The White Album*), 1968

5.1. **Introducción**

Uno de los principales objetivos del trabajo de Maestría consistió en investigar la aplicabilidad de los algoritmos evolutivos y en particular de los algoritmos genéticos para resolver problemas de diseño redes de comunicaciones confiables. Tomando en cuenta la dimensión de los complejos problemas de optimización involucrados, se propuso diseñar modelos paralelos de algoritmos genéticos, capaces de manejar la complejidad intrínseca del problema considerado y de utilizar de modo eficiente poblaciones numerosas, en caso de ser necesario.

Luego del estudio teórico de los algoritmos evolutivos y de las estrategias de procesamiento paralelo aplicables para mejorar su eficiencia, y después de identificar y analizar el Problema de Steiner Generalizado como un importante problema de optimización subyacente a los problemas de diseño de redes de comunicaciones confiables, este capítulo presenta la primera propuesta elaborada de un algoritmo genético paralelo específico para la resolución del Problema de Steiner Generalizado.

El capítulo se organiza del modo que se describe a continuación. Comienza presentando los aspectos vinculados con la codificación del problema para su resolución mediante un algoritmo genético paralelo, detalles sobre los operadores evolutivos empleados, una descripción de la función de fitness utilizada y las características del proceso de decodificación. A continuación se presentan los comentarios sobre la estrategia de paralelismo utilizada y su implementación. Por último, se introduce un conjunto de problemas de prueba diseñados específicamente para la evaluación del algoritmo implementado y se presentan los resultados numéricos correspondientes a las pruebas de validación del modelo, un detallado análisis de configuración de los múltiples parámetros del algoritmo y los resultados obtenidos sobre el conjunto de problemas de prueba diseñado. Para cerrar el capítulo, se resumen las conclusiones del trabajo y aspectos inconclusos que se plantearon para profundizar en el futuro.

5.2. Codificación del problema

La elección de una codificación adecuada para representar el problema que se desea resolver constituye un factor influyente sobre el funcionamiento del mecanismo evolutivo, que es destacado en múltiples referencias del área (Goldberg, 1989a) (Mitchell, 1996). En el caso de la resolución del Problema de Steiner Generalizado, los objetos a codificar son los grafos que representan diferentes diseños factibles, es decir que verifican los requisitos del problema, para la red de comunicaciones.

5.2.1. Antecedentes

El análisis de los trabajos previos relacionados presentado en el Capítulo 4 permitió identificar dos propuestas principales de representación aplicables a la clase de problemas de Steiner. Asimismo, se tomó conocimiento de una tercera alternativa poco utilizada en trabajos previos pero que se identificó como promisoria para ser aplicada al caso específico del Problema de Steiner Generalizado.

La primera idea consiste en codificar los grafos mediante una representación basada en aristas. En trabajos como los de Kapsalis et al. (1993) y Esbensen (1993, 1994) se propone el trabajo con representaciones binarias simples que indican las aristas presentes en un grafo, para codificar soluciones al problema del árbol de Steiner.

Un segundo enfoque, utilizado principalmente en las propuestas que abordan el problema del árbol de Steiner y sus variantes, sugiere codificar los grafos que representan la red de comunicaciones mediante listas de nodos. En estos problemas, donde las soluciones tienen topología de árbol, en los individuos solución no existirán ciclos y las aristas de cada árbol se determinan de modo trivial a partir del conjunto de nodos. Este tipo de representación no es utilizada cuando se trata con los problemas de k conexión o con el Problema de Steiner Generalizado, ya que la estructura de la red diseñada puede contener ciclos y no es posible determinar los caminos contando solamente con la información de los nodos presentes en la solución. Para adaptar esta representación para la resolución del caso generalizado del problema es necesario introducir información adicional respecto a las aristas involucradas en cada solución.

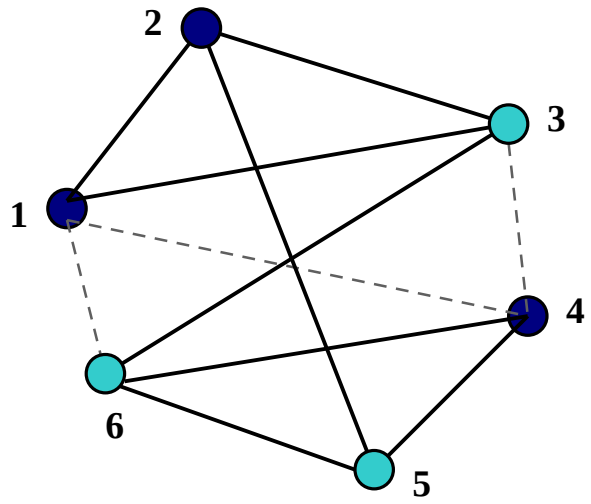
Por último, debe mencionarse una tercera propuesta de codificación, de la cual solo se encontró una referencia de aplicación. Corresponde a la utilizada en el trabajo de Hwang et al. (1997), donde los autores proponen una codificación basada en caminos en un algoritmo genético aplicado a la resolución del problema de 3 conexión. La propuesta consiste en utilizar un cromosoma que incluye la información relativa a tres caminos diferentes entre cada par de nodos terminales considerados como origen y destino. Se destaca la aplicabilidad de esta propuesta para resolver problemas cuya solución tiene una mayor complejidad topológica, como en el caso del Problema de Steiner Generalizado.

5.2.2. La codificación binaria utilizada

El esquema de representación considerado en este trabajo propone utilizar una codificación basada en aristas para representar soluciones factibles al Problema de Steiner Generalizado. Esta elección se realizó tomando en cuenta que una representación simple basada en codificar los nodos del grafo no sería sencilla de adaptar, debido a la ya mencionada imposibilidad de determinar caminos contando exclusivamente con la información de los nodos presentes en una solución.

Considerando las ventajas derivadas de su sencillez de implementación, se propuso trabajar con una codificación binaria simple. Cada cromosoma consta de un arreglo de bits cuyas posiciones, indexadas en el rango 0, …, número de aristas−1, representan a las aristas del grafo original. La presencia de una arista que forma parte del grafo original en un grafo representado queda determinada por el valor 1 en la posición correspondiente en el cromosoma.

La Figura 5.1 presenta a modo de ejemplo un grafo reducido y su codificación utilizando la representación binaria propuesta. En el grafo se han marcado con color oscuro los nodos terminales, etiquetados con los números 1, 2 y 4, mientras que los nodos de Steiner figuran con color claro y se han etiquetado por los números 3, 5 y 6. Las aristas presentes en el grafo solución se han marcado con líneas llenas y las aristas que forman parte del grafo original pero no están presentes en la solución representada se han marcado con líneas punteadas.



1	1	0	0	1	1	0	1	1	1	1
e_{12}	e_{13}	e_{14}	e_{16}	e_{23}	e_{25}	e_{34}	e_{36}	e_{45}	e_{46}	e_{56}

Figura 5.1 : Codificación binaria basada en aristas.

La principal ventaja de la representación binaria presentada consiste en que el diseño de los operadores evolutivos es directo, y su implementación muy sencilla, ya que los operadores genéricos de recombinación y mutación pueden ser utilizados sin necesidad de adaptarlos a la codificación. El principal inconveniente que incorpora este enfoque surge del hecho de que luego de aplicar un operador evolutivo tradicional, los individuos resultantes pueden no representar una solución factible del Problema de Steiner Generalizado. Como consecuencia, luego de cada recombinación o mutación, es necesario verificar la factibilidad de los nuevos individuos generados. En este trabajo se adoptó como criterio chequear la factibilidad de los individuos generados al aplicar los operadores evolutivos y descartar aquellos individuos no factibles. Este esquema simplifica la operativa del mecanismo evolutivo, evitando dos tareas complejas desde el punto de vista teórico: cuantificar cuán lejos de la factibilidad se encuentra cada solución no factible generada y determinar un modelo de penalización adecuado que permita asignar un valor de fitness a soluciones no factibles. Esta idea sigue la propuesta de Esbensen en sus trabajos sobre el problema del árbol de Steiner (Esbensen, 1993, 1994).

5.2.3. Codificaciones alternativas

Si bien en esta primera instancia se decidió, por las consideraciones de simplicidad expuestas, trabajar con la codificación binaria simple basada en aristas, otras propuestas de codificación fueron estudiadas en experimentos no formalizados y como parte de un proyecto de grado (Calegari, 2002). En particular se consideraron dos codificaciones, una basada en representar caminos entre nodos terminales y otra basada en representar grafos para cada requerimiento de conexión.

Listas de caminos

La codificación basada en listas de caminos representa cada solución con un cromosoma que constituye un arreglo de largo igual al número total de restricciones de caminos planteadas entre nodos terminales. Cada posición del cromosoma almacena una lista de caminos disjuntos entre los nodos terminales correspondientes a la restricción considerada. La información sobre los nodos origen y destino no se incluye en la lista de caminos ya que está implícita en la restricción. Solo se incluyen los identificadores de los nodos intermedios por los cuales pasa cada camino. En caso de incluir un camino de largo unitario (es decir, sin nodos intermedios entre los nodos terminales origen y destino), se utiliza una lista vacía para representarlo. Un ejemplo de representación basada en lista de caminos para el grafo de ejemplo ya presentado se ofrece en la Figura 5.2.

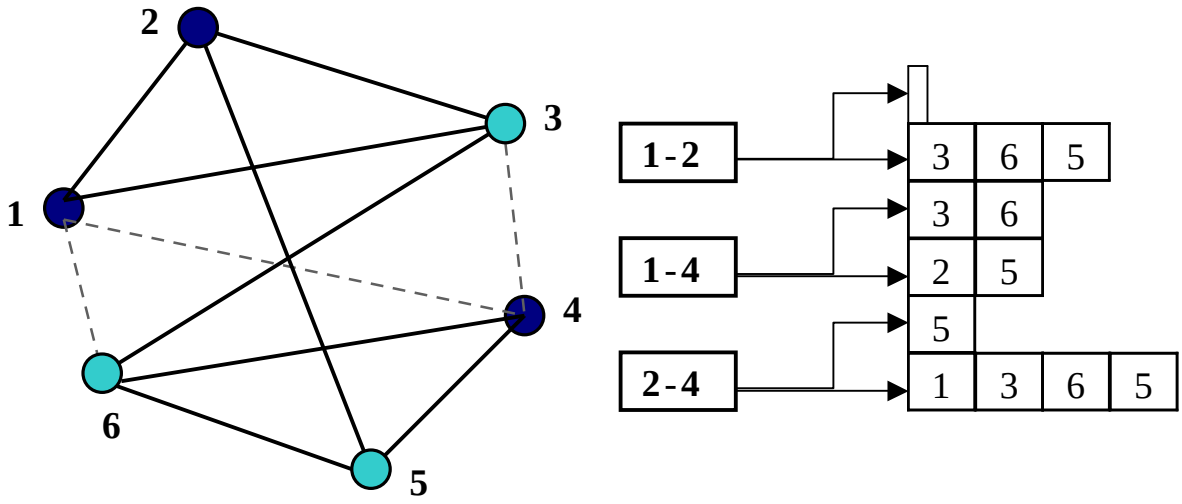


Figura 5.2 : Codificación basada en listas de caminos.

La representación basada en lista de caminos cuenta con una característica que la diferencia de la representación binaria simple basada en aristas: utiliza los *nodos* del grafo y no las aristas en la codificación, como lo hace la codificación binaria. Es posible utilizar este tipo de representación –similar a la utilizada frecuentemente para el problema del árbol de Steiner–, dado que el cromosoma tiene como índice los pares de nodos terminales sobre los que se plantean restricciones. Precisamente, una ventaja de esta representación consiste en que no se indican los nodos que no participan en la solución. Sin embargo, la representación agrega redundancia al replicar nodos utilizados en más de un camino; haciendo necesario el diseño de operadores evolutivos específicos y más complejos para considerar las características de la representación utilizada.

Una versión de representación basada en caminos pero utilizando aristas en lugar de nodos también fue estudiada en experimentos no formalizados. Esta representación tiene características similares a las presentadas en el párrafo anterior, siendo necesario adaptar los operadores evolutivos para manejar la redundancia en la representación. Esta complejidad motivó descartar su uso.

Subgrafos de requerimientos

La codificación basada en subgrafos de requerimientos es un híbrido entre las dos representaciones presentadas anteriormente. Guarda algunos puntos de contacto con la codificación de lista de caminos, ya que mantiene un arreglo de posiciones correspondientes a cada restricción de camino planteada entre nodos terminales, pero los caminos no se representan por separado para cada restricción, sino que se almacena el grafo solución para el requerimiento correspondiente. Cada grafo solución se representa mediante una codificación binaria simple basada en aristas.

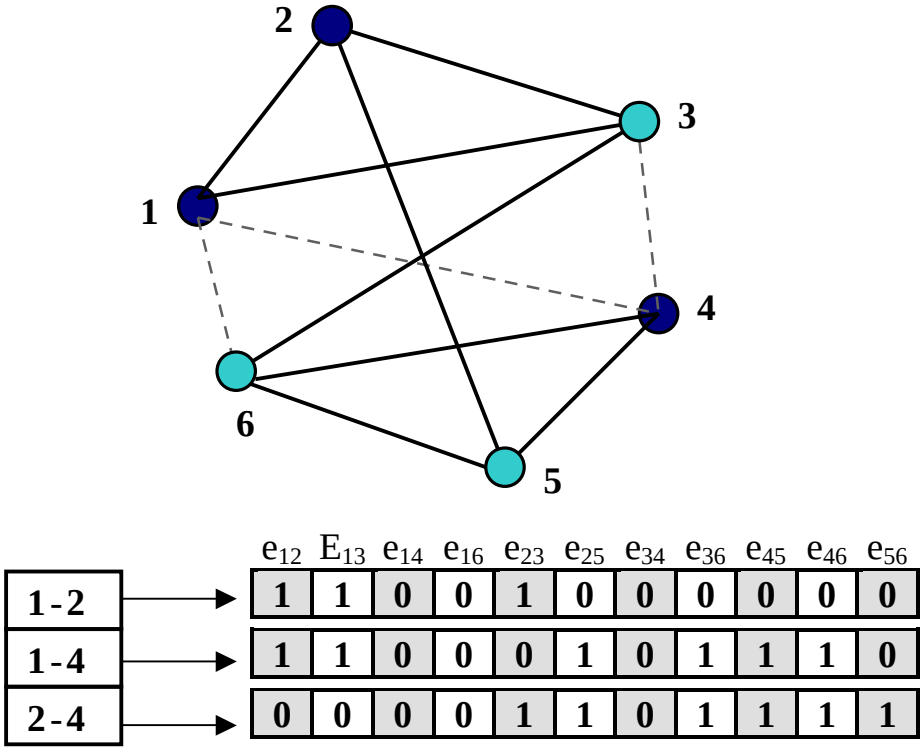


Figura 5.3: Codificación basada en subgrafos de requerimientos.

Estas dos codificaciones alternativas proponen el uso de genotipos más complejos que el de la codificación binaria simple. Como consecuencia, los operadores de recombinación y mutación deben ser especialmente diseñados para lidiar con la existencia de redundancia en la representación. Adicionalmente, los mecanismos de generación de la población inicial requieren de heurísticas para hallar caminos o soluciones factibles al problema. Como contrapartida a su complejidad debe mencionarse que ambas representaciones mantienen la factibilidad de individuos generados luego de la aplicación de los operadores evolutivos, obviando chequeos de consistencia y de verificación de los requisitos del problema, lo que redunda en una operativa más eficiente.

Como se mencionó anteriormente, el trabajo se realizó sobre la codificación binaria basada en aristas presentada en primer término. La utilización de las codificaciones alternativas se ha propuesto como trabajo futuro. Experimentos preliminares fueron realizados en proyectos de grado (Árraga y Aroztegui, 2001) (Calegari, 2002) para determinar si el uso de codificaciones más complejas y operadores específicos permitían obtener resultados de mejor calidad, pero no se lograron detectar mejoras significativas respecto a los obtenidos utilizando la codificación binaria simple. El resto del capítulo se concentra en presentar las características del algoritmo genético paralelo diseñado a partir de la codificación binaria basada en aristas.

5.3. Función de fitness

La función de fitness utilizada evalúa el costo total del diseño de la red de comunicaciones que se representa por una solución factible del Problema de Steiner Generalizado. Su formulación se presenta en la Ecuación 5.1. El problema de minimización del costo de la red se mapea a un problema de maximización sustrayendo el costo de la solución de una constante C_{ORIG}, que representa el valor de costo máximo para el grafo considerado, correspondiente al caso en que la totalidad del conjunto de aristas se incluye en la solución.

En la Ecuación 5.1, $|E|$ designa a la cardinalidad del conjunto de aristas, la función $C : N \rightarrow R$ retorna el costo de una arista y la función $EDGE: N \rightarrow [0,1]$ retorna el valor binario correspondiente a la presencia o ausencia de la arista en la posición i-ésima en la representación.

$$f = C_{ORIG} - \sum_{i=0}^{|E|-1} [EDGE(i) * C(i)]$$

Ecuación 5.1: función de fitness para el Problema de Steiner Generalizado.

Se decidió incorporar un mecanismo de escalado de los valores de fitness con el fin de evitar los conocidos problemas de convergencia prematura y de indiferencia en la diversidad. El problema de convergencia prematura se manifiesta cuando en etapas tempranas de la evolución un individuo muy adaptado domina ampliamente la población e impide la reproducción de individuos con diferentes características. Por su parte, la indiferencia en la diversidad se produce en etapas avanzadas de la evolución, en donde la competencia entre individuos con valores de fitness similares puede llevar al mecanismo evolutivo a un funcionamiento similar al de la caminata aleatoria entre los individuos de la población (Goldberg, 1989a).

Se implementó un escalado lineal del fitness, tomando en cuenta el promedio de fitness de la población f_{AVG}, el máximo valor de fitness f_{MAX}, y un factor de escalado k_{ESC}. Para el escalado es necesario definir una función F que cumpla las condiciones presentadas en la Ecuación 5.2.

$$F(f_{AVG}) = f_{AVG} \qquad (5.2.1)$$
$$F(f_{MAX}) = k.f_{AVG} \qquad (5.2.2)$$

Ecuación 5.2: Condiciones para la función de escalado del fitness.

El mecanismo de escalado tiende a separar el fitness de los individuos del valor medio f_{AVG}. La Ecuación 5.2.1 asegura que los individuos con valores promedios de fitness tengan al menos una copia en las nuevas generaciones. La ecuación 5.2.1 regula el fitness de los individuos con fitness máximo. El valor constante k corresponde al número de copias esperadas para el individuo con máximo fitness.

La función de escalado lineal se presenta en la Ecuación 5.3. Los valores de los coeficientes a y b se calculan a partir del valor definido de la constante k_{ESC} mediante un procedimiento de *preescalado* que se describe en Goldberg (1989a).

$$F = a.f + b$$

Ecuación 5.3: Función de escalado lineal.

5.4. Operadores

Al trabajar con la codificación binaria basada en aristas, es posible utilizar los operadores evolutivos de recombinación y mutación tradicionales diseñados para las representaciones binarias que figuran en múltiples referencias bibliográficas (Goldberg, 1989a) (Michalewicz, 1993) (Mitchell, 1996), sin necesidad de realizar modificaciones a su operativa para adaptarlos a una codificación específica.

Los operadores utilizados por el algoritmo genético diseñado corresponden a

- Selección proporcional al fitness escalado.
- Recombinación de un punto, seleccionado aleatoriamente en el intervalo de enteros $[0,|E|-1]$, que intercambia trozos del cromosoma.
- Mutación de inversión del valor de un alelo, seleccionado probabilísticamente.

La población se inicializa al azar, mediante un procedimiento que elimina aleatoriamente hasta un máximo de 5% de las aristas a partir de la representación del grafo original. Luego de la eliminación aleatoria, se aplica el chequeo de factibilidad sobre los individuos generados para descartar soluciones iniciales no factibles. Cada solución no factible detectada se elimina de la población inicial, y el procedimiento de inicialización se aplica nuevamente para generar otra solución. Este procedimiento aleatorio evita la aplicación de heurísticas específicas para hallar soluciones apropiadas del problema, continuando con la idea de utilizar estrategias simples para la resolución del problema.

5.4.1. Cálculo de factibilidad

Para determinar la factibilidad de un individuo luego de aplicados los operadores evolutivos, se utiliza un chequeo de dos etapas para verificar que el grafo resultante cumpla las restricciones de caminos impuestas para los nodos terminales del Problema de Steiner Generalizado.

Una sencilla heurística se utiliza en primera instancia para verificar los grados de los nodos terminales y descartar soluciones cuando el grado es menor que el máximo valor de número de caminos impuestos como restricción para ese nodo terminal. Esta primera etapa del chequeo tiene orden cuadrático en la cardinalidad del conjunto de nodos terminales.

Cuando los grados de todos los nodos terminales son compatibles con los requerimientos de conexión, se utiliza el algoritmo de Ford–Fulkerson (1962) para hallar los caminos entre cada par de nodos terminales, considerando uno de ellos como fuente y asignando al otro el rol de pozo. Como el algoritmo de Ford-Fulkerson se define para trabajar sobre grafos dirigidos, cada arista del grafo se considera como un par de aristas dirigidas de sentidos opuestos. Asumiendo la capacidad de las aristas unitaria, el flujo máximo entre fuente y pozo coincide con el número máximo de caminos disjuntos entre los nodos terminales considerados. Si el número máximo de caminos disjuntos es menor que el valor del requerimiento correspondiente para el par de nodos, el grafo es no factible. La variante de Ford-Fulkerson empleada considera una función de capacidad de flujo unitaria (integral) para cada arista, y tiene orden $O\ (\ |E|\ .\ r_{MAX}\ .\ n_T^{\,2}\ )$ siendo $|E|$ el conjunto de aristas, y r_{MAX} el valor máximo del conjunto de requerimientos, de acuerdo a Baratz (1980).

5.5. Generación de números pseudoaleatorios

La naturaleza no determinística de los algoritmos evolutivos implica la necesidad de disponer de un mecanismo para la generación de números pseudoaleatorios para utilizar en los diferentes operadores probabilísticos que determinan el modelo evolutivo. En el resto del trabajo se designará a estos números genéricamente como *números aleatorios*, pese a la naturaleza determinística de los métodos utilizados para su generación, en un evidente abuso del lenguaje.

La bibliografía de referencia habitualmente destaca la importancia de la generación de secuencias de números aleatorios para utilizar en los operadores evolutivos. Sin embargo, en la descripción de implementaciones particulares de bibliotecas y de algoritmos evolutivos específicos, rara vez se profundiza en este aspecto, utilizándose las funciones habituales para generación de números aleatorios que proveen los lenguajes de programación estándar.

En este trabajo se decidió implementar un método de generación de números aleatorios basado en una fórmula de congruencia lineal, tomando en cuenta que constituyen una familia de métodos ampliamente utilizados, de formulación sencilla y que permiten adaptar las características de la secuencia generada a través de sus parámetros. Este hecho es importante y debe tenerse en cuenta al realizar experimentos complejos que requieren utilizar una gran cantidad de números aleatorios, ya que modificando los parámetros es posible, de ser necesario, aumentar el período de la secuencia seudoaleatoria generada. La Ecuación 5.4 presenta la formulación recursiva genérica para la generación de una secuencia de números aleatorios por medio de un método congruencial lineal, tal como se presenta en Knuth (1981). Los parámetros del método son los números enteros constantes A –el *multiplicador*–, C –el *incremento*–, M –el *módulo*– y S –la *semilla* que determina el valor inicial. Los valores de estos parámetros deben ser elegidos de acuerdo a criterios especificados para producir una secuencia adecuada de números aleatorios.

La Ecuación 5.4.2 suele abreviarse como $X_{n+1} = MCG(A, X_n, C, M)$, denotando la sigla *MCG* al operador que define a la congruencia lineal utilizada.

$$X_0 = S \qquad (5.4.1)$$

$$X_{n+1} = (A.X_n + C) \bmod M \qquad (5.4.2)$$

Ecuación 5.4: Método de congruencia lineal para la generación de números aleatorios.

El módulo M es también llamado *período* de la secuencia, ya que de cumplirse ciertas condiciones entre los valores de los parámetros seleccionados, la secuencia generada por un método de congruencia lineal como el presentado recién comenzará a repetirse luego de haber generado M números aleatorios.

Luego de estudiar las características de varios métodos se decidió emplear un algoritmo que utiliza *dos* secuencias de números aleatorios (Knuth, 1981), tal como se presenta a continuación.

El algoritmo de generación de números aleatorios implementado utiliza dos secuencias X e Y generadas por dos métodos congruencias lineales con diferentes valores de sus parámetros. Uno de estos métodos (en el caso del algoritmo presentado, la secuencia X) es utilizado para asignar valores a un arreglo de números reales V, mientras que el otro método es utilizado para determinar aleatoriamente un índice del arreglo V desde donde se seleccionará un número aleatorio ya generado. El contenido de la posición seleccionada en el arreglo será modificado posteriormente a la elección del número aleatorio. Un esquema del algoritmo implementado para obtener el siguiente número de la secuencia aleatoria se describe en la Figura 5.4.

```
Y(n+1) = MCG(A2, Y(n), C2, M2)          (5.4.3)
indice = LARGO_BUFFER * Y(n+1) / M2     (5.4.4)
resultado = V[indice]                   (5.4.5)
X(n+1) = MCG(A1, X(n), C1, M1)          (5.4.6)
V[indice] = X(n+1)                      (5.4.7)
```

Figura 5.4: Algoritmo para la generación de números aleatorios.

La Ecuación 5.4.3 define recursivamente a la secuencia aleatoria *Y*, utilizando un método de congruencia lineal. En la Ecuación 5.4.4 se detemina el índice para seleccionar un elemento en el arreglo *V*, tomando en cuenta su tamaño, indicado por el entero constante LARGO_BUFFER. En la asignación 5.4.5 se extrae el valor seleccionado de la posición determinada. La Ecuación 5.4.6 genera un valor sustituto para el número seleccionado, utilizando el segundo método de congruencia lineal. Por último, en la asignación 5.4.7 se actualiza el valor del arreglo *V* en la posición del índice.

Knuth (1981) prueba que el período de un generador combinado es considerablemente grande si los períodos M_1 y M_2 de las dos congruencias lineales utilizadas son números primos entre sí.

El procedimiento descrito anteriormente se utiliza para generar números aleatorios reales uniformemente distribuidos en el intervalo [0,1]. Utilizando el algoritmo explicado como base, se diseñaron rutinas complementarias para generar números aleatorios enteros y números aleatorios enteros en un rango especificado, escalando los valores apropiadamente.

En la implementación presentada, cada proceso del algoritmo paralelo cuenta con su propio generador de números aleatorios mediante un método congruencial de dos secuencias. Para que cada población trabaje con diferentes secuencias se deben especificar diferentes valores para los parámetros A_1, A_2, C_1, C_2, M_1 y M_2 y para los valores iniciales X_{01}, X_{02}, Y_{01} y Y_{02} en el archivo de configuración del sistema (las propiedades que deben cumplir estos parámetros y un conjunto de valores de ejemplo se presentan en el propio archivo). Cada proceso aceptará como entrada únicamente aquellos parámetros especificados para su *rango* o identificador de proceso.

Se destaca que la elección del método de generación de números aleatorios no estuvo motivada por sugerencias específicas, sino que se trató de una elección arbitraria, escogiendo un método robusto para potenciar el algoritmo genético y que a la vez fuera sencillo de implementar. No se realizaron experimentos para garantizar las propiedades de la implementación realizada, ni su comparación con las funciones de los lenguajes de programación estándar, ni tampoco para determinar la conveniencia del mecanismo de parametrización utilizado sobre otros métodos usuales como variar solamente el valor de la semilla.

5.6. Modelo de paralelismo

El algoritmo genético paralelo diseñado corresponde a un modelo de población distribuida, organizada en subpoblaciones semi independientes que evolucionan en paralelo. De acuerdo a la taxonomía de Nowostawski y Poli (1999) se clasifica dentro de la categoría de modelos distribuidos de subpoblaciones con migración.

El modelo de paralelismo implementado divide la población en varias subpoblaciones –denominadas *islas* o *demes*– que evolucionan asincrónicamente en forma semi-independiente, interactuando esporádicamente. Cada cierto número de generaciones, cada subpoblación selecciona un conjunto de individuos para migrar a las subpoblaciones vecinas. La topología de migración entre poblaciones es estática durante la ejecución del algoritmo, pero el usuario tiene la posibilidad de definirla de antemano en el archivo de configuración del sistema.

La Figura 5.5 ofrece el seudocódigo del algoritmo genético paralelo diseñado.

```
Generar población inicial pop con sizepop cromosomas.
Mientras (no se cumple criterio de parada)
    Escalar fitness de población.
    Crear nueva población vacía newpop.
    Mientras (tamaño de newpop < sizepop)
        Seleccionar dos individuos padre de pop.
        Cruzar y mutar, se obtienen hijo1 e hijo2.
        Si hijo1 es factible
            Insertar hijo1 en newpop.
        Si hijo2 es factible
            Insertar hijo2 en newpop.
    fin
    Si (condición migración)
        Migración: Recibir y enviar inmigrantes.
        Insertar inmigrantes en newpop.
    fin
    pop = newpop
fin
Retornar mejor solución hallada
```

Figura 5.5: Seudocódigo del algoritmo genético paralelo.

El sistema se compone de varios procesos, cada uno de los cuales manipula de modo independiente una subpoblación del algoritmo genético paralelo. Un proceso distinguido actúa de controlador del sistema, manteniendo la información de la mejor solución encontrada hasta el momento, su valor de fitness y otros valores útiles para reportes y estadísticas en general.

El operador de migración utiliza selección proporcional para elegir los individuos emigrantes y selección proporcional inversa para determinar los individuos reemplazados en las poblaciones destino, seleccionando con mayor probabilidad individuos de peor fitness. Este modelo constituye una variante probabilística del mecanismo elitista descrito por Cantú-Paz (2000) donde los mejores individuos de cada población emigran y reemplazan a los peores individuos en la población destino.

La comunicación de individuos emigrantes se realiza de forma asincrónica, mediante envío y recepción de mensajes no bloqueantes. Esta característica permite a cada subpoblación continuar su procesamiento sin necesidad de confirmación del arribo de sus mensajes a destino, ni de bloquearse aguardando por mensajes de otras subpoblaciones.

Se implementaron varios criterios de parada independientes para finalizar la ejecución del algoritmo. Es posible detener el algoritmo por un criterio de esfuerzo prefijado, cuando el proceso controlador alcanza un límite de generaciones o de tiempo de ejecución (en segundos) que se especifica en la configuración. La alternativa consiste en detener el algoritmo por convergencia, utilizando el criterio de que 3/4 de las subpoblaciones utilizadas cumplan que la relación entre fitness promedio en la generación actual y en la generación anterior sea menor que un valor indicado en la configuración. Cuando el algoritmo genético paralelo finaliza su ejecución, se presenta la mejor solución encontrada hasta el momento.

5.7. Implementación y plataforma de ejecución

Para la implementación del algoritmo genético paralelo se utilizó el lenguaje C++, con el compilador GNU G++ y la implementación MPICH de la biblioteca de desarrollo de programas paralelos y distribuidos MPI (MPI Forum, 2003). La ejecución distribuida se realizó sobre un cluster de cuatro equipos Intel monoprocesador Pentium III 400 MHz, con sistema operativo SuSE Linux 8.0 conectados mediante una LAN Ethernet a 100 Mbps.

5.8. Resultados Obtenidos

Esta sección presenta los resultados obtenidos en los experimentos de validación, configuración y evaluación del algoritmo genético paralelo propuesto sobre un conjunto de problemas de prueba diseñado específicamente con ese fin.

5.8.1. Instancias de prueba para el Problema de Steiner Generalizado

Como fue mencionado en el Capítulo 4, prácticamente no existen referencias bibliográficas sobre la aplicación de técnicas heurísticas al Problema de Steiner Generalizado. Como consecuencia, no se han desarrollado conjuntos estandarizados de instancias de prueba para evaluar comparativamente los resultados de diferentes algoritmos o métodos de resolución del problema. En general, los investigadores que han abordado el problema generalizado han trabajado con instancias de prueba derivadas de los casos más simples, como el caso del problema del árbol de Steiner, o han generado aleatoriamente sus propios casos de prueba.

En el contexto de este trabajo no fue posible recolectar información sobre configuraciones de redes de comunicaciones existentes ni relevar características de diseños que puedan tener aplicación práctica en escenarios reales. Asimismo, la tarea de extender los conjuntos existentes de instancias de prueba para las versiones simples de la clase de problemas de Steiner no es sencilla, requiriéndose un estudio teórico de las propiedades del problema y las características de sus soluciones. Para las pruebas de configuración y validación del algoritmo genético diseñado en este trabajo se utilizó el conjunto de grafos presentados por Robledo (2001). Sin embargo, la reducida complejidad de estos casos de prueba no imponían un reto al algoritmo genético, por lo cual se contempló diseñar un conjunto de problemas más complejos para evaluar los resultados y el desempeño del algoritmo genético paralelo diseñado.

Tomando en cuenta las anteriores observaciones, se optó por diseñar un conjunto de instancias de prueba aleatorias, compuesto de tres problemas, para la evaluación de calidad de resultados y desempeño computacional del algoritmo genético paralelo diseñado. El estudio teórico del diseño de un conjunto de problemas de prueba se propone como una línea inmediata de trabajo futuro.

Los grafos del conjunto de prueba creado se designan por un nombre que referencia a la cantidad total de nodos y el número de nodos terminales. Por ejemplo, *grafo 50-15* designa al grafo más pequeño del conjunto de problemas de prueba, que tiene 50 nodos de los cuales 15 son terminales, y así sucesivamente para el resto de las instancias. Las tres instancias diseñadas para evaluar el algoritmo genético fueron generadas seleccionando aleatoriamente topologías de grafos, considerado los nodos como puntos geográficos en un plano euclídeo. Los costos asociados a los enlaces fueron considerados proporcionales a las distancias euclideanas entre nodos para las instancias *grafo 75-25* y *grafo 50-15*, y fueron seleccionados aleatoriamente entre los valores enteros en el intervalo [1, …, 20] para la instancia *grafo 100-10*. Los requerimientos de conexión para cada par de nodos terminales fueron seleccionados aleatoriamente de modo uniforme entre los valores enteros en el intervalo [0, …, 4] para las tres instancias consideradas.

Los detalles de las tres instancias de prueba diseñadas para el Problema de Steiner Generalizado se presentan en la tabla de la Figura 5.6, indicando el número total de nodos, número de nodos terminales, número de aristas y el grado de conectividad promedio (cociente entre el número de aristas presentes en el grafo sobre el número de aristas del grafo completo, expresado en aristas por nodo). Se incluyen además los valores de costo total (C_{ORIG}) para cada grafo, correspondiente al caso en que la totalidad del conjunto de aristas del grafo original se incluye en la solución. Este valor es útil como referencia para comparar los valores de fitness alcanzados por el algoritmo.

	grafo 100-10	*grafo 75-25*	*grafo 50-15*
Nodos	100	75	50
Terminales	10	25	15
Aristas	500	360	249
Grado Conectividad Promedio	0.1	0.13	0.2
Costo original (C_{ORIG})	4925.00	6294.93	10949.98

Figura 5.6: Características de los grafos de prueba para el Problema de Steiner Generalizado.

Los problemas diseñados pueden considerarse "representativos" para redes de comunicaciones de mediano tamaño con requisitos de conexión variables. El formato utilizado en la notación de los grafos de prueba se presenta en el Anexo II de este documento.

El conjunto de problemas de prueba diseñado, conjuntamente con una descripción del formato utilizado en la notación y el programa generador aleatorio de grafos utilizado se encuentran disponibles públicamente en la URL del Grupo de Trabajo en Procesamiento Paralelo y sus Aplicaciones del Centro de Cálculo, Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Uruguay: http://www.fing.edu.uy/inco/grupos/cecal/hpc/gsp.

Se destaca que con el fin de evitar sesgos en los resultados se utilizaron dos conjuntos de problemas independientes, uno en la etapa de configuración (grafos de tamaño reducido) y otro en la etapa de evaluación de resultados del algoritmo genético (grafos de tamaño mediano).

5.8.2. Configuración y validación del algoritmo genético

La configuración de los valores de tamaño de población y probabilidades de recombinación y mutación se realizó utilizando la versión panmíctica serial del algoritmo genético para encontrar soluciones del Problema de Steiner Generalizado sobre un conjunto de instancias de complejidad reducida, obtenidas del trabajo de Robledo (2001).

Para configurar los parámetros probabilísticos del algoritmo, se consideraron tres valores candidatos para la probabilidad de mutación (0.001, 0.005, y 0.01) y tres valores candidatos para la probabilidad de recombinación (0.7, 0.8, y 0.9). Para cada una de las 9 combinaciones posibles, se realizaron 5 ejecuciones independientes del algoritmo genético sobre cada uno de los 7 casos de prueba considerados, utilizando un criterio de parada de esfuerzo prefijado en 200 generaciones (determinado en experimentos preliminares previos como adecuado para alcanzar soluciones de calidad razonable para las instancias abordadas). Se analizó la calidad de las soluciones obtenidas, evaluando el fitness promedio y los mejores valores obtenidos y el desempeño del algoritmo, evaluando el número promedio de generaciones necesarias para encontrar la mejor solución hallada y el tiempo medio de ejecución en segundos.

Los resultados no permitieron sacar conclusiones definitivas, ya que en promedio todas las configuraciones obtuvieron soluciones finales de calidad similar. Sin embargo, se observa que al aumentar los valores de las probabilidades de mutación y recombinación disminuye el número promedio de generaciones necesarias para hallar el mejor individuo. Como contrapartida, se incrementa el tiempo total de ejecución del algoritmo genético, a consecuencia del mayor cómputo requerido. Tomando en cuenta que el aumento de los tiempos de ejecución no es significativo, se decidió utilizar los mayores valores candidatos para probabilidades de recombinación y mutación considerados (0.9 y 0.01 respectivamente).

Para determinar un tamaño de población adecuado se evaluaron las soluciones obtenidas y la eficiencia al utilizar poblaciones de 30 y 50 individuos para 5 ejecuciones independientes del AG sobre los casos de configuración. Se utilizaron los mejores valores determinados en la etapa anterior para las probabilidades de recombinación y mutación, 0.9 y 0.01 respectivamente y el criterio de parada de esfuerzo prefijado en 200 generaciones. No se detectaron mejoras significativas en la calidad de resultados obtenidos al aumentar el tamaño de la población, y tomando en cuenta el incremento en el costo computacional demandado, que se amplificará considerablemente al resolver problemas más complejos, se decidió trabajar con poblaciones de 30 individuos.

No se realizó un estudio de configuración para el factor de escalado de fitness, utilizándose el valor $k_{ESC} = 2$ sugerido en el texto de Goldberg (1989a).

Conjuntamente con la etapa de configuración, se realizaron experimentos de validación de los resultados obtenidos, tomando en cuenta que se resolvían casos para los cuales se disponía de soluciones calculadas previamente mediante otra técnica metaheurística. Este hecho posibilitó la comparación de las soluciones obtenidas por el algoritmo genético con las presentadas en el trabajo de Robledo (2001). Los resultados de los experimentos de validación que se reportan a continuación fueron obtenidos con los mejores valores para los parámetros determinados en los experimentos de configuración, que se presentan en la tabla de la Figura 5.7.

Parámetro	***Valor***
Número de procesos	1
Condición de parada	200 generaciones.
Tamaño de la población	30 individuos.
Probabilidad de mutación	0.01
Probabilidad de recombinación	0.9
Factor de escalado de fitness	2

Figura 5.7: Configuración del algoritmo genético utilizada en los experimentos de validación

Los mejores resultados obtenidos en cada caso considerado en los experimentos de validación se resumen en la tabla de la Figura 5.8. Se presentan las características de los grafos utilizados (número total de nodos, número de aristas y número de nodos terminales), el mejor resultado conocido previamente, el resultado obtenido por el algoritmo genético y el valor óptimo calculado mediante un motor de backtracking diseñado específicamente para la resolución del problema. Una descripción de las topologías de los grafos del conjunto de instancias de prueba utilizado en los experimentos de validación se presenta en el Anexo II.

Grafo	*Nodos*	*Aristas*	*Terminales*	*Mejor solución (Robledo, 2001)*	*Mejor solución algoritmo genético*	*Solución óptima*
grafo_v1	5	9	2	5	5	5
grafo_v2	6	9	3	7	7	7
grafo_v3	6	14	6	9	9	9
grafo_v4	6	12	3	9	9	9
grafo_v5	5	16	4	22	18	18
grafo_v6	10	15	6	20	20	20
grafo_v7	16	25	6	41	39	39

Figura 5.8: Resultados de los experimentos de validación..

Las soluciones encontradas por el algoritmo genético coincidieron con las soluciones óptimas del problema para todos los casos reducidos considerados en los experimentos de validación. Para dos de los problemas el algoritmo genético encontró mejores soluciones que las publicadas por Robledo (2001), obtenidas mediante la metaheurística Ant Systems.

5.8.3. Configuración del modelo de migración

Para obtener una buena configuración del modelo paralelo, se investigó la sensibilidad de los resultados en función de los parámetros que caracterizan al modelo de migración. El objetivo de los experimentos de configuración consistió en comprender los efectos de la migración en la intensidad de selección de individuos y sus implicancias en el mecanismo evolutivo.

Fueron analizados los principales parámetros que caracterizan al modelo de migración, a saber:

- Topología de migración: modelada como un grafo dirigido cuyos nodos representan a las subpoblaciones y sus aristas indican la migración de individuos desde subpoblación origen a subpoblación destino.
- Frecuencia de migración: asiduidad con que se aplica el operador de migración.
- Tasa de migración: cantidad de individuos de la población local que emigra.

Tomando en cuenta la limitación del equipamiento disponible, se decidió trabajar con un número fijo de subpoblaciones (4 demes, uno por cada equipo disponible). Respecto a la política de selección y reemplazo, que determina cuáles individuos son elegidos para migrar y cuáles se reemplazan en la población destino, se trabajó con la estrategia elitista probabilística para la migración comentada en la Sección 5.5, que provee un aumento en la presión de selección. El resto de los parámetros fueron analizados en las pruebas de configuración del modelo de migración, utilizando como valores candidatos los que se presentan en la tabla de la Figura 5.9.

Parámetro	*Valores candidatos*
Frecuencia de migración	1,2,4,6,8,10
Tasa de migración	0.01,0.05,0.1,0.2,0.3,0.4,0.5
Topología de migración	grafo completo, anillo, anillo+1

Figura 5.9: Valores candidatos para la configuración del modelo de migración.

Los valores candidatos intentaron abarcar una gama representativa de configuraciones del modelo de migración del algoritmo genético paralelo. De todos modos, el estudio se realizó sobre la base del conocimiento previo de que la escasa disponibilidad de recursos computacionales impediría un análisis exhaustivo de las topologías de migración consideradas, cuyo esquema se presenta en la Figura 5.10.

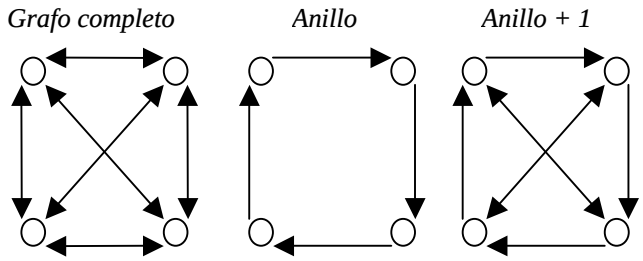


Figura 5.10: Topologías de migración consideradas.

El estudio de los parámetros del modelo de migración involucró experimentos sobre el conjunto de tres instancias de prueba diseñado aleatoriamente, ya que los resultados del algoritmo secuencial presentados en la tabla de la Figura 5.8 indicaron que los grafos reducidos no plantean una dificultad suficiente que permita evaluar comparativamente los resultados del algoritmo paralelo. Se asumieron valores constantes, que se presentan en la tabla de la Figura 5.11, para aquellos parámetros del algoritmo genético que no caracterizan al modelo de migración. Se utilizaron los mejores valores hallados en los experimentos de configuración, los mismos que fueron utilizados también en los experimentos de validación. Solamente se modificó el número de generaciones utilizado como criterio de parada, que se incrementó hasta 500 generaciones tomando en cuenta el incremento en la dimensión de los casos de prueba.

La utilización de un criterio de parada de esfuerzo prefijado (500 generaciones) fue motivada por el objetivo de concentrarse exclusivamente en la influencia de los parámetros del modelo de migración en la calidad de los resultados obtenidos. Fue posible comprobar a posteriori que el algoritmo genético paralelo era capaz de mejorar la calidad de resultados si se especificaba un tope mayor para el número de generaciones, pero este hecho no invalida la correctitud del análisis realizado, orientado a obtener una configuración adecuada del modelo de migración. Por otra parte, utilizar un valor prefijado de esfuerzo permite realizar una comparación justa de calidad de resultados a los efectos de determinar una buena configuración para el modelo de migración. Por este motivo, no se utilizó un criterio de parada basado en la convergencia de las subpoblaciones.

La eficiencia de los modelos se evaluó secundariamente en estos experimentos, tan solo para decidir entre configuraciones que ofrecieran similar calidad en los resultados obtenidos.

Parámetro	***Valor***
Número de demes	4
Condición de parada	500 generaciones.
Tamaño de la población en cada deme	30 individuos.
Probabilidad de mutación	0.01
Probabilidad de recombinación	0.9
Factor de escalado de fitness	2

Figura 5.11: Constantes utilizadas en la configuración del algoritmo genético paralelo.

Los resultados de los experimentos de configuración de los parámetros del modelo de paralelismo y sus conclusiones se presentan a continuación.

5.8.3.1. Topología de migración

Los experimentos se realizaron tomando en cuenta las tres topologías de migración propuestas, sin apreciarse diferencias significativas en la calidad de los resultados finales al utilizar diferentes topologías. Este resultado es coherente con las especulaciones a priori, formuladas sobre la base de que el escaso número de procesadores disponibles hace a las topologías muy similares entre sí, sin permitir a una de ellas destacarse sobre las restantes.

5.8.3.2. Frecuencia de migración

El operador de migración introduce una complejidad adicional en la operativa del algoritmo genético paralelo que puede afectar su eficiencia computacional. Por este motivo es pertinente considerar los efectos de retrasar la migración sobre la calidad de resultados obtenidos. La Figura 5.12 resume el análisis de los efectos del uso de diferentes intervalos de migración sobre la calidad del resultado obtenido para un caso representativo sobre el *grafo 100-10*, utilizando los cinco valores de tasa de migración considerados en el estudio.

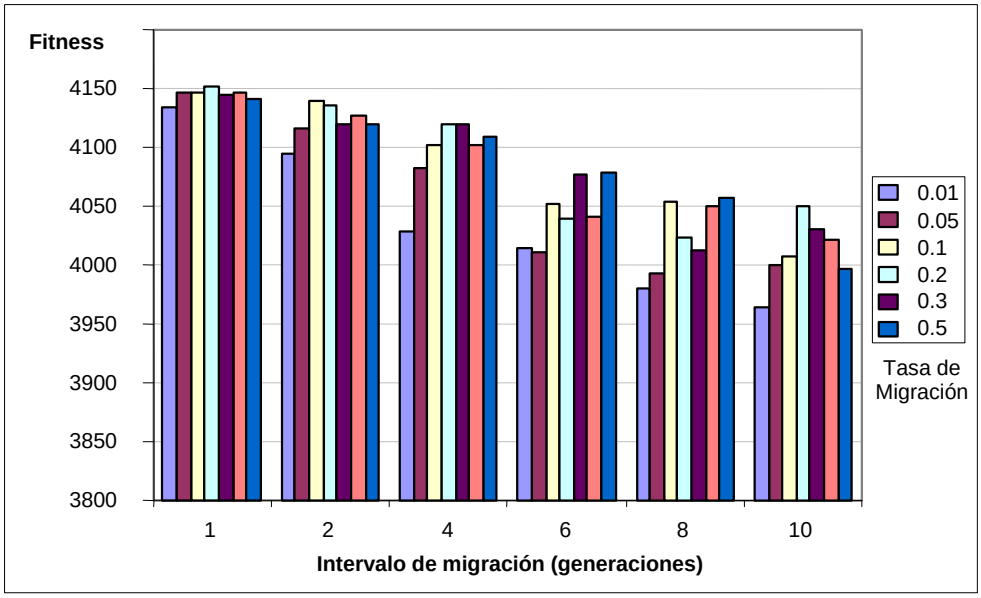


Figura 5.12: Caso representativo en el estudio de la frecuencia de migración.

Para todos los valores de tasa de migración considerados los mejores resultados ocurren con intervalo de migración igual a uno, que corresponde al caso de mayor frecuencia de comunicaciones. Al no detectarse significativas mejoras en la eficiencia computacional al retrasar las migraciones, se adoptó el criterio de utilizar esta cota superior para la frecuencia de migración, conduciendo a un modelo cercano al panmíctico de población distribuida (Cantú-Paz, 2000).

5.8.3.3. Tasa de migración

El número de individuos participantes en la migración influye sobre la presión de selección del algoritmo y por ello sobre la calidad de los resultados. Un modelo con un número bajo de individuos migrados tiene muchos puntos de contacto con el modelo panmíctico. Por otra parte, la comunicación de un número considerable de individuos entre subpoblaciones puede afectar notoriamente la eficiencia computacional del algoritmo genético paralelo. Por estos motivos es pertinente estudiar la influencia del número de individuos emigrantes en la calidad de las soluciones obtenidas por el algoritmo.

El gráfico de la Figura 5.13 presenta los resultados obtenidos en función de la tasa de migración utilizada considerando los diversos valores para intervalos de migración para un caso representativo sobre la instancia de prueba *grafo 100-10*. En el gráfico puede apreciarse que el análisis no refleja diferencias significativas en la calidad de los resultados finales dependiendo de la tasa de migración utilizada. De acuerdo a la argumentación de Tanese (1989) en uno de los primeros estudios paramétricos sobre algoritmos genéticos paralelos, tasas *moderadas* de migración son necesarias para que un modelo paralelo pueda alcanzar y eventualmente superar la calidad de resultados de un algoritmo genético panmíctico. Tanese sugiere un valor de 0.2 como "aceptable" para alcanzar resultados de calidad similar a un modelo panmícitico.

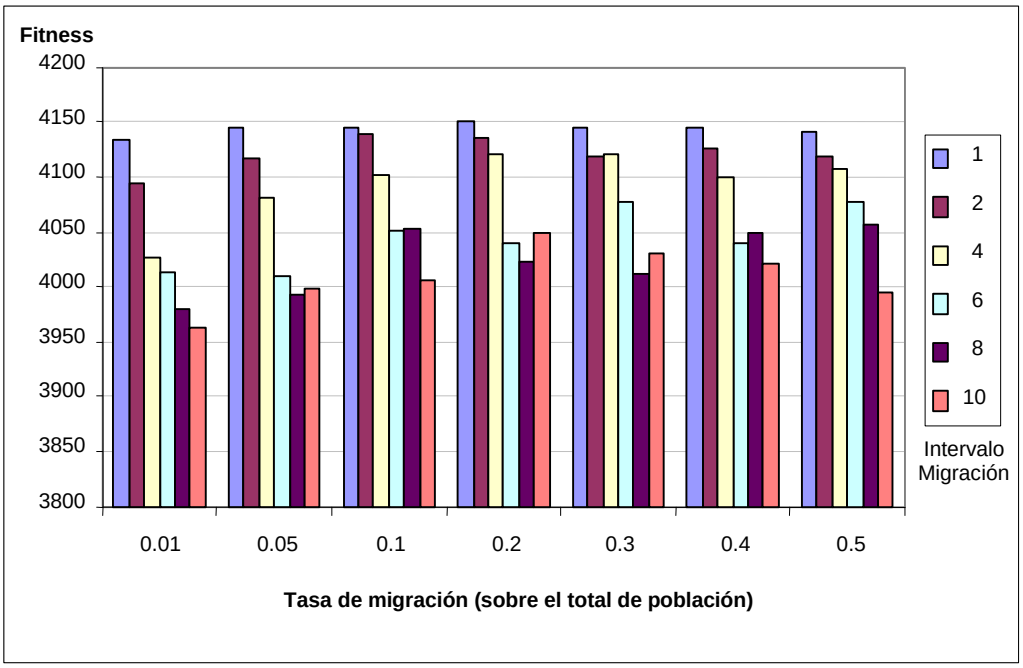


Figura 5.13: Caso representativo en el estudio de la tasa de migración.

5.8.3.4. Conclusiones sobre la calibración

En función de cada parámetro estudiado, es posible extraer algunas conclusiones respecto a la configuración del algoritmo genético paralelo:

- Topología de migración: La escasa disponibilidad de equipos no permite extraer resultados concluyentes sobre la topología de migración óptima. En los experimentos realizados no fue posible apreciar diferencias significativas en la calidad de los resultados obtenidos, ya que al disponer de solamente cuatro elementos de procesamiento, las topologías son muy similares entre sí. Siguiendo la idea de Cantú-Paz (2000) quien indica que "aunque la topología de conexión completa no integra eficientemente a varios demes, su configuración óptima es una buena elección para reducir el tiempo de ejecución", se decidió utilizar la topología completa como configuración del modelo de migración del algoritmo genético paralelo, aún teniendo en cuenta que es la configuración menos escalable de las consideradas. Esta decisión debe ser reconsiderada en el caso de abordar problemas más complejos con la disponibilidad de mayor número de recursos computacionales.
- Frecuencia de migración: Los mejores resultados se obtuvieron consistentemente con frecuencia máxima. La calidad de los resultados empeora al retrasar la migración y no se aprecian deterioros significativos en la eficiencia computacional al aumentar la frecuencia con la que se aplica el operador de migración. Se optó por utilizar un intervalo de migración de una generación, lo cual implica que existirá comunicación para el intercambio de individuos entre subpoblaciones en cada una de las generaciones del algoritmo genético paralelo.
- Tasa de migración: La cantidad de individuos migrados no afectó significativamente la calidad de los resultados obtenidos en los experimentos realizados. Se optó por utilizar un valor intermedio de 0.3 para la tasa de migración, considerado como un valor de equilibrio entre un modelo fuertemente distribuido y un modelo de población panmíctica (Cantú-Paz, 1999).

5.8.4. Comparación entre modelo paralelo y panmíctico.

La comparación entre el modelo paralelo y el modelo panmíctico serial toma en cuenta la calidad de los resultados obtenidos sobre cada una de las tres instancias del conjunto de prueba, y la eficiencia del modelo, cuantificando los tiempos de ejecución y ejemplificando un caso representativo de evolución del fitness con relación al tiempo de procesamiento.

Los valores de los parámetros utilizados en los experimentos de comparación se presentan en la tabla de la Figura 5.14. Corresponden a los valores determinados como óptimos en los experimentos de configuración comentados anteriormente. Se utilizó un criterio de parada de esfuerzo prefijado de 2000 generaciones. Este valor alto para el número tope de generaciones fue fijado considerando la complejidad de los problemas del conjunto de prueba y tratando de dar posibilidades al algoritmo genético de encontrar los mejores resultados posibles para cada instancia.

Modelo	Panmíctico	Paralelo
Número de demes	1	4
Tamaño de la población	120	30
Tasa de migración	No utilizada.	0.3
Frecuencia de migración	No utilizada.	1
Topología	No utilizada.	Completa
Parámetros con iguales valores en ambos modelos		
Condición de parada	2000 generaciones	
Probabilidad de mutación	0.01	
Probabilidad de recombinación	0.9	
Factor de escalado de fitness	2	

Figura 5.14: Configuración para la comparación entre modelo paralelo y panmíctico.

Los resultados obtenidos por el algoritmo panmíctico y el algoritmo paralelo se resumen en la tabla de la Figura 5.15, indicando valores de fitness promedio, mejores valores de fitness y desviación estándar obtenidos en 30 ejecuciones independientes realizadas sobre cada una de las tres instancias del conjunto de prueba considerado.

Fitness	***Panmíctico***	***paralelo***
Grafo 100-10		
Promedio	4431.5	4472.5
Mejor	4529.0	4561.0
Desviación Estándar	46.7	50.2
Grafo 75-25		
Promedio	5298.9	5305.4
Mejor	5379.3	5406.7
Desviación Estándar	49.8	51.2
Grafo 50-15		
Promedio	9301.3	9304.6
Mejor	9354.3	9373.0
Desviación Estándar	70.7	85.3

Figura 5.15: Resultados comparativos entre modelo panmíctico y paralelo.

Analizando la calidad de los resultados obtenidos, no se aprecia una superioridad en la calidad de los resultados finales del modelo paralelo sobre el panmíctico. El modelo paralelo logra puntualmente mejores valores de fitness que el panmíctico serial, pero los valores promedio sobre las 30 ejecuciones realizadas solo muestran una diferencia notoria para la instancia *grafo 100-10.* , De todos modos, ninguna de las mejoras obtenidas en los valores promedio puede considerarse significativa estadísticamente. Este hecho puede explicarse tomando en cuenta la alta precisión de los resultados obtenidos por el algoritmo panmíctico serial, al haberse utilizado como criterio de parada un número elevado de generaciones.

Respecto al estudio de la eficiencia computacional, la tabla de la Figura 5.16 resume los tiempos promedio de ejecución (en segundos) de ambos modelos sobre las tres instancias de prueba consideradas y los valores obtenidos de *speedup* y de *eficiencia* al utilizar cuatro procesadores. Puede apreciarse que los elevados tiempos de ejecución del algoritmo panmíctico serial son reducidos notoriamente por el modelo paralelo al trabajar con cuatro procesadores. Si bien se manifiesta un comportamiento de speedup sublineal, el valor de eficiencia es muy cercano a uno, lo cual demuestra la capacidad del modelo paralelo para utilizar los recursos computacionales disponibles.

Instancia	*Panmíctico*	*Paralelo*	*Speedup*	*Eficiencia*
grafo 100-10	6424.3	1745.3	3.68	0.92
grafo 75-25	14340.1	3988.6	3.59	0.90
grafo 50-15	5846.6	1653.2	3.53	0.88

Figura 5.16: Comparación tiempos de ejecución entre modelo panmíctico y paralelo.

El gráfico de la Figura 5.17 muestra la evolución de los valores promedio de la función de fitness para ambos modelos comparados, respecto al tiempo de ejecución, en una ejecución representativa de la instancia de prueba *grafo 100-10*. Puede observarse el patrón característico de evolución de los valores de fitness, que sigue un comportamiento de crecimiento casi lineal hasta que el algoritmo detiene su mejora, estancándose en valores cercanos al mejor valor de fitness hallado como consecuencia de la pérdida de diversidad en la población.

En cuanto a la eficiencia en términos de tiempo, el gráfico ejemplifica el hecho de que el modelo de subpoblaciones distribuidas resultó del orden de 3.5 veces más rápido que el modelo panmíctico, al utilizar 4 procesadores. Se destacan los valores finales del tiempo de ejecución T_{PR} = 1734 segundos para el modelo paralelo y T_{PN} = 6382 segundos para el modelo panmíctico. El speedup casi lineal obtenido alienta a investigar la escalabilidad del modelo para la resolución de problemas de dimensión mayor.

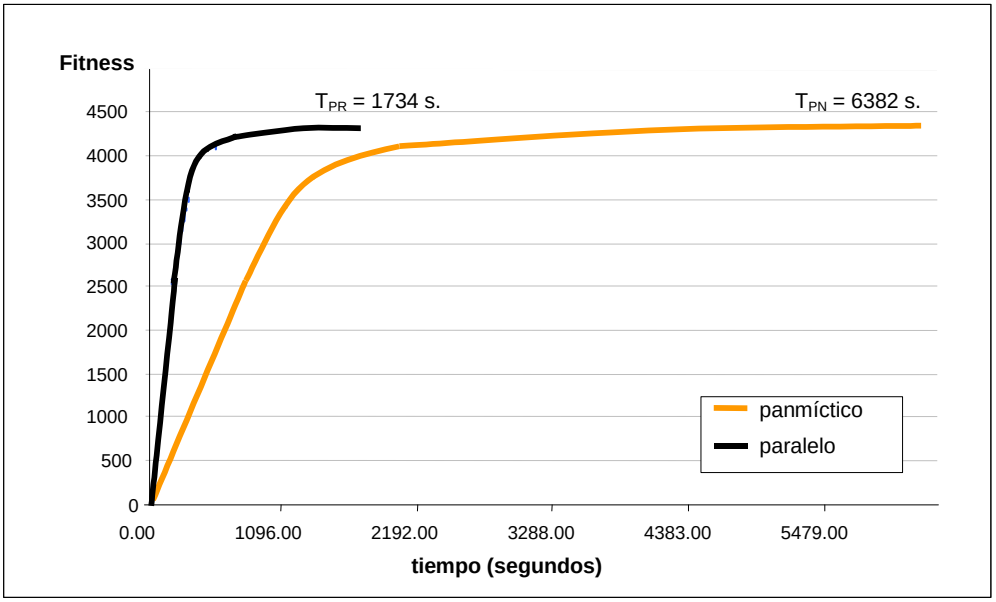


Figura 5.17: Comparación de eficiencia entre modelo panmíctico y paralelo.

5.9. Conclusiones

Los resultados obtenidos en la primer propuesta de resolución del Problema de Steiner Generalizado mediante un algoritmo genético paralelo ofrecen perspectivas promisorias sobre la aplicabilidad de las técnicas de computación evolutiva para resolver problemas de optimización vinculados al diseño de redes de comunicaciones confiables.

El enfoque evolutivo panmíctico serial se mostró preciso para las pruebas de validación, obteniendo mejores resultados que la heurística Ant Systems. Los resultados del algoritmo genético paralelo demostraron que configurando adecuadamente los parámetros de migración es posible alcanzar soluciones de calidad similar a las del algoritmo genético panmíctico serial, aún utilizando un pequeño número de demes.

Respecto a la performance del modelo de poblaciones distribuidas, evaluada mediante el tiempo de ejecución del algoritmo, el análisis mostró que es posible obtener significativas mejoras de eficiencia respecto al modelo panmíctico serial, cuando se utiliza un número reducido de subpoblaciones.

Estos resultados alientan a continuar la investigación en el área, estudiando tópicos de diseño no considerados en el estudio preliminar, ampliando el horizonte de aplicabilidad y analizando aquellos factores para los cuales no fue posible extraer afirmaciones concluyentes.

Algunos aspectos dejaron líneas de investigación abiertas para abordar en el futuro inmediato. Tomando en cuenta los promisorios resultados obtenidos, una línea lógica de trabajo futuro se orienta hacia el estudio de otros modelos de algoritmos evolutivos paralelos para abordar instancias de grandes dimensiones y su comparación con posibles resultados de algoritmos exactos y obtenidos aplicando otras heurísticas de optimización.

La escasa disponibilidad de recursos computacionales limitó los estudios iniciales de configuración y eficiencia del modelo. Contemplando alternativas para aumentar el número de equipos del cluster será posible llevar a cabo las investigaciones en torno a los aspectos cuya influencia no pudo ser aclarada. Algunos puntos a abordar en el futuro comprenden:

- Investigar la escalabilidad del modelo al aumentar los recursos computacionales, estudiando las mejoras de eficiencia al atacar problemas de mayor dimensión.
- Estudiar las topologías de migración, con el objetivo de determinar una topología óptima que garantice un equilibrio que no disminuya de modo excesivo la presión de selección atribuida a la migración, sin generar comunicaciones excesivas que afecten la eficiencia del modelo para múltiples demes.
- Investigar las representaciones alternativas planteadas para las soluciones al problema, que trabajan con soluciones factibles, de modo de evitar el descarte o la corrección luego de aplicados los operadores evolutivos. Otra alternativa es trabajar con soluciones no factibles, penalizando el valor de fitness de modo de conservar características de soluciones no factibles pero próximas a los óptimos.
- Estudiar modelos alternativos para algoritmo genético paralelo de población distribuida. Se han realizado avances en un modelo de poblaciones dinámicas (Nowostaski y Poli, 1999) que adapta el tamaño de los demes a la potencia computacional del equipo donde ejecuta. Esta solución permite mantener la uniformidad de uso de los recursos computacionales de modo de evitar desfasajes evolutivos entre las poblaciones en un entorno de cómputo heterogéneo.

CAPÍTULO 6

TÉCNICAS EVOLUTIVAS APLICADAS AL PROBLEMA DE STEINER GENERALIZADO

"Now, here, you see, it takes all the running you can do, to keep in the same place.
If you want to get somewhere else, you must run at least twice as fast as that."

LEWIS CARROLL
Through the Looking-Glass, 1872.

6.1. **Introducción**

Los resultados obtenidos en el trabajo con la primera propuesta diseñada de un algoritmo genético paralelo presentaron promisorias perspectivas sobre la aplicación de técnicas metaheurísticas evolutivas para la resolución del Problema de Steiner Generalizado.

En una instancia de difusión de los resultados preliminares del proyecto se tomó contacto con el grupo de trabajo en Redes y Técnicas de Optimización Emergente (NEO – *Networking and Emerging Optimization*) del Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga en España, cuyos integrantes se mostraron interesados en el contexto del problema y el enfoque utilizado para su resolución. Del intercambio de ideas sobre el problema y los métodos de resolución surgieron diversas propuestas para intentar abordar la resolución del Problema de Steiner Generalizado mediante otras técnicas evolutivas y de búsqueda global con el objetivo de obtener resultados de alta eficiencia numérica.

Este capítulo presenta la aplicación de otras metaheurísticas evolutivas y de búsqueda local para la resolución del Problema de Steiner Generalizado, que complementa al enfoque del algoritmo genético paralelo específico presentado en el capítulo precedente y tiene como principal objetivo disponer de nuevos algoritmos para evaluar comparativamente la calidad de las soluciones obtenidas. Considerando el enfoque utilizado en el capítulo previo, basado en la idea de mantener la codificación del problema y los operadores evolutivos tan simples como fuera posible, se diseñaron otros algoritmos aplicados al problema en cuestión. Para la implementación de los algoritmos se utilizó una biblioteca de propósito general para optimización combinatoria que provee esqueletos de software de diversos algoritmos para optimización y que permite tratar de un modo eficiente con los aspectos de paralelismo.

El capítulo comienza ofreciendo los detalles del enfoque del problema, que se corresponden a los presentados en el capítulo previo para el algoritmo genético paralelo específico. En la sección siguiente se introducen los algoritmos estudiados: algoritmos genéticos en su formulación clásica y en su variante CHC, simulated annealing y dos técnicas híbridas que combinan algoritmos genéticos y simulated annealing, presentando los conceptos de sus implementaciones seriales y paralelas sobre la biblioteca utilizada. Luego se presenta un detallado análisis de la calidad de los resultados obtenidos y de la eficiencia computacional para los diferentes algoritmos al resolver las instancias de prueba presentadas en el capítulo previo. La sección final del capítulo presenta las conclusiones sobre la aplicación de técnicas metaheurísticas evolutivas para resolver el problema considerado y líneas abiertas para continuar la investigación en el futuro.

6.2. Enfoque del problema

El enfoque utilizado para la codificación, implementación y resolución del problema es el mismo que se utilizó en el trabajo con el algoritmo genético paralelo específico que se presentó en el capítulo previo.

Se utilizó la codificación binaria simple basada en aristas para representar grafos que constituyen soluciones factibles del Problema de Steiner Generalizado. Se aplicó la propuesta de descartar las soluciones no factibles generadas por los operadores evolutivos. Asimismo, se utilizó el chequeo de factibilidad de dos componentes, la heurística para chequear los grados de los nodos terminales, combinada con la variante del algoritmo de Ford–Fulkerson para hallar los caminos entre cada par de nodos terminales del problema.

La función de fitness corresponde a la utilizada en el capítulo anterior para evaluar el costo del grafo representado, transformando el problema a una maximización.

6.3. Algoritmos considerados en el estudio

Esta sección presenta los algoritmos heurísticos considerados para aplicarse a la resolución del Problema de Steiner Generalizado en el marco de este trabajo: algoritmos genéticos en su formulación clásica y en su variante elitista CHC, simulated annealing y dos técnicas híbridas que combinan algoritmos genéticos y simulated annealing. Se presentan asimismo las características de las implementaciones paralelas de los algoritmos estudiados.

Simulated annealing no es una técnica evolutiva, sino que es un método de búsqueda local inspirado en el proceso físico de enfriamiento gradual utilizado en elementos sólidos con el fin de obtener un estado cristalino de energía mínima. Hemos utilizado este método para disponer de valores de referencia para comparar los resultados de las técnicas evolutivas, tomando en cuenta que un esqueleto de su algoritmo se encuentra disponible en la biblioteca de propósito general para optimización combinatoria utilizada. Asimismo, se ha utilizado el método simulated annealing para diseñar algoritmos híbridos combinándolo con el mecanismo evolutivo de un algoritmo genético.

6.3.1. Algoritmo Genético

La formulación clásica de un algoritmo genético fue presentada en el Capítulo 2. Basado en el esquema genérico de un algoritmo evolutivo presentado en la Figura 2.1, el algoritmo genético define operadores de selección, cruzamiento y mutación para aplicar a la población en cada generación. El algoritmo genético diseñado se basa en el presentado en el capítulo previo, aunque tiene como principal diferencia el utilizar un operador de cruzamiento de dos puntos en lugar del operador de cruzamiento de un punto utilizado por el algoritmo presentado en el Capítulo 5.

6.3.2. Algoritmo CHC

El acrónimo CHC corresponde a *Cross generational elitist selection, Heterogeneous recombination, and Cataclysmic mutation*, un algoritmo evolutivo propuesto por Eshelman (1991). CHC es una especialización del algoritmo genético tradicional que incorpora una estrategia conservadora o *elitista* de selección, perpetuando los mejores ***k*** individuos presentes en la población, que son siempre seleccionados para formar parte de la siguiente generación. El algoritmo incorpora una política de restricción al cruzamiento, mediante la utilización de un mecanismo especial para la recombinación: los padres se seleccionan aleatoriamente, pero solamente se permite la reproducción entre padres que difieran entre sí en un número determinado de bits especificado por una *cota de cruzamiento*.

El valor inicial de la cota de cruzamiento habitualmente se fija en 1/4 del largo del cromosoma empleado en la codificación del problema a resolver. En cada generación en que no se inserta ningún descendiente en la nueva población durante el proceso de reproducción (como consecuencia de la similitud en los individuos seleccionados como padres) el valor de la cota se reduce en una unidad.

Además de la política especial de restricción al cruzamiento mencionada, el propio operador de recombinación utilizado en el algoritmo CHC es diferente al operador de cruzamiento multipunto tradicionalmente utilizado en la versión clásica de un algoritmo genético. El operador de cruzamiento de CHC es llamado *operador de cruzamiento uniforme* y trabaja intercambiando entre los dos individuos padre exactamente la mitad de los bits que sean diferentes entre ellos.

Otra característica distintiva del algoritmo CHC es que no utiliza un operador de mutación, sino que la diversidad en la población se introduce mediante un procedimiento de *reinicialización* que utiliza al mejor individuo hallado hasta el momento como patrón para crear nuevos individuos cuando se detecta la convergencia de la población, es decir, cuando no se insertan descendientes luego de un número de generaciones, situación indicada por el hecho que la cota de cruzamiento alcanza el valor cero. Al aplicar el mecanismo de reinicialización, la cota de cruzamiento vuelve a tomar su valor inicial y se continúa el mecanismo evolutivo basado exclusivamente en recombinaciones.

La Figura 6.1 presenta un seudocódigo del algoritmo CHC, basado en la propuesta original de Eshelman (1991). En resumen, el algoritmo CHC incorpora ciertas características que lo diferencian de los algoritmos genéticos tradicionales:

- La estrategia de selección y reemplazo fuertemente elitista.
- La utilización del operador de cruzamiento uniforme.
- La ausencia de mutación, que es sustituida por el operador de reinicialización.
- La incorporación de una política de restricción al cruzamiento, que no permite la recombinación de dos individuos muy similares.

```
Inicializar(P(0))
generacion = 0
cota_cruzamiento = LargoCromosoma/4
Evaluar(P(0))
Mientras (no CriterioParada) hacer
   Padres = Seleccion(P(generacion))
   Hijos = HUX(Padres)
   Evaluar(Hijos))
   NuevaPop = Reemplazar(Hijos,P(generacion))
   Si (NuevaPop == P(generacion)) entonces
      cota_cruzamiento --
   fin
   generacion ++
   P(generacion) = NuevaPop
   Si (cota_cruzamiento == 0) entonces
      Reinicializacion(P(generacion))
      cota_cruzamiento = LargoCromosoma/4
   fin
fin
retornar Mejor Solucion Hallada
```

Figura 6.1: Esquema del Algoritmo CHC.

En este trabajo se diseñaron dos variantes del algoritmo CHC, que se diferencian únicamente en el mecanismo de reinicialización utilizado. Para la variante originalmente diseñada, que fue denominada CHC1, se propuso utilizar un operador de reinicialización basado en el operador de mutación utilizado en el algoritmo genético, pero modificando un mayor número de aristas en cada solución. Luego de analizar un conjunto de resultados preliminares obtenidos por esta variante, se tuvo la percepción de que el mecanismo de reinicialización no era capaz de proveer la diversidad necesaria para el correcto funcionamiento del algoritmo. Se detectó que un elevado número de individuos generados eran descartados por ser soluciones no factibles al Problema de Steiner Generalizado y que el algoritmo caía en una situación de convergencia prematura, lejos de los mejores valores óptimos conocidos debido a la pérdida de diversidad en la población. Por lo tanto, se decidió diseñar una segunda variante del algoritmo, que fue denominada CHC2, utilizando un operador de reinicialización que forzara la factibilidad de las soluciones generadas, iterando su aplicación hasta obtener un individuo factible. Esta nueva variante mejorada demanda tiempos de cómputos significativamente mayores a los de la variante original, pero en contrapartida no presenta los problemas de convergencia prematura al proporcionar la diversidad necesaria para la correcta evolución del algoritmo.

6.3.3. Simulated Annealing

Simulated annealing es un método de búsqueda local que se basa en una simulación del proceso físico mediante el cual se obtiene la configuración de menor energía de un sistema de n partículas luego de un calentamiento. El estado de menor energía corresponde a la configuración más estable del sistema de partículas, y por ello reúne características de resistencia y cristalinidad deseables en la industria para mejorar las cualidades de un material.

El proceso físico consiste en calentar el material a una temperatura muy alta, de modo que sus átomos adquieran una distribución fuertemente aleatoria dentro de la estructura del material y la energía del sistema alcance un valor máximo. Se aplica entonces un proceso en el cual se hace descender lentamente y en etapas la temperatura del material, permitiendo que en cada etapa se alcance una configuración de equilibrio, es decir, una configuración optima para la temperatura considerada. Al final del proceso, los átomos forman una estructura cristalina altamente regular, el material alcanza así una máxima resistencia y se minimiza la energía del sistema.

Se ha comprobado experimentalmente que si durante el proceso se hace descender la temperatura bruscamente o no se aguarda el tiempo suficiente para alcanzar el estado de equilibrio en cada etapa, la estructura final del material no es la óptima.

Los métodos que explican el comportamiento de grandes cantidades de átomos de un sistema son estudiados en el área de la física conocida como Mecánica Estadística. Debido a que en un sistema hay del orden de 10^{23} átomos por cm^3, solamente puede estudiarse el comportamiento probabilístico del sistema en equilibrio a una temperatura dada. El proceso de enfriamiento fue modelado por Metrópolis et al. (1953), dando a cada átomo un desplazamiento aleatorio en cada paso del algoritmo y evaluando el cambio en la función de energía. Si la variación de energía es negativa se acepta el desplazamiento, mientras que si la variación de energía es positiva se acepta el desplazamiento con una probabilidad $e^{-\Delta E/(T.K)}$, siendo T la temperatura del sistema y K la constante de Boltzman. La generalización de este método para aplicarlo a la resolución de problemas de optimización fue propuesta por Kirkpatrick et al. (1983).

Simulated annealing no es una técnica evolutiva, ya que no está basada en una población que progresa de acuerdo a una emulación de la evolución natural. En su lugar, consiste en un método de búsqueda local basado *en trayectoria*, ya que en cada instante de tiempo mantiene una única solución candidata para el problema (análoga al estado actual del sistema físico de *n* partículas). El método tiene una función objetivo asociada (análoga a la función de energía del sistema) para la cual se pretende hallar el mínimo global (análogo al estado más estable del sistema físico).

El espacio de búsqueda se explora mediante un *operador de transición*, denominado también *movimiento*, que debe definirse de acuerdo a las características del problema a resolver. Para evitar la convergencia prematura en óptimos globales del problema, el método es capaz de aceptar soluciones peores que la actual, de acuerdo a una probabilidad relacionada con un parámetro de *temperatura* (*T*). Este parámetro del método no tiene una analogía obvia en el problema de optimización, ya que no existe tal parámetro libre, y por ello definir un esquema adecuado para la variación de temperatura (conocido como *esquema de enfriamiento* o de *decaimiento*) que permita evitar mínimos locales es un arte en sí mismo, y usualmente debe ser determinado de modo empírico mediante experimentos apropiados. El operador de transición define una exploración en etapas para diferentes valores del parámetro temperatura, que corresponde a proceso estocástico modelado mediante una cadena de Markov.

Los parámetros del método, el valor inicial para la temperatura, el número total de iteraciones a realizar y el largo de la cadena de Markov utilizada en cada etapa antes de actualizar el valor del parámetro temperatura, son usualmente determinados empíricamente para cada problema abordado.

Como se mencionó anteriormente, simulated annealing no es una técnica orientada a población, sino que mantiene una única solución tentativa para el problema en cada instante de tiempo y explora en su vecindad mediante un procedimiento de búsqueda local definido para el problema en cuestión por intermedio del operador de transición. La Figura 6.2 presenta un esquema para el algoritmo simulated annealing basada en el texto de Ycart (2002).

```
Inicializar(T)
paso = 0
sol = SolucionInicial()
valor = Evaluar(Sol)
repetir
   repetir
      paso ++
      nuevaSol = Generar(sol,T)     // Transición
      nuevoValor = Evaluar(nuevaSol)
      si Aceptar(valor,nuevoValor,T)
         sol = nuevaSol
         valor = nuevoValor
      fin
   hasta ((paso mod largoCadenaMarkov)==0)
   T = Actualizar(T)
hasta (CriterioParada)
retornar sol
```

Figura 6.2: Esquema del algoritmo simulated annealing.

6.3.4. Algoritmos Híbridos

Las técnicas de hibridación refieren a la inclusión de conocimiento dependiente del problema en cuestión en un método de búsqueda, con el objetivo de mejorar el mecanismo de exploración del espacio de soluciones (Davis et al, 1991).

La inclusión de conocimiento puede instrumentarse a nivel de la codificación empleada y/o a través de operadores específicos para la resolución del problema, diseñando los que se conocen como algoritmos *híbridos fuertes*.

Una segunda posibilidad consiste en combinar dos o más métodos de resolución capaces de resolver el mismo problema, tratando de tomar ventajas de sus características distintivas para diseñar un nuevo algoritmo más preciso y más eficiente para la resolución del problema. De este modo se construyen los conocidos como algoritmos *híbridos débiles*. El algoritmo híbrido construido de esta manera define un nuevo mecanismo de exploración del espacio de soluciones del problema, que determina de que modo y en qué momento se aplica cada uno de los métodos combinados, y cómo en los estados internos de cada algoritmo componente se reportan los resultados intermedios al otro algoritmo para que éste continúe la búsqueda. Usualmente, mediante el intercambio de pequeños conjuntos de soluciones parciales y/o de valores estadísticos es posible combinar algoritmos diferentes, diseñando algoritmos híbridos débiles de modo sencillo y con la posibilidad de esperar resultados eficientes y precisos en la resolución de un problema determinado.

En este trabajo hemos utilizado este último enfoque. Se diseñaron dos algoritmos híbridos combinando el algoritmo genético con el método simulated annealing. En la primer propuesta, denominada *GASA1* (acrónimo para *Genetic Algorithm* y *Simulated Annealing*, versión 1), el algoritmo genético utiliza en cada generación al método simulated annealing como un operador evolutivo interno, aplicado estocásticamente de acuerdo a una probabilidad asociada. La idea detrás de esta propuesta de combinación de algoritmos está dado por el hecho de que mientras el mecanismo evolutivo del algoritmo genético permite identificar regiones promisorias del espacio de búsqueda, el mecanismo de búsqueda local del método simulated annealing permite la explotación, es decir, la mejora de resultados puntuales mediante exploración en las vecindades de las soluciones actuales.

Un diagrama del funcionamiento del algoritmo híbrido GASA1 se presenta en la Figura 6.3.

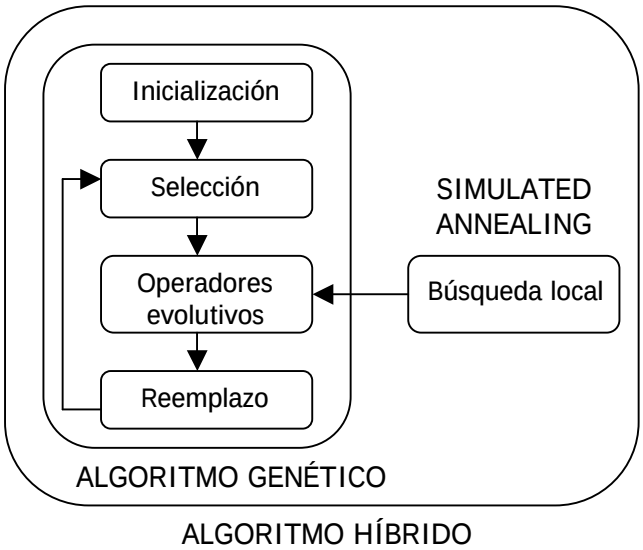


Figura 6.3: Esquema del algoritmo híbrido GASA1.

En la segunda propuesta, denominada *GASA2* (acrónimo para *Genetic Algorithm* y *Simulated Annealing*, versión 2), se ejecuta normalmente el algoritmo genético hasta su finalización y luego se seleccionan individuos de la población final, que se toman como soluciones iniciales sobre las cuales aplicar el algoritmo de búsqueda local del método simulated annealing. Concretamente, el algoritmo GASA2 utiliza torneo estocástico para seleccionar los individuos sobre los cuales aplicar simulated annealing, aunque es posible diseñar otras variantes que utilicen estrategias de selección diferentes.

Un diagrama del funcionamiento del algoritmo híbrido GASA2 se presenta en la Figura 6.4.

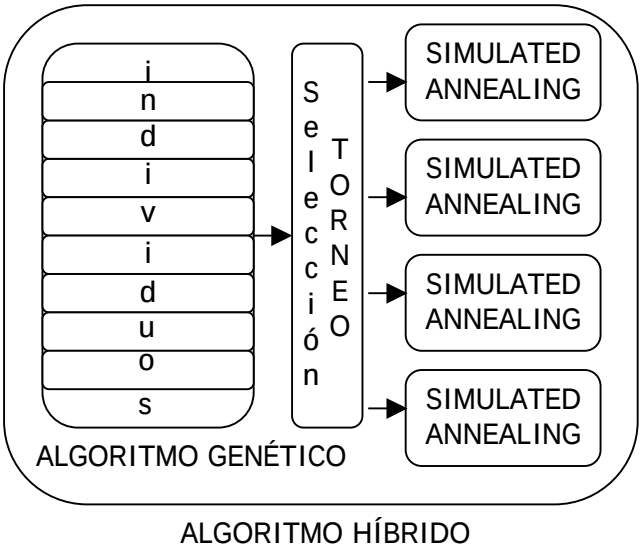


Figura 6.4: Esquema del algoritmo híbrido GASA2.

6.3.5. Algoritmos Paralelos

Las implementaciones paralelas se han popularizado como un mecanismo para mejorar el desempeño de los algoritmos evolutivos. Dividiendo la población entre varios elementos de procesamiento, los algoritmos evolutivos paralelos permiten abordar problemas difíciles de resolver o de grandes dimensiones y hallar resultados de buena calidad en tiempos razonables, tal como se expuso en el Capítulo 3.

Los modelos paralelos de los algoritmos basados en población presentados en este capítulo se clasifican en la categoría de subpoblaciones distribuidas (Nowostaski y Poli, 1999) (Alba y Tomassini, 2002). La población original se divide en varias subpoblaciones (*islas*) separadas geográficamente, cada una de las cuales ejecuta un algoritmo evolutivo secuencial donde los individuos solo son capaces de interactuar con otros en su misma isla. El operador adicional de *Migración* permite el intercambio ocasional de individuos entre islas, introduciendo una nueva fuente de diversidad en el mecanismo evolutivo.

La Figura 6.5 presenta un esquema paralelo genérico para un algoritmo evolutivo basado en población como los implementados en el marco de este trabajo. Dos condiciones controlan el procedimiento de migración: `EnviarEmigrantes` determina cuando se lleva a cabo el envío de individuos hacia una isla vecina, mientras que `RecibirInmigrantes` establece si se debe aguardar o no la recepción de un conjunto de individuos procedentes de otra isla. Estas dos condiciones toman los mismos valores simultáneamente en un modelo *sincrónico* de algoritmo evolutivo paralelo, donde las operaciones de envío y recepción se ejecutan de modo sincronizado; mientras que en un modelo *asincrónico* de algoritmo evolutivo paralelo estas dos condiciones están separadas en el tiempo. `Emigrantes` denota al conjunto de individuos a intercambiar con otra isla, seleccionados de acuerdo a una política determinada, mientras que `Inmigrantes` denota al conjunto de individuos recibidos desde una población vecina. En el esquema presentado se distingue explícitamente entre los operadores de `Selección para Reproducción` y `Selección para Migración`; si bien ambos retornan conjuntos seleccionados de individuos sobre los cuales aplicar la operación respectiva, habitualmente se utilizan políticas de selección diferentes en cada caso. Las operaciones `MigraciónEnviar` y `MigraciónRecibir` llevan a cabo el intercambio de individuos entre islas de acuerdo a una topología de conexión definida sobre ellas, utilizándose usualmente un anillo unidireccional.

```
Inicializar(P(0))
generación = 0
Evaluar(P(0))
Mientras (no CriterioParada) hacer
  Evaluar(P(generación))
  Padres = Selección para Reproducción(P(generación))
  Hijos = Operadores de Reproducción(Padres)
  NuevaPob = Reemplazar(Hijos, P(generación))
  generación ++
  P(generación) = NuevaPob
  Si (EnviarEmigrantes)
    Emigrantes = Selección para Migracion(P(generación))
    MigraciónEnviar(Emigrantes)
  fin
  Si (RecibirInmigrantes)
    Inmigrantes = MigraciónRecibir()
    Insertar(Inmigrantes, P(generación))
  fin
fin
Retornar Mejor Solución Hallada
```

Figura 6.5: Esquema de un algoritmo evolutivo paralelo.

Por otra parte, varios enfoques han sido propuestos para implementar versiones paralelas del método simulated annealing (Kluwer, 2000). En este trabajo se utilizó un modelo de paralelismo simple, que ejecuta varias instancias secuenciales del algoritmo simulated annealing a partir de distintas soluciones iniciales. Las diferentes instancias cooperan esporádicamente, intercambiando las mejores soluciones obtenidas hasta el momento.

6.4. Operadores y parámetros

Los operadores utilizados por los algoritmos evolutivos implementados y la configuración de sus parámetros se presentan a continuación.

6.4.1. Operadores

El algoritmo genético utiliza selección proporcional, cruzamiento de dos puntos y mutación que invierte el valor de una arista en la representación.

El algoritmo simulated annealing utiliza un operador de transición que invierte los valores de cinco aristas en la representación y controla la factibilidad de los individuos generados mediante el mismo procedimiento que los algoritmos evolutivos. Utiliza un esquema proporcional de decaimiento para el parámetro temperatura, de acuerdo a la formulación de la Ecuación 6.1.

$$T_k = a \,.\, T_{k-1}$$

Ecuación 6.1: Esquema proporcional de decaimiento para la temperatura.

Por su parte, el algoritmo CHC emplea cruzamiento uniforme, selección elitista y un mecanismo de reinicialización que se basa en la inversión de valores de aristas en la representación.

La población es inicializada mediante el procedimiento de eliminación aleatoria de hasta un 5% de las aristas del grafo original presentado en el capítulo anterior, aplicando el chequeo de factibilidad para eliminar de la población inicial soluciones no factibles.

6.4.2. Configuración de parámetros

No se realizaron experimentos para analizar específicamente la configuración óptima de los valores de los parámetros para cada algoritmo evolutivo implementado. Se decidió trabajar con los valores paramétricos del trabajo presentado en el capítulo anterior, donde se hallaron valores óptimos para el tamaño de población y probabilidades de cruzamiento y mutación para un algoritmo genético específico que utilizó la misma codificación del problema y el mismo enfoque de resolución.

Se utilizó el mismo criterio de parada de esfuerzo prefijado para todos los algoritmos evolutivos basados en población, que se detienen al alcanzar un número determinado de generaciones. Se definió un alto valor límite para el número de generaciones (2000) intentando proveer a los algoritmos la capacidad de alcanzar resultados de calidad elevada. Se descartó utilizar un criterio basado en detectar la convergencia de la población porque no conduciría a una comparación justa de la calidad de resultados ni de los tiempos de ejecución de los algoritmos.

En el algoritmo simulated annealing se utilizó un factor de decaimiento proporcional para la temperatura fijado en $\alpha = 0.99$. Este valor alto eleva la posibilidad de aceptar soluciones peores que la actual, en un intento por brindarle al algoritmo mayores posibilidades de escapar de los numerosos mínimos locales del problema. Dado que no existían referencias sobre la aplicación de simulated annealing a la clase de problemas de Steiner, se realizó una serie de experimentos iniciales para determinar la parametrización adecuada del algoritmo. Luego de estos experimentos preliminares se decidió utilizar un criterio de parada de esfuerzo prefijado en 10000 iteraciones y una cadena de Markov de 250 pasos para la exploración local en la vecindad de las soluciones.

En los algoritmos evolutivos basados en población se utilizó un tamaño de 120 individuos para la población. En el algoritmo genético y en las dos variantes de híbridos diseñadas se fijó la probabilidad de cruzamiento en 0.9 y la probabilidad de mutación en 0.01, valores de configuración tomados del trabajo presentado en el capítulo precedente.

En el algoritmo CHC se fijó la probabilidad de aplicación del cruzamiento uniforme en 0.8, mientras que el proceso de reinicialización involucró al 35% de la población, valores determinados en experimentos preliminares orientados a hallar una configuración adecuada para estos parámetros que no tienen un análogo en el algoritmo genético. Se utilizó como operador de reinicialización una variante del operador utilizado como mutación en el algoritmo genético y como movimiento en el simulated annealing: un operador no determinístico que modifica hasta un 40% de los valores correspondientes a las aristas del grafo con probabilidad 0.5. En CHC1 las soluciones no factibles se descartan durante el proceso de reinicialización, mientras que en CHC2 el operador de reinicialización se aplica iterativamente hasta encontrar una solución factible. Los experimentos preliminares realizados sugirieron que la utilización de otro operador de reinicialización más simple, aplicado con probabilidad baja, no era capaz de generar la diversidad necesaria en la población para evitar que el algoritmo CHC convergiera prematuramente luego de aplicada un proceso de reinicialización.

En la variante de algoritmo híbrido GASA1 se aplicó el operador de simulated annealing con probabilidad 0.01. Se utilizó una cadena de Markov corta, de 10 pasos, y 20 iteraciones como criterio de parada, tratando de reducir el costo computacional extra que demanda la aplicación de simulated annealing como operador interno al algoritmo genético. En GASA2, donde el costo computacional extra demandado es inferior al de la variante anterior, dado que el método simulated annealing se aplica sobre las soluciones luego que el algoritmo genético finalizó su ejecución, se aplicó el método simulated annealing con probabilidad 0.01, se mantuvo el largo de la cadena de Markov en 10 pasos, pero se aumentó el número de iteraciones utilizadas como criterio de parada a 100, en un intento por proporcionar al algoritmo una mayor posibilidad de mejorar los resultados obtenidos.

En los algoritmos evolutivos paralelos basados en población se dividió la población total en 8 islas que se ejecutaron en forma distribuida, cada isla en uno de los 8 computadores disponibles. Se utilizó un operador de migración que envía 5 individuos a la subpoblación vecina más cercana, considerando a las islas conectadas en una topología de anillo unidireccional. El número de individuos migrados corresponde a una tasa de migración de 1/3 del total de la población, similar a la utilizada en el trabajo presentado en el capítulo anterior. Sin embargo, para reducir el tiempo dedicado a las comunicaciones, se utilizó una frecuencia baja de migración, aplicando el operador de intercambio de individuos cada 25 generaciones. Las políticas utilizadas para la selección de emigrantes y para reemplazo de inmigrantes se basan en la aplicación del operador de torneo estocástico en una muestra de 5 individuos de la población local.

6.5. La biblioteca MALLBA

El proyecto MALLBA (Alba y Cotta, 1999) constituye un esfuerzo para desarrollar una biblioteca de algoritmos para la resolución de problemas de optimización, capaz de tratar con el paralelismo. La biblioteca mantiene una interfaz amigable para el usuario sin descuidar los aspectos de eficiencia.

Los algoritmos presentados en este trabajo se encuentran implementados como *esqueletos de software* en la biblioteca MALLBA. Los esqueletos son esquemas genéricos que deben ser instanciados por parte del usuario con las características del problema a resolver. Cada esqueleto incorpora los detalles del método de resolución correspondiente, su interacción genérica con el problema y las consideraciones de su implementación paralela, mediante clases C++ *requeridas* y *provistas* que representan una abstracción de las entidades involucradas en cada método de resolución:

- Las *clases provistas* implementan aspectos internos del algoritmo de optimización, de un modo independiente al problema a resolver. Las principales clases provistas que proporciona la biblioteca son `Solver`, que implementa el propio algoritmo de optimización y `SetUpParams`, utilizada para fijar los parámetros de cada método. Otras clases provistas son utilizadas en métodos de optimización específicos.
- Las *clases requeridas* especifican la información relevante del problema a resolver. Cada esqueleto incluye las clases requeridas `Problem` y `Solution` que encapsulan las entidades dependientes del problema necesarias para los métodos de resolución, que deben ser instanciadas por parte del usuario de la biblioteca para indicar las características del problema y de la representación utilizada. Dependiendo del esqueleto correspondiente al algoritmo considerado, otras clases pueden ser requeridas.

La infraestructura utilizada en el proyecto MALLBA se compone de clusters de computadores localizados en las universidades de Málaga, La Laguna y Barcelona en España, conectados por redes de comunicaciones de alta velocidad. El código de los algoritmos que componen la biblioteca MALLBA está disponible públicamente en el sitio web del grupo NEO de la Universidad de Málaga: `http://neo.lcc.uma.es/mallba/easy-mallba.`

Mediante el uso de la biblioteca MALLBA fue posible realizar una codificación rápida y eficiente de diferentes prototipos de los algoritmos presentados en la Sección 6.2 para abordar las dificultades inherentes a la resolución del Problema de Steiner Generalizado. Considerando que la biblioteca proporciona esqueletos genéricos para los algoritmos considerados y facilidades para su ejecución distribuida, el esfuerzo de implementación se limitó a adaptar el enfoque del problema para incorporarlo a las clases requeridas de la biblioteca MALLBA y a implementar aquellos escasos aspectos específicos de los operadores utilizados.

6.6. Plataforma de ejecución

Todos los algoritmos fueron implementados utilizando la biblioteca MALLBA, codificada en el lenguaje C++ y que utiliza la implementación MPICH de la biblioteca de desarrollo de programas paralelos y distribuidos MPI. Para la ejecución de los algoritmos se utilizó un cluster de ocho computadores Intel Pentium IV a 2.4 GHz, cada uno con 512 Mb RAM y sistema operativo SuSE Linux 8.0, conectados por una red de área local Fast Ethernet a 100Mb/sec.

La infraestructura de hardware utilizada forma parte del proyecto TRACER, que propone el estudio de técnicas de optimización modernas y su aplicación para la resolución de problemas complejos de un modo preciso y eficiente. El proyecto TRACER se encuentra actualmente en desarrollo en forma conjunta entre la Universidad de Málaga y otras universidades españolas (TRACER, 2003).

6.7. Análisis empírico

Esta sección presenta los resultados obtenidos en los experimentos de evaluación de los modelos secuenciales y paralelos de los algoritmos estudiados, analizados desde el punto de vista de su calidad de resultados y de su eficiencia computacional.

Los experimentos de evaluación fueron realizados sobre el conjunto de problemas de prueba diseñado en el marco del proyecto, que fue presentado en el capítulo precedente. Se realizaron treinta ejecuciones independientes (utilizando diferentes semillas para el generador de números aleatorios utilizado) de los modelos secuenciales y paralelos de cada algoritmo sobre cada una de las tres instancias que componen el conjunto de problemas de prueba.

Los resultados se resumen y comentan en las siguientes subsecciones.

6.7.1. Resultados de los Algoritmos Secuenciales

Los resultados de los algoritmos secuenciales se resumen en la tabla de la Figura 6.6. Los algoritmos han sido identificados por sus acrónimos ya presentados (CHC1, CHC2, GASA1, GASA2). Además, hemos designado por sus acrónimos GA (*Genetic Algorithm*) al algoritmo genético y por SA al método simulated annealing. La tabla de la Figura 6.6 presenta los valores de fitness promedio, los mejores valores de fitness y la desviación estándar obtenidos en las 30 ejecuciones independientes realizadas sobre los tres problemas de prueba para cada uno de los algoritmos estudiados.

	GASA1	***GASA2***	***CHC1***	***CHC2***	***GA***	***SA***
			grafo 100-10			
Promedio	4491.5	4471.5	4449.0	4602.0	4469.5	4186.0
Mejor	4549.0	4507.0	4567.0	4634.0	4531.0	4302.0
Desv. Estándar	36.7	30.2	42.6	15.1	28.2	48.6
			grafo 75-25			
Promedio	5363.9	5335.4	3657.0	5479.0	5308.2	5171.7
Mejor	5429.3	5427.1	4574.9	5521.8	5406.7	5249.0
Desv. Estándar	39.8	49.2	372.6	21.2	50.2	51.91
			grafo 50-10			
Promedio	9373.3	9330.6	7646.3	9514.2	9316.8	9177.2
Mejor	9474.3	9538.3	9143.0	9596.5	9456.0	9321.9
Desv. Estándar	70.7	85.3	726.3	43.2	85.8	114.6

Figura 6.6: Resultados de los algoritmos secuenciales.

Al analizar los resultados presentados en la tabla de la Figura 6.6, es posible apreciar que el método simulated annealing muestra los peores resultados, notoriamente inferiores a los de las técnicas evolutivas basadas en población. Simulated annealing parece incapaz de manejar la complejidad intrínseca del Problema de Steiner Generalizado, al menos para la implementación propuesta, basada en un operador de transición muy simplificado. Dado que el método mantiene una única solución candidata para el problema, no es capaz de explorar el espacio de soluciones con la misma precisión de los algoritmos evolutivos basados en población. Se realizaron experimentos adicionales tratando de proveer diferentes patrones de búsqueda para el algoritmo, variando el operador de transición para introducir mayor diversidad, permitiendo la modificación de un mayor número de aristas, pero los resultados no mejoraron significativamente. Aún cuando se modificó el criterio de parada del algoritmo, permitiendo su ejecución durante un número muy elevado de iteraciones (se realizaron experimentos utilizando hasta 30000 iteraciones como criterio de parada), los resultados obtenidos por el método no fueron competitivos al compararlos con los obtenidos por los algoritmos evolutivos basados en población.

El algoritmo genético clásico obtuvo resultados aceptables para las tres instancias de prueba consideradas. Sin embargo, no fue capaz de superar el mejor valor conocido para el *grafo 100-10* (4561, obtenido con el algoritmo genérico paralelo especifico presentado en el capítulo anterior). Asimismo, los valores promedios de fitness alcanzados para los tres casos considerados se encuentran lejanos a los mejores valores conocidos para cada problema, sugiriendo que aún resta trabajo por realizar para calibrar adecuadamente el algoritmo genético y mejorar el mecanismo de exploración del espacio de soluciones.

La calidad de los resultados obtenidos por el algoritmo CHC fue inesperadamente superior a la del resto de los algoritmos evolutivos estudiados, siendo capaz de obtener resultados de alta precisión numérica con una simple configuración de parámetros. La primer variante diseñada, CHC1, obtuvo resultados de calidad similar a los del algoritmo genético y los algoritmos híbridos para el *grafo100-10*, pero su comportamiento estadístico fue errático y poco consistente, al presentar menores valores de fitness promedio y elevados valores de desviación estándar en los resultados. Una posible explicación, como fue adelantado en la Sección 6.2.2, es que el procedimiento de reinicialización utilizado en esta primera variante del algoritmo es muy débil, conduciendo a situaciones en las cuales no existe la diversidad necesaria en la población para que el mecanismo evolutivo del algoritmo progrese. Este comportamiento fue detectado en un número considerable de ejecuciones de CHC1 sobre las instancias de prueba *grafo 50-15* y *grafo 75-25*, en las cuales el algoritmo no fue capaz de alcanzar resultados precisos. En estos casos CHC1 manifestó problemas de convergencia prematura o bien se mantuvo estancado durante muchas generaciones debido a la similaridad genotípica entre individuos de la población. Este hecho llevó a proponer la mejora del mecanismo de reinicialización, diseñando el algoritmo CHC2. Este algoritmo, que itera el proceso de reinicialización hasta lograr individuos factibles, mostró un comportamiento superior, sobrepasando significativamente la calidad de resultados de la variante inicial de CHC, del algoritmo genético y de los híbridos implementados, para las tres instancias de prueba estudiadas. Este resultado sugirió que el uso del operador de cruzamiento uniforme, que guía el mecanismo evolutivo del algoritmo CHC, era una idea promisoria para obtener resultados precisos para el Problema de Steiner Generalizado que merecía un estudio profundizado en el futuro. En un trabajo posterior se estudió empíricamente el uso de diferentes familias de operadores de cruzamiento en un algoritmo genético para la resolución del problema (Pedemonte y Nesmachnow, 2003), comprobando que la familia de operadores de cruzamiento uniforme permite alcanzar rápidamente individuos de alta calidad y globalmente mejores resultados para el problema que los operadores de cruzamiento multipunto. El trabajo referido se presenta en el Anexo III.

La técnica de hibridación aplicada en el algoritmo GASA1 produjo resultados con superiores valores de fitness promedio que los del algoritmo genético clásico, mientras que la aplicada en GASA2 no logró superarlos significativamente. De este resultado se concluye que el refinamiento gradual obtenido al aplicar el método de simulated annealing como operador interno constituye una mejor estrategia que aplicar un refinamiento final "fuerte". Este hecho fue verificado al estudiar la evolución de los mejores valores de fitness en las generaciones, en la cual se detectó que GASA1 permite obtener individuos adaptados rápidamente, ayudando al algoritmo genético a mejorar la población en cada generación, mientras que cuando GASA2 aplica el método simulated annealing los individuos ya tienen una calidad tal que rara vez es capaz de mejorarlos. Al utilizarse como operador interno en GASA1, la técnica de simulated annealing introduce una nueva fuente de diversidad que ayuda al proceso evolutivo del algoritmo genético. Si bien GASA1 mejora tanto los valores promedio de fitness como los mejores valores obtenidos para los grafos considerados, el factor de mejora no puede considerarse como de relevancia significativa.

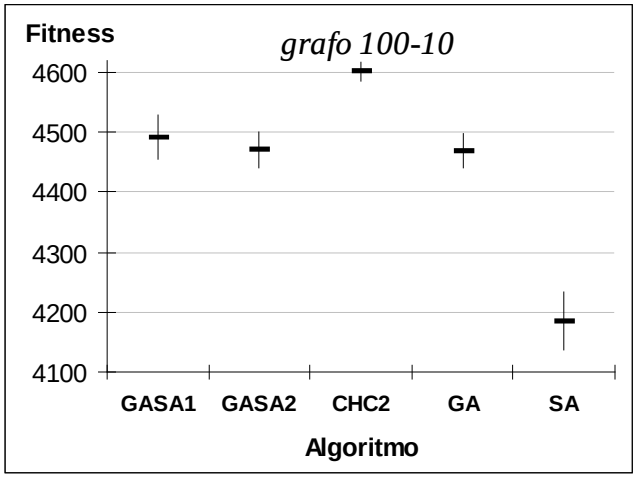


Figura 6.7: Resultados de los algoritmos seriales para la instancia grafo 100-10.

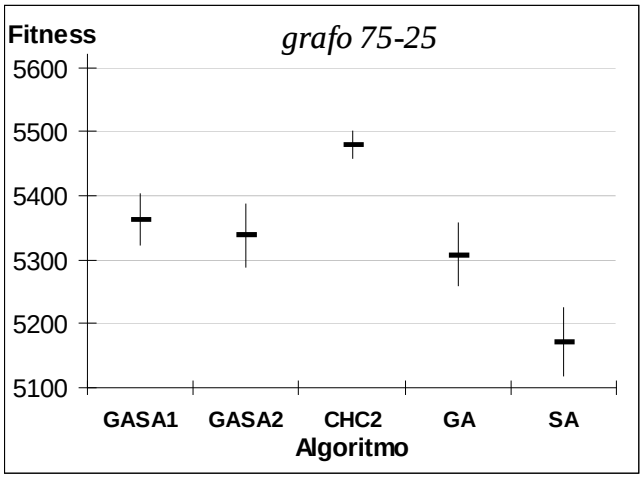


Figura 6.8: Resultados de los algoritmos seriales para la instancia grafo 75-25.

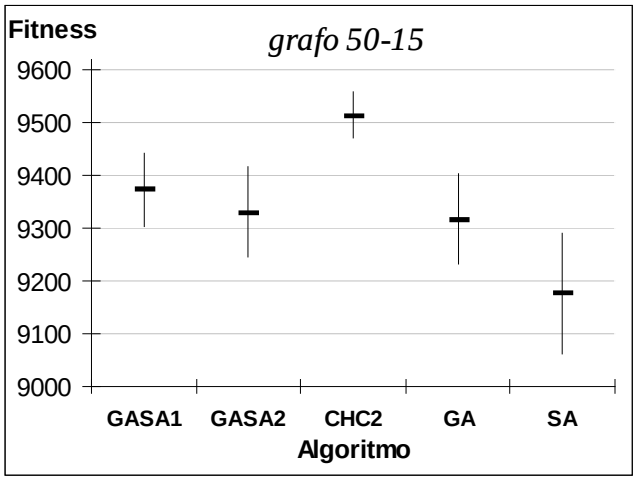


Figura 6.9: Resultados de los algoritmos seriales para la instancia grafo 50-15.

Las Figuras 6.7, 6.8 y 6.9 presentan gráficamente los resultados comparativos de las versiones seriales de los algoritmos estudiados, mostrando los valores de fitness promedio y las desviaciones estándar para cada problema del conjunto de prueba. No se han incluido los resultados de CHC1 en las gráficas ya que el algoritmo presentó problemas de convergencia prematura en valores lejanos al óptimo para las instancias *grafo 50-15* y *grafo 75-25*.

En las Figuras 6.7, 6.8 y 6.9 se pone de manifiesto claramente la superioridad de los resultados obtenidos por el algoritmo CHC2, la pobre precisión alcanzada por el método simulated annealing y la mejora poco significativa de los algoritmos híbridos cuando se comparan sus resultados con los del algoritmo genético clásico.

6.7.2. Análisis de la Evolución del Fitness

En esta subsección se analiza un importante aspecto de los algoritmos evolutivos: la evolución de los valores de fitness sobre las generaciones. Las Figuras 6.10, 6.11 y 6.12 presentan la evolución de los mejores valores de fitness sobre las generaciones, observados para las versiones seriales del algoritmo genético, el algoritmo CHC2, y el híbrido GASA1 durante ejecuciones representativas sobre las instancias *grafo100-10*, *grafo75-25* y *grafo50-15* respectivamente. Las gráficas presentan un acercamiento a los valores de fitness en las primeras 300 generaciones, para apreciar mejor las diferencias entre los valores obtenidos.

Los algoritmos GASA2 y CHC1 fueron omitidos intencionadamente en el análisis. GASA2 tiene el mismo comportamiento respecto a la evolución de valores de fitness que el algoritmo genético, ya que no existe diferencia entre su operativa interna hasta que el algoritmo genético se detiene, mientras que CHC1 presenta un comportamiento similar a CHC2 en las primeras generaciones pero luego se estanca debido a los problemas de convergencia prematura.

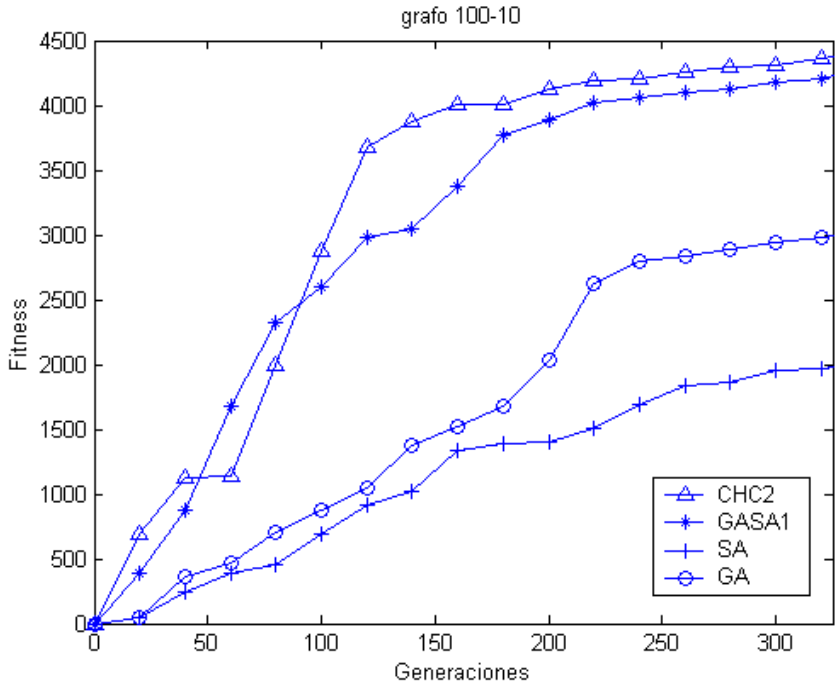


Figura 6.10: Evolución del fitness para la instancia grafo100-10.

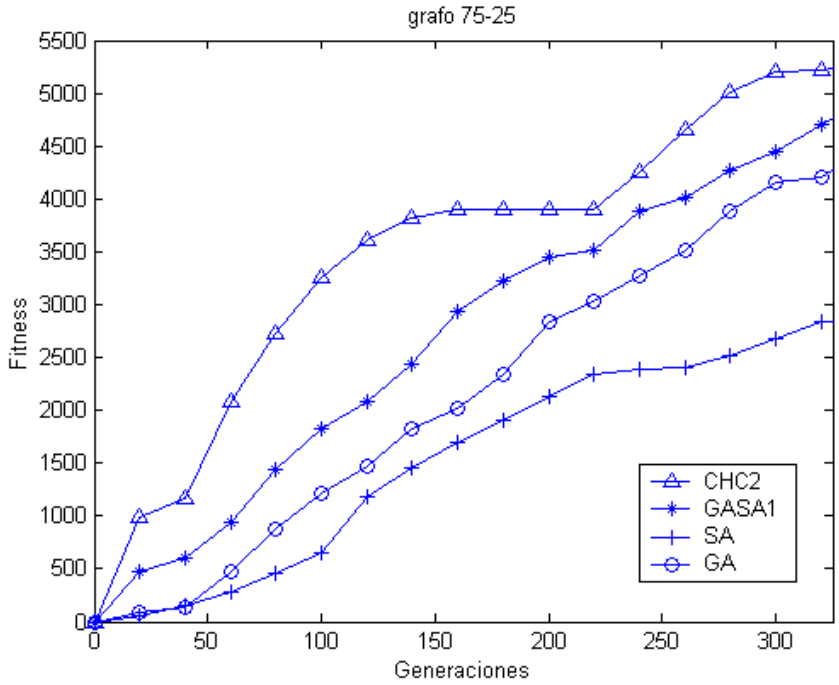


Figura 6.11: Evolución del fitness para la instancia grafo75-25.

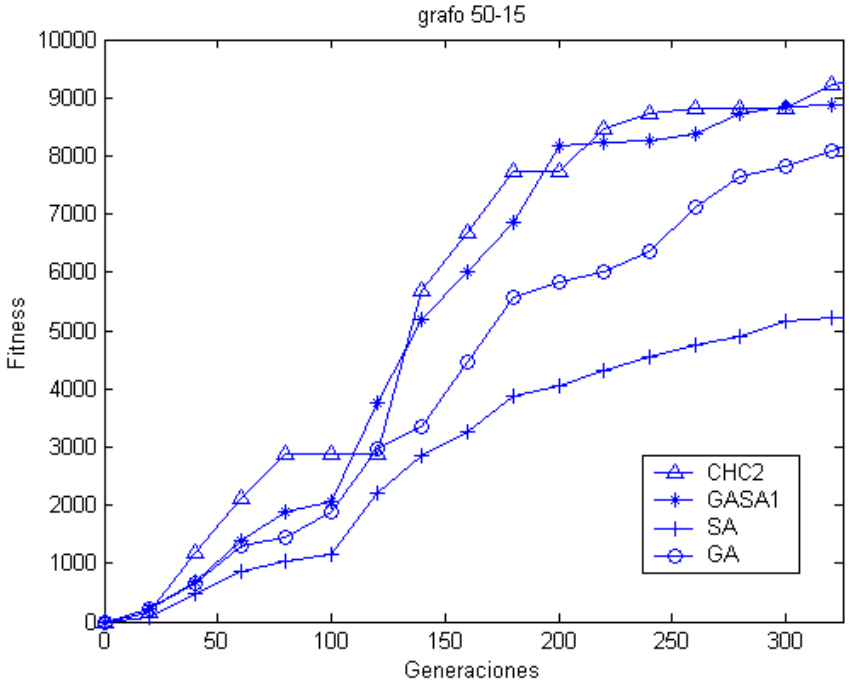


Figura 6.12: Evolución del fitness para la instancia grafo50-15.

Es posible apreciar que sobre la instancia *grafo100-10* tanto el híbrido GASA1 como CHC2 son capaces de encontrar individuos muy adaptados en un número moderado de generaciones, alcanzando altos valores de fitness en menos de doscientas generaciones. El algoritmo genético evoluciona más lentamente, requiriendo usualmente más generaciones antes de hallar resultados de calidad comparable. El método simulated annealing presenta el peor comportamiento, siguiendo un patrón más letárgico de evolución de fitness.

El comportamiento descrito anteriormente no se mantiene sobre las instancias *grafo75-25* y *grafo50-15*. Aunque GASA1 y CHC2 alcanzan rápidamente valores elevados de fitness, las distancias entre sus resultados y los del algoritmo genético no son tan marcadas. Sobre la instancia *grafo50-15* las diferencias se reducen significativamente. Nuevamente, el método simulated annealing presenta los peores resultados en cada instancia estudiada del problema.

En los algoritmos evolutivos basados en población, fue posible identificar a la pérdida de diversidad como el principal motivo por el cual los algoritmos detienen la evolución de sus valores de fitness al avanzar las generaciones. Tomando en cuenta que el espacio de soluciones factibles del problema tiene una dimensión muy reducida comparado con el número de soluciones no factibles, y que a medida que los algoritmos avanzan, moverse de una solución factible a otra factible se hace cada vez menos probable, la importancia de los mecanismos para mantener la diversidad y permitir la exploración de otras secciones del espacio de búsqueda es evidente.

En las gráficas 6.10, 6.11 y 6.12 puede apreciarse que los métodos más efectivos (CHC2 y GASA1) alcanzan buenas soluciones rápidamente, pero la mejora se detiene luego de algunos cientos de generaciones. El resto de la evolución discurre entre la generación de soluciones no factibles que son eliminadas o la exploración de soluciones de calidad inferior a la mejor solución ya encontrada. Las características del problema hacen que la incorporación de mecanismos para incorporar diversidad sean de gran ayuda para el proceso evolutivo, y por ello el operador de reinicialización "fuerte" que utiliza el algoritmo CHC2 le permite alcanzar resultados de calidad superior a los de los restantes algoritmos.

6.7.3. Resultados de los Algoritmos Paralelos

La tabla de la Figura 6.13 resume los resultados de los algoritmos paralelos para los problemas estudiados. Como fue comentado, los algoritmos paralelos utilizaron ocho islas, cada una ejecutando sobre un equipo del cluster empleado, cuya configuración se presentó en la sección 6.6. La tabla de la Figura 6.13 reporta los promedios de valores de fitness y la desviación estándar en los resultados para las 30 ejecuciones independientes de cada algoritmo. Como consecuencia de que el alto número de combinaciones de algoritmos e instancias de prueba demanda extensos períodos de tiempo para completar los experimentos, sólo se realizaron ejecuciones paralelas para el algoritmo simulated annealing y para aquellos algoritmos basados en población que mostraron resultados promisorios en los experimentos secuenciales (el algoritmo genético clásico, la versión de CHC mejorada –CHC2– y el algoritmo híbrido GASA1).

	GASA1	***CHC2***	***GA***	***SA***
		grafo 100-10		
Promedio	4529.0	4436.5	4489.0	4170.5
Mejor	4608.0	4566.0	4537.0	4289.0
Desv. estándar	32.5	51.3	24.5	48.2
		grafo 75-25		
Promedio	5415.5	5353.2	5379.4	5174.2
Mejor	5440.9	5408.2	5447.2	5290.9
Desv. estándar	28.4	34.5	38.8	41.4
		grafo 50-15		
Promedio	9473.1	9418.2	9393.3	9266.3
Mejor	9569.7	9596.5	9535.3	9441.0
Desv. estándar	34.0	70.7	86.7	81.6

Tabla 6.13: Resultados de los algoritmos paralelos.

El estudio se concentró principalmente en evaluar la calidad de los resultados obtenidos por los modelos paralelos, tratando de determinar si éstos eran capaces de mejorar la calidad de los resultados de los algoritmos secuenciales. Una de las características de los modelos paralelos de algoritmos evolutivos lo constituye el hecho de que las islas tienen la capacidad de evolucionar en diferentes direcciones, explorando distintas secciones del espacio de búsqueda. Basados en este aspecto, los modelos paralelos introducen un mecanismo de evolución diferente, que es capaz de aportar la diversidad necesaria para mejorar los resultados de los algoritmos seriales en problemas complejos (Cantú-Paz, 2000).

En el trabajo previo presentado en el capítulo anterior no fue posible detectar mejoras significativas en los resultados del modelo paralelo de algoritmo genético sobre el modelo serial. Este hecho fue explicado por la alta precisión de los resultados obtenidos por el algoritmo serial, como consecuencia del alto número de generaciones utilizado como criterio de parada. Por su parte, en el estudio de los modelos seriales de los algoritmos presentados en este capítulo fue posible confirmar que existía un margen importante para mejorar los resultados de varios de los algoritmos implementados en caso de incorporar mecanismos para mejorar la diversidad de la población. Tomando en cuenta estos argumentos, el objetivo de los experimentos con los modelos paralelos consistió en evaluar si una mejora en la calidad de resultados podía ser alcanzada.

La eficiencia de los algoritmos paralelos se evaluó como un objetivo secundario, ya que no se realizó trabajo extra de implementación para optimizar los tiempos de ejecución de los algoritmos. Como ejemplo, por las características del modelo evolutivo empleado por los esqueletos de software de la biblioteca utilizada para su desarrollo, los algoritmos implementados realizan chequeos de factibilidad superfluos luego de aplicado cada operador evolutivo. Por motivos de tiempo en el desarrollo del proyecto, la implementación no fue optimizada para reunir la recombinación y la mutación en un único operador de modo de evitar chequeos de factibilidad superfluos.

De todos modos, se evaluaron los tiempos de ejecución de los algoritmos y se calcularon los valores de speedup y eficiencia al utilizar las ocho islas ejecutando en ocho procesadores. El objetivo consistió en tratar de determinar si los algoritmos evolutivos paralelos implementados sobre la biblioteca MALLBA mantenían el comportamiento de speedup casi lineal, como el observado en el algoritmo genético específico presentado en el capítulo precedente.

Analizando los datos presentados en la tabla de la Figura 6.13, y comparándolos con los resultados de los algoritmos seriales presentados en la tabla de la Figura 6.6 es posible concluir que desde un punto de vista global, los modelos paralelos superaron los valores de fitness promedio y los mejores valores obtenidos por los algoritmos seriales, excepto para el algoritmo CHC2. Cuando se divide la población en islas de tamaño reducido, el algoritmo CHC2 paralelo no es capaz de mejorar la calidad de resultados de su contraparte serial, y ni siquiera llega a alcanzar una calidad similar de resultados. Este efecto es consistente con un aspecto de los algoritmos evolutivos paralelos frecuentemente reportado en la literatura del área: si las poblaciones son pequeñas, la división en múltiples nichos puede ser contraproducente para el funcionamiento del modelo de evolución. Como consecuencia de su mecanismo de selección elitista, trabajando sobre demes de población reducida el algoritmo CHC pierde la diversidad necesaria en la población, y el caso estudiado sufre un problema de pérdida de diversidad y convergencia prematura similar al manifestado por la versión originalmente diseñada para el algoritmo serial CHC1. Tomando en cuenta este resultado, el diseño de un modelo apropiado de paralelismo para aplicar al algoritmo CHC constituye una línea lógica para abordar como trabajo futuro.

6.7.4. Resultados comparativos

La tabla de la Figura 6.14 resume las mejoras en calidad de resultados obtenidas por los modelos paralelos de los algoritmos diseñados para cada instancia del conjunto de prueba considerado. Se presentan comparativamente los valores promedio de fitness y los mejores valores alcanzados en las 30 ejecuciones realizadas para cada instancia del problema, calculando un *factor de mejora* relativo al valor correspondiente del algoritmo serial.

Los factores de mejora no son elevados, como consecuencia de la alta precisión de los resultados seriales. Sin embargo, para el algoritmo genético, para el híbrido GASA1 y en un caso para el método simulated annealing, se verifica que las mejoras obtenidas son mayores que la desviación estándar de los resultados para los tres casos estudiados. Para el algoritmo genético y el híbrido GASA1, donde fueron detectadas mejoras significativas, se realizó el test de Kruskal-Wallis para analizar las distribuciones de los resultados. Como ejemplo, para GASA1 los *p-values* son inferiores a 0.01, y por tanto puede considerarse que las diferencias entre promedios de valores de fitness de los modelos paralelo y secuencial son significativas al nivel de 1%. Este resultado indica que aunque pequeña, la mejora obtenida por los algoritmos paralelos en los casos mencionados puede considerarse significativa estadísticamente.

	GASA1	***CHC2***	***GA***	***SA***
grafo100-10				
Fitness Promedio Secuencial	4491.5	4602.0	4469.5	4186.0
Fitness Promedio Paralelo	4529.0	4436.5	4489.0	4170.5
Factor de Mejora en Fitness Promedio	1.008	0.964	1.004	0.996
p-value	$1.8\ 10^{-6}$	n/c	0.035	n/c
Mejor Fitness Secuencial	4549.0	4634.0	4531.0	4302.0
Mejor Fitness Paralelo	4608.0	4566.0	4537.0	4289.0
Factor de Mejora en Mejor Fitness	1.013	0.986	1.001	0.997
grafo 75-25				
Fitness Promedio Secuencial	5363.9	5479.0	5308.2	5171.8
Fitness Promedio Paralelo	5415.5	5353.2	5379.4	5174.2
Factor de Mejora en Fitness Promedio	1.010	0.977	1.014	1.001
p-value	$5.8\ 10^{-6}$	n/c	$1.9\ 10^{-6}$	n/c
Mejor Fitness Secuencial	5429.3	5521.8	5406.7	5249.0
Mejor Fitness Paralelo	5440.9	5408.2	5447.2	5290.9
Factor de Mejora en Mejor Fitness	1.002	0.980	1.008	1.008
grafo50-15				
Fitness Promedio Secuencial	9373.3	9514.2	9316.8	9177.2
Fitness Promedio Paralelo	9473.1	9418.3	9393.3	9266.3
Factor de Mejora en Fitness Promedio	1.011	0.990	1.008	1.010
p-value	$8.9.10^{-9}$	n/c	$5.1\ 10^{-4}$	n/c
Mejor Fitness Secuencial	9474.3	9596.5	9456.0	9321.9
Mejor Fitness Paralelo	9569.7	9596.5	9535.3	9441.0
Factor de Mejora en Mejor Fitness	1.010	1.000	1.008	1.013

Figura 6.14: Resultados comparativos entre modelos seriales y paralelos.

6.7.5. Análisis de la eficiencia computacional

Aunque la mejora de desempeño no fue el objetivo principal del trabajo, se analizaron los tiempos de ejecución de los algoritmos secuenciales y paralelos. La tabla de la Figura 6.15 resume un breve análisis de la eficiencia computacional, presentando los tiempos promedio de ejecución (en minutos) sobre las 30 ejecuciones independientes realizadas para cada problema de prueba.

	GASA1	***GA***	***CHC2***	***SA***	***GASA2***	***CHC1***
			grafo 100-10			
Secuencial	520.0	122.8	59.4	18.6	139.2	52.6
Paralelo	81.8	24.2	28.7	24	*	*
Eficiencia	0.79	0.63	0.26	0.09	*	*
			grafo 75-25			
Secuencial	462.6	125.4	51.0	9.0	126	11.4
Paralelo	81.0	24.0	20.4	14.4	*	*
Eficiencia	0.71	0.65	0.31	0.14	*	*
			grafo 50-15			
Secuencial	1229.4	310.2	275.4	31.8	319.8	31.2
Paralelo	219.6	59.4	74.4	28.8	*	*
Eficiencia	0.70	0.65	0.46	0.14	*	*

Figura 6.15: Tiempos de ejecución promedio (minutos).

Analizando los tiempos de ejecución para cada algoritmo, es posible observar que el método simulated annealing es el algoritmo más rápido. Este hecho puede ser explicado como consecuencia de que simulated annealing trabaja con una única solución en cada instante de tiempo mientras que los algoritmos evolutivos basados en población trabajan con un conjunto numeroso de individuos. Adicionalmente, ya en etapas intermedias del algoritmo comienza a manifestarse un comportamiento sistemático que domina las etapas avanzadas en la "evolución": al utilizar el simple operador de transición propuesto, el método simulated annealing descarta la mayoría de los individuos explorados de dos etapas por uno de dos motivos, o bien las nuevas soluciones generadas no mejoran a la solución actual y no son aceptadas probabilísticamente, o bien porque la no−factibilidad de las soluciones se detecta con el chequeo heurístico y no es necesario calcular los caminos entre pares de nodos terminales. En ambos casos se evita aplicar completamente el costoso chequeo de factibilidad identificado como el principal demandante de tiempo de cómputo en los algoritmos implementados. Pese a ser el algoritmo más rápido, simulated annealing no logra alcanzar soluciones de precisión aceptable comparadas con el resto de los algoritmos.

CHC1 presenta un comportamiento similar al de simulated annealing, mostrando tiempos de ejecución moderadamente bajos, pero alcanzando solamente resultados competitivos para la instancia *grafo 100-10*, donde logra superar sus problemas de convergencia prematura. Sobre las instancias *grafo 75-25* y *grafo* 5*0-15* presenta resultados muy pobres, como consecuencia de la convergencia prematura debida a la pérdida de diversidad en la población. La similitud entre individuos provoca la detención del mecanismo evolutivo, ya que la política de restricción al cruzamiento impide que los individuos similares se recombinen entre sí, y por ello los tiempos de ejecución se reducen globalmente, ya que al no aplicarse los operadores evolutivos el chequeo de factibilidad no es necesario.

El algoritmo híbrido GASA1 tiene una operativa interna mucho más compleja que el resto de los algoritmos, y este hecho se refleja en los elevados tiempo de cómputo que demanda para su ejecución. En promedio, el algoritmo GASA1 demanda tiempos de ejecución cuatro veces superiores a los del algoritmo genético tradicional. Por otra parte, aplicar el método simulated annealing luego de finalizada la ejecución del algoritmo genético no incrementa significativamente los tiempos de ejecución, y por ello el híbrido GASA2 tiene aproximadamente las mismas demandas de tiempo que el algoritmo genético clásico.

Es posible apreciar que el algoritmo CHC2 es significativamente más veloz que el algoritmo genético tradicional. Dado que el algoritmo CHC2 no aplica mutaciones continuas durante el mecanismo evolutivo e introduce condiciones para la restricción de cruzamiento entre individuos similares, los efectos de los chequeos de factibilidad superfluos se reducen notoriamente. También se aprecia que esta ventaja del algoritmo CHC2 tiende a reducirse cuando el numero de nodos terminales crece, dado que la razón entre el tiempo promedio de ejecución de CHC2 y el del algoritmo genético aumenta de 0.48 a 0.89 cuando el número de nodos terminales crece de 10 a 25. La tabla de la Figura 6.16 presenta los tiempos de ejecución comparativos para las versiones secuenciales de los algoritmos evolutivos basados en población, utilizando los tiempos de ejecución del algoritmo genético tradicional como referencia.

	grafo 100-10	***grafo 75-25***	***grafo 50-15***
GASA1/GA	4.23	3.96	3.69
GASA2/GA	1.13	1.03	1.00
CHC2/GA	0.48	0.89	0.41

Figura 6.16:Comparación de tiempos de ejecución para los algoritmos secuenciales.

Tomando en cuenta los tiempos de ejecución de los algoritmos, es posible conjeturar que la complejidad del problema se relaciona principalmente con el número de nodos terminales y es ligeramente afectada por el número total de nodos. Este hecho puede notarse en la comparación de tiempos de ejecución de cualquiera de los algoritmos utilizados al considerar las tres instancias abordadas: el esfuerzo máximo es siempre requerido para la resolución de la instancia *grafo 75-25*. Este comportamiento puede explicarse por el hecho de que el chequeo de factibilidad de soluciones, que demanda la mayor parte del esfuerzo computacional utilizado en cada generación, tiene una complejidad proporcional al cuadrado del número de nodos terminales del problema ($O(n_T^2)$).

Adicionalmente, se calcularon valores intuitivos del speedup alcanzado al ejecutar los algoritmos paralelos sobre 8 procesadores. Las versiones paralelas de los algoritmos evolutivos basados en población demandan menores tiempos de ejecución que sus contrapartes seriales ya que trabajan con poblaciones reducidas, mientras que el método simulated annealing constituye un caso especial, ya que no toma ventaja de la división de la población porque trabaja con un solo individuo y por ello no logra mejorar sus tiempos de ejecución al disponer de un mayor número de procesadores. La tabla de la Figura 6.15 ofrece un valor calculado de ***Eficiencia*** = ***Speedup/#Procesadores*** para aquellos algoritmos para los cuales se estudió su modelo paralelo (GASA1, CHC2, algoritmo genético y simulated annealing).

Si bien los algoritmos evolutivos paralelos basados en población muestran un *speedup* sublineal, puede apreciarse que el híbrido GASA1 y el algoritmo genético tienen buenos valores de eficiencia, en el entorno de 0.7. Por su parte, el algoritmo CHC2 mostró una eficiencia muy pobre al trabajar con poblaciones distribuidas, en el entorno de 0.3 para los grafos con pocos nodos terminales, y variando hasta 0.46 para el *grafo 75-25*. Este detalle muestra nuevamente que el algoritmo CHC2 paralelo no tiene el mismo patrón de comportamiento evidenciado por los otros algoritmos evolutivos paralelos basados en población.

6.8. Conclusiones y Trabajo Futuro

En el trabajo presentado en este capítulo se estudian varias técnicas evolutivas y de búsqueda local aplicadas a la resolución del Problema de Steiner Generalizado. Versiones seriales y paralelas de los algoritmos fueron implementadas sobre una biblioteca de propósito general para optimización combinatoria. Se ofrece un detallado análisis de los resultados obtenidos y un estudio del comportamiento de los algoritmos y de eficiencia computacional al resolver el conjunto de instancias de prueba presentado en el capítulo anterior.

Analizando los resultados obtenidos para el conjunto de problemas de prueba estudiado es posible extraer conclusiones sobre la aplicabilidad de los algoritmos implementados para la resolución del Problema de Steiner Generalizado.

El método simulated annealing no produjo resultados precisos para el Problema de Steiner Generalizado utilizando el sencillo operador aleatorio propuesto para la exploración de vecindades. Ni siquiera al permitir un número muy elevado de iteraciones el método fue capaz de alcanzar resultados competitivos con respecto a los algoritmos evolutivos basados en población. Con el simplificado operador de transición propuesto, el algoritmo simulated annealing trabaja prácticamente siguiendo una exploración aleatoria del espacio de búsqueda. Cabe destacar que el uso de este método fue propuesto para contar con una referencia para comparar los resultados de los algoritmos evolutivos, y por ello no se realizaron esfuerzos por mejorar su mecanismo de exploración, adoptándose una simple búsqueda aleatoria que extiende el operador de mutación de los algoritmos evolutivos. Un perfeccionamiento del operador de búsqueda local se impone como necesaria para mejorar los resultados obtenidos utilizando el método simulated annealing.

Aunque se obtuvieron valores similares desde un punto de vista numérico (promedios de mejores valores de fitness obtenidos) para el algoritmo genético y los dos algoritmos híbridos estudiados, ciertos detalles importantes pueden destacarse como conclusiones del estudio. El algoritmo híbrido GASA2, que aplica el método simulated annealing sobre la población final luego de aplicado el algoritmo genético, ha producido pobres resultados, mostrando una mejora negligible respecto a los resultados del algoritmo genético tradicional. Por su parte, el algoritmo híbrido GASA1, que utiliza el método simulated annealing como un operador interno en el algoritmo genético, es capaz de hallar mejores soluciones puntuales y acelerar el mecanismo de exploración del algoritmo genético tradicional, aunque las mejoras logradas no son significativas en el largo plazo, a medida que avanzan las generaciones. Los promedios de los mejores valores de fitness alcanzados son muy similares a los del algoritmo genético tradicional cuando se evoluciona durante 2000 generaciones. Por lo tanto, el principal aporte del método simulated annealing utilizado como un operador interno consiste en introducir una nueva fuente de diversidad, acelerando la exploración inicial del espacio de soluciones. Como contrapartida, los tiempos de ejecución demandados crecen considerablemente. Otras características del mecanismo de introducción de diversidad no son útiles para el algoritmo genético, como ejemplo, dado que no se estanca habitualmente en óptimos locales del problema, la capacidad del método simulated annealing de aceptar soluciones peores que la analizada no constituye una ayuda relevante para el mecanismo evolutivo.

CHC se manifestó como un algoritmo muy promisorio para la resolución del Problema de Steiner Generalizado, posiblemente tomando ventajas de su mecanismo especial de cruzamiento y del propio operador de cruzamiento uniforme que emplea. El algoritmo CHC fue capaz de obtener individuos de alta calidad en un número reducido de generaciones y es capaz de mejorar sus propios valores de fitness de modo más rápido y consistente que el resto de los algoritmos evolutivos estudiados. La primera versión diseñada (CHC1) mostró con frecuencia problemas de convergencia prematura como consecuencia del pobre mecanismo de reinicialización utilizado. Cuando se mejoró el operador de reinicialización, se obtuvo una versión mucho más robusta del algoritmo (CHC2) que fue capaz de alcanzar resultados de precisión muy elevada para las tres instancias del Problema de Steiner Generalizado estudiadas. El algoritmo CHC2 superó los mejores valores de fitness hallados por los restantes algoritmos, produjo valores de fitness promedio significativamente superiores, e inclusive presentó mejores valores de eficiencia computacional que el algoritmo genético y los híbridos implementados.

Estudiando los resultados algoritmo por algoritmo, se confirmó que los modelos paralelos de algoritmos evolutivos son capaces de hallar mejores soluciones para el Problema de Steiner Generalizado, mejorando ligeramente los valores hallados por los modelos seriales, salvo en el caso del algoritmo CHC2. Adicionalmente, la eficiencia computacional mejora notoriamente al dividir la población global en subpoblaciones que evolucionan en paralelo. Aunque los modelos paralelos de los algoritmos estudiados tuvieron un comportamiento de speedup sublineal, tanto el algoritmo genético clásico como el algoritmo híbrido GASA1 presentaron buenos valores de eficiencia al utilizar ocho subpoblaciones, mostrándose adaptables para la aplicación de técnicas de procesamiento paralelo.

Dos líneas principales surgen como inmediatas para abordar como parte del trabajo futuro relacionado con los aspectos inconclusos del trabajo: la profundización del estudio del comportamiento de los algoritmos y la investigación sobre los mecanismos pra mejorar la eficiencia computacional de los métodos estudiados.

Respecto al estudio del comportamiento de los algoritmos, debe analizarse en profundidad los mecanismos de hibridación para determinar sus contribuciones al resolver el problema, en especial tomando en cuenta el enorme tiempo computacional extra que demanda la aplicación de métodos de búsqueda local como operadores internos en el algoritmo genético. Asimismo, debe estudiarse en detalle la influencia del operador de reinicialización en el mecanismo evolutivo del algoritmo CHC, considerando que cuando no se provee la diversidad suficiente en la población el algoritmo presenta problemas de convergencia prematura, tal como fue detectado en la primera versión diseñada CHC1 y en el modelo paralelo de la versión CHC2. Complementariamente, el operador de reinicialización utilizado en la versión CHC2 no es eficiente, ya que en ocasiones es necesario aplicarlo en reiteradas ocasiones hasta producir una solución factible del problema. El diseño de un operador de reinicialización eficiente, que garantice la factibilidad de las soluciones generadas y proporcione la diversidad necesaria para el mecanismo evolutivo no es una tarea trivial, y requerirá estudios teóricos del problema y sus características conjuntamente con estudios experimentales que se planean desarrollar en el futuro.

Por otra parte, existe un amplio margen para mejorar la eficiencia tanto de los modelos seriales como de los paralelos de los algoritmos estudiados, y para realizar un estudio detallado de la eficiencia computacional que complemente el breve estudio presentado. Las implementaciones utilizadas en el trabajo no estuvieron optimizadas, y por ello derrochan tiempo de cómputo en chequeos de factibilidad superfluos que incrementan los tiempos de ejecución de todos los algoritmos. Una etapa de rediseño de los algoritmos aparece como evidente para poder abordar en el futuro instancias del Problema de Steiner Generalizado de mayor dimensión y complejidad, para las cuales las implementaciones actuales de los algoritmos demandarían tiempos de ejecución prohibitivos. Relacionado con este punto, debe profundizarse el estudio de la escalabilidad del modelo de poblaciones distribuidas para determinar su utilidad para la resolución de complejas instancias del Problema de Steiner Generalizado utilizando el poder computacional de grandes clusters de computadores.

CAPÍTULO 7

CONCLUSIONES Y TRABAJO FUTURO

"Les choses que nous concevons très clairement et très distinctement sont toutes vraies."

("Las cosas que concebimos muy clara y distintamente son todas verdaderas")

RENÉ DESCARTES

Discurso del Método IV, 1637.

7.1. Introducción

Este capítulo resume las conclusiones del trabajo de Maestría presentado en los capítulos precedentes. Conjuntamente, se presentan aquellas principales líneas de investigación que han dejado interesantes aspectos abiertos que merecen un estudio o una profundización en el futuro.

7.2. Conclusiones

Tomando en cuenta los objetivos planteados en el trabajo de Maestría, una evaluación general permite concluir que se cumplieron exitosamente las principales metas fijadas.

Como primer aspecto debe indicarse que se estudiaron en profundidad los fundamentos de las técnicas de computación evolutiva de desde un punto de vista general y el paradigma de los algoritmos genéticos en particular. En esta etapa se adquirieron valiosos conocimientos sobre esta familia de metaheurísticas que han permitido una formación adecuada al responsable del proyecto y adicionalmente iniciar una línea de trabajo e investigación en el área de algoritmos evolutivos y paralelismo en el Instituto de Computación de la Facultad de Ingeniería. Asimismo, durante el desarrollo del proyecto se realizaron importantes contactos con otros grupos de trabajo e investigación en las áreas de metaheurísticas, algoritmos evolutivos y paralelismo que permitieron un valioso intercambio de opiniones e ideas sobre la aplicabilidad de las técnicas de computación evolutiva para la resolución de problemas relacionados con el diseño de redes de comunicaciones confiables.

En el marco del trabajo se realizó un relevamiento de los métodos de aplicación de las técnicas de procesamiento paralelo y distribuido a los algoritmos evolutivos. En este sentido, se diseñó una completa reseña de propuestas realizadas por parte de los investigadores en el área con el objetivo de mejorar el desempeño computacional y la calidad de los resultados de los algoritmos genéticos. La visión de la evolución histórica presentada, que resume los detalles principales de los artículos, textos, taxonomías y otras reseñas disponibles, reúne los modelos más relevantes de algoritmos genéticos paralelos y los enmarca dentro de una propuesta de taxonomía unificada. Este relevamiento fue publicado bajo el título "Evolución en el diseño y la clasificación de Algoritmos Genéticos Paralelos", en la XXVIII Conferencia Latinoamericana de Informática (Nesmachnow, 2002b), y merece destacarse como un valioso aporte del proyecto.

En el aspecto relacionado con el área de aplicación, se estudió el Problema de Steiner Generalizado, un problema de optimización combinatoria utilizado para modelar el diseño de redes de comunicaciones de alta confiabilidad topológica. La clase de problemas de Steiner ha sido escasamente estudiada en sus formulaciones generalizadas fuera de nuestro entorno de trabajo, y el análisis de propuestas de técnicas evolutivas constituye otro novedoso y relevante aporte del trabajo desarrollado en el marco de la Maestría.

En una primera aproximación a la resolución del problema estudiado, se presentó el diseño y la implementación de un algoritmo genético paralelo específico. De acuerdo al relevamiento realizado, constituye la primer propuesta de resolución del caso generalizado del problema mediante el uso de un algoritmo evolutivo. La estrategia de resolución del problema se basó en el uso de una representación binaria simple, y se trabajó siempre bajo la idea de mantener la formulación y los operadores evolutivos tan simples como fuera posible, sin agregar información específica del problema. Utilizando este enfoque, se realizaron estudios de validación y configuración del algoritmo diseñado, comparando los resultados con resultados obtenidos utilizando otras metodologías de resolución. Los resultados fueron publicados bajo el título "Resolución del Problema de Steiner Generalizado utilizando un Algoritmo Genético Paralelo" en el Segundo Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (Árraga et al, 2002).

Como parte del análisis experimental se diseñó un conjunto de instancias del Problema de Steiner Generalizado par utilizar en los experimentos de evaluación de resultados. Este conjunto de problemas (generados aleatoriamente) constituye el único conjunto conocido diseñado específicamente para evaluar los resultados de algoritmos de resolución del caso generalizado del problema considerado Si bien la dimensión del conjunto de instancias de prueba es reducida (solamente incluye a tres problemas) puede considerarse como un punto de partida para el diseño de nuevas instancias que permitan abarcar un mayor espectro de complejidad.

Sobre el conjunto de problemas de prueba diseñado, se realizó un estudio de los resultados obtenidos por los modelos serial y paralelo del algoritmo genético paralelo específico diseñado para la resolución de problema.

Complementariamente, se diseñaron versiones seriales y paralelas de otros algoritmos evolutivos y de búsqueda local, puros e híbridos, aplicados a la resolución del problema, manteniendo la idea de trabajar con una representación sencilla y operadores de complejidad reducida. Se realizó un exhaustivo análisis comparativo de la calidad de los resultados obtenidos desde el punto de vista numérico y un estudio de la eficiencia computacional al utilizar un cluster de ocho computadores para la ejecución distribuida de los modelos paralelos. Los resultados fueron publicados bajo el título "Técnicas Evolutivas Aplicadas al Diseño de Redes de Comunicaciones Confiables" en el Tercer Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados.

El análisis comparativo permitió detectar algoritmos evolutivos simples con un comportamiento muy promisorio para la resolución del Problema de Steiner Generalizado, tal como fue mencionado en las conclusiones del Capítulo 7. Aunque el estudio estuvo limitado por la disponibilidad de recursos computacionales existentes, se pudo comprobar las ventajas de utilizar modelos paralelos, tanto desde el punto de vista de la calidad de resultados obtenidos como desde el punto de vista de la mejora de eficiencia computacional alcanzada. El análisis empírico permitió identificar numerosos aspectos y características del problema abordado y del comportamiento de los diferentes algoritmos evolutivos considerados, que agregan un conocimiento importante para utilizar al momento de decidir una continuación del trabajo.

Simultáneamente al trabajo de experimentación, fue posible realizar una evaluación actualizada de la potencia de cálculo de la arquitectura paralela disponible en nuestro entorno de trabajo (cluster de PCs) para la resolución de problemas complejos y de gran porte.

7.3. Trabajo Futuro

A lo largo del trabajo de Maestría, numerosas líneas de investigación reportaron interesantes resultados o han dejado aspectos abiertos que merecen un estudio o una profundización como continuación lógica en el futuro inmediato.

Abordar un estudio teórico y experimental con el objetivo de ampliar el conjunto de problemas de prueba puede considerarse como el principal aspecto no relacionado directamente con los algoritmos de resolución del problema. Con este fin, deben estudiarse las características y las propiedades del problema para tratar de incorporar nuevas instancias, obtenidas de adaptar conjuntos ya existentes para problemas de complejidad reducida (como el problema del árbol de Steiner o los problemas de biconexión) y para diseñar instancias con valores óptimos conocidos que permitan abarcar una amplia gama en cuanto a la complejidad de los problemas planteados.

Los resultados obtenidos permitieron comprobar que existe un margen considerable para mejorar algunos de los algoritmos evolutivos propuestos. Tomando en cuenta que se identificó a la pérdida de diversidad como el principal motivo por el cual los algoritmos evolutivos no son capaces de alcanzar resultados de mejor calidad, el estudio de los diferentes mecanismos de introducción de diversidad se manifiesta como una tarea lógica para mejorar los resultados obtenidos. Conjuntamente, deberá estudiarse la influencia de los propios mecanismos de introducción de diversidad en el comportamiento de los algoritmos, tanto en lo referente al método de exploración del espacio de soluciones y la calidad de resultados obtenidos como el aspecto relacionado con su eficiencia computacional. En esta línea de trabajo, el estudio de modelos y representaciones más complejas, como las representaciones basadas en caminos, la utilización de soluciones no factibles complementadas con mecanismos de penalización de fitness y/o corrección de soluciones, así como la introducción de información dependiente del problema y operadores especializados constituyen líneas claras de trabajo a abordar en el futuro inmediato.

Ciertos aspectos relacionados con los modelos de paralelismo fueron poco analizados como consecuencia del escaso número de recursos computacionales disponibles, que limitó el alcance del trabajo en esta área de investigación. En caso de poder acceder a plataformas paralelas de alto poder computacional, el estudio del modelo de migración podrá profundizarse adecuadamente, para determinar sus contribuciones a mejorar la calidad de resultados del problema.

El estudio de otros modelos de paralelismo aplicados a algoritmos evolutivos para la resolución del Problema de Steiner Generalizado ha sido planificado como una tarea a desarrollar en el futuro. Durante el desarrollo del trabajo se decidió postergar el estudio de un modelo celular aplicado al problema, tomando en cuenta la falta de disponibilidad de una arquitectura paralela de memoria compartida durante buena parte del tiempo de desarrollo del trabajo. Sobre el final del trabajo, y evaluando en especial los buenos resultados obtenidos con el modelo de subpoblaciones con migración, se postergó definitivamente el diseño de modelos de algoritmos genéticos celulares. En la actualidad se dispone de un equipo multiprocesador en la Facultad de Ingeniería y el diseño de modelos de algoritmos evolutivos celulares surge como una línea clara a abordar en el futuro con el objetivo de estudiar un nuevo modelo de paralelismo y sus características, comparando con los resultados obtenidos para el modelo de subpoblaciones con migración.

Por último, la extensión de las ideas aplicadas en el trabajo a otros problemas de optimización vinculados con el diseño de redes de comunicaciones confiables constituye en si misma una interesante línea de trabajo que se plantea como probable origen de nuevos proyectos en el futuro cercano.

ANEXOS

ANEXO I

PROBLEMAS DE VALIDACIÓN

"What? Will the line stretch out to the crack of doom?."

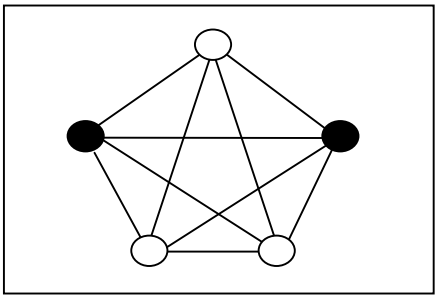
WILLIAM SHAKESPEARE
Macbeth, 1606

AI.1. Introducción

Este anexo presenta el conjunto de instancias del Problema de Steiner Generalizado utilizado para la validación del algoritmo genético paralelo presentado en el Capítulo 5. Se presentan las características de los problemas de prueba y las soluciones encontradas por el algoritmo genético, que corresponden con las soluciones óptimas de cada uno de los problemas considerados. Los problemas de validación fueron tomados del trabajo de Robledo (2001).

AI.2. Problema 1

El primer problema considerado está dado por el *grafo_v1* que se presenta en la Figura A1.1. La instancia se compone de diez aristas y cinco nodos, de los cuales dos son nodos terminales. Los costos de las aristas son uniformes e unitarios. La matriz de requerimientos de conectividad exige la existencia de tres caminos disjuntos entre los dos nodos terminales del problema.



Todas las aristas tienen costo 1.

$$R = \begin{pmatrix} 0 & 3 \\ 3 & 0 \end{pmatrix}$$

Figura A1.1: Instancia de validación grafo_v1

La solución encontrada por el algoritmo genético se presenta en la Figura A1.2. La solución incluye dos de los tres nodos de Steiner y cinco de las diez aristas originales del problema. El costo total de la solución hallada tiene valor 5.

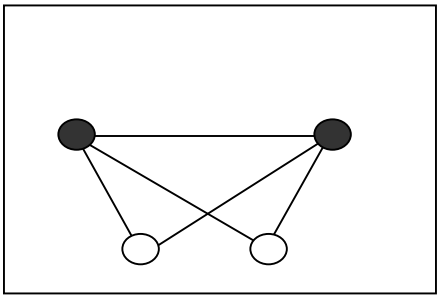
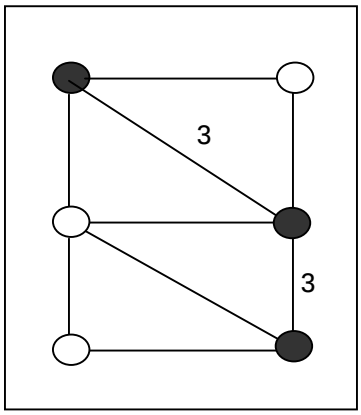


Figura A1.2: Solución encontrada por el algoritmo genético para la instancia de validación grafo_v1.

AI.3. Problema 2

El segundo problema considerado está dado por el grafo_v2 que se presenta en la Figura A1.3. La instancia se compone de nueve aristas y seis nodos, de los cuales tres son nodos terminales. Los costos de las aristas son uniformes e unitarios, salvo las dos aristas marcadas, que tienen un costo con valor 3. La matriz de requerimientos de conectividad exige la existencia de dos caminos disjuntos entre cada par de nodos terminales del problema (corresponde a un problema de biconexión).



Las aristas cuyo costo no se especifica tienen un valor de costo igual a 1.

$$R = \begin{pmatrix} 0 & 2 & 2 \\ 2 & 0 & 2 \\ 2 & 2 & 0 \end{pmatrix}$$

Figura A1.3: Instancia de validación grafo_v2

La solución encontrada por el algoritmo genético se presenta en la Figura A1.4. La solución incluye dos de los tres nodos de Steiner y cinco de las diez aristas originales del problema. El costo total de la solución hallada tiene valor 7.

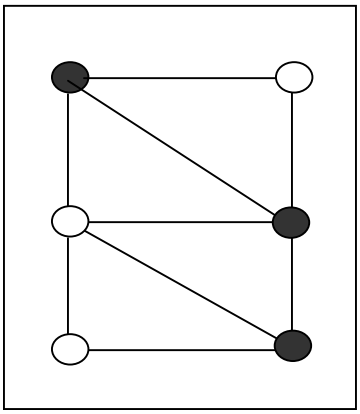
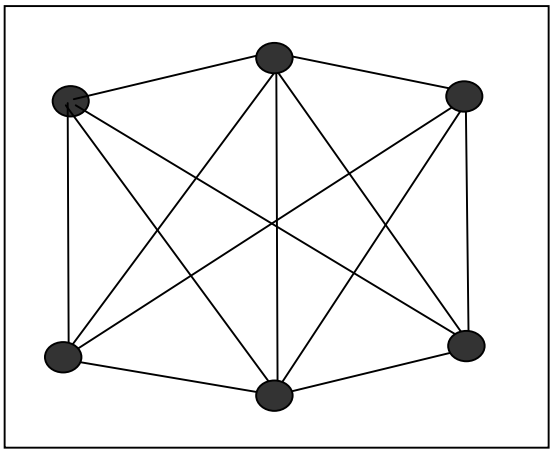


Figura A1.2: Solución encontrada por el algoritmo genético para la instancia de validación grafo_v2.

AI.4. Problema 3

El tercer problema considerado está dado por el grafo_v3 que se presenta en la Figura A1.5. La instancia se compone de doce aristas y seis nodos, todos ellos considerados como nodos terminales. Los costos de las aristas son uniformes e unitarios. La matriz de requerimientos de conectividad exige la existencia de tres caminos disjuntos entre cada par de nodos terminales del problema (corresponde a un problema de 3 conexión).



Todas las aristas tienen costo 1

$$R = \begin{pmatrix} 0 & 3 & 3 & 3 & 3 & 3 \\ 3 & 0 & 3 & 3 & 3 & 3 \\ 3 & 3 & 0 & 3 & 3 & 3 \\ 3 & 3 & 3 & 0 & 3 & 3 \\ 3 & 3 & 3 & 3 & 0 & 3 \\ 3 & 3 & 3 & 3 & 3 & 0 \end{pmatrix}$$

Figura A1.1: Instancia de validación grafo_v3

La solución encontrada por el algoritmo genético se presenta en la Figura A1.6. La solución incluye nueve de las doce aristas originales del problema. El costo total de la solución hallada tiene valor 9.

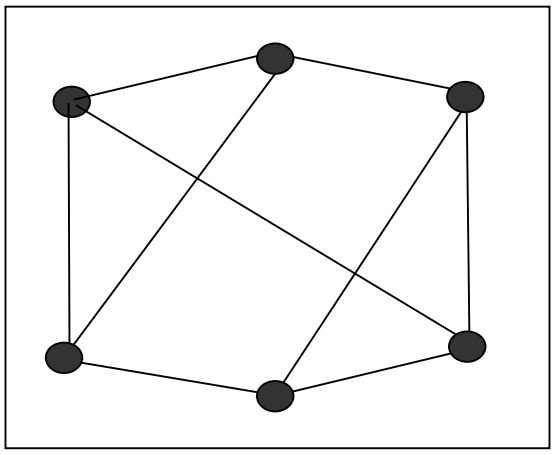
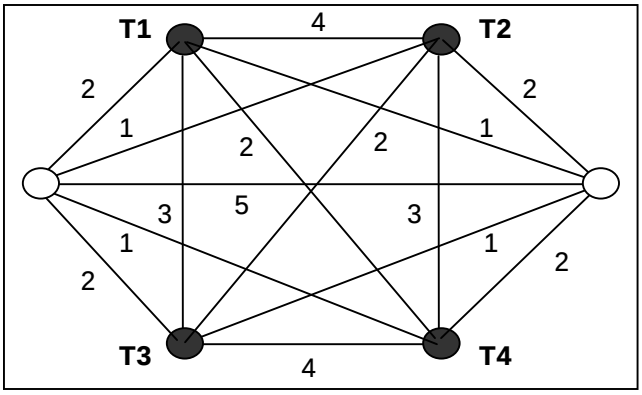


Figura A1.2: Solución encontrada por el algoritmo genético para la instancia de validación grafo_v3

AI.5. Problema 4

El cuarto problema considerado está dado por el grafo_v4 que se presenta en la Figura A1.7. La instancia se compone de quince aristas y seis nodos, cuatro de ellos considerados como nodos terminales. Los costos de las aristas son valores enteros variables entre 1 y 5. La matriz de requerimientos de conectividad exige la existencia de un número variable de caminos disjuntos entre cada par de nodos terminales del problema (entre dos y cuatros caminos).



$$R = \begin{pmatrix} 0 & 2 & 3 & 4 \\ 2 & 0 & 2 & 3 \\ 3 & 2 & 0 & 2 \\ 4 & 3 & 2 & 0 \end{pmatrix}$$

Figura A1.1: Instancia de validación grafo_v4

La solución encontrada por el algoritmo genético se presenta en la Figura A1.8. La solución incluye a los dos nodos de Steiner y a diez de las quince aristas originales del problema. El costo total de la solución hallada tiene valor 18.

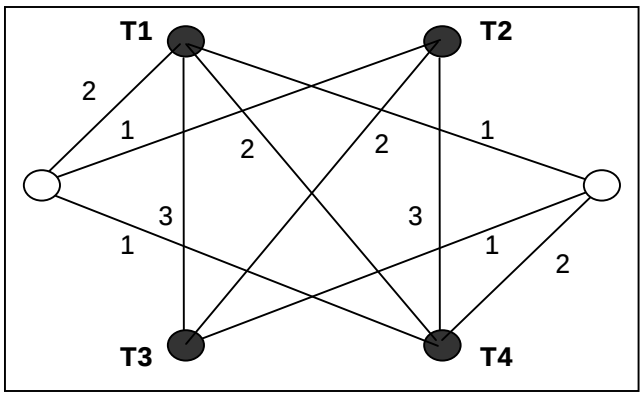
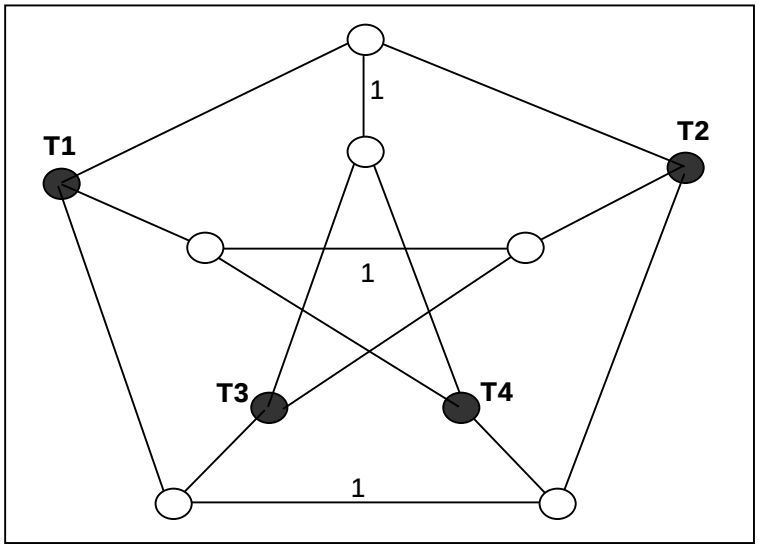


Figura A1.2: Solución encontrada por el algoritmo genético para la instancia de validación grafo_v4.

AI.6. Problema 5

El quinto problema considerado está dado por el grafo_v5 que se presenta en la Figura A1.9. La instancia se compone de quince aristas y diez nodos, de los cuales cuatro son considerados nodos terminales. Los costos de las aristas son uniformes de valor dos, salvo las tres aristas marcadas, que tienen costo de valor 1. La matriz de requerimientos de conectividad exige la existencia de un número variable de caminos disjuntos entre cada par de nodos terminales del problema (entre uno y tres caminos).



$$R = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 2 & 3 \\ 2 & 2 & 0 & 3 \\ 3 & 3 & 3 & 0 \end{pmatrix}$$

Figura A1.1: Instancia de validación grafo_v5

La solución encontrada por el algoritmo genético se presenta en la Figura A1.10. La solución incluye a los seis nodos de Steiner y a doce de las quince aristas originales del problema. El costo total de la solución hallada tiene valor 22. Puede apreciarse que la solución optima está compuesta por todas las aristas del grafo original, excepto aquellas tres que tienen costo unitario.

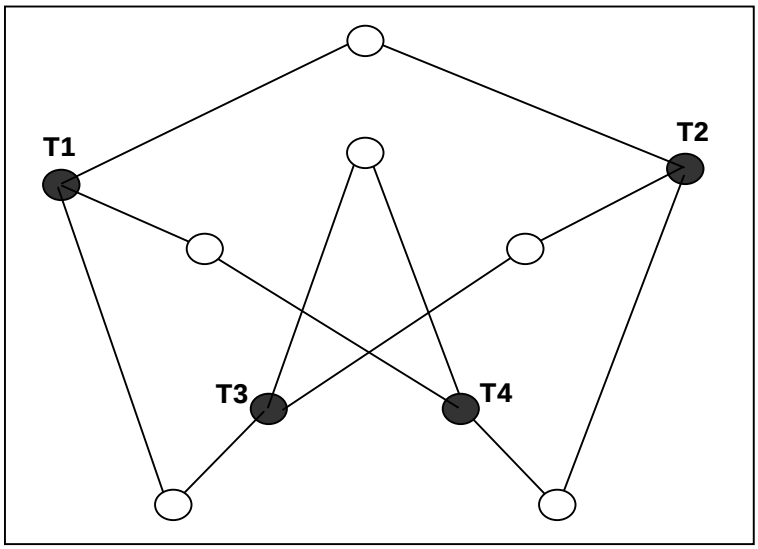
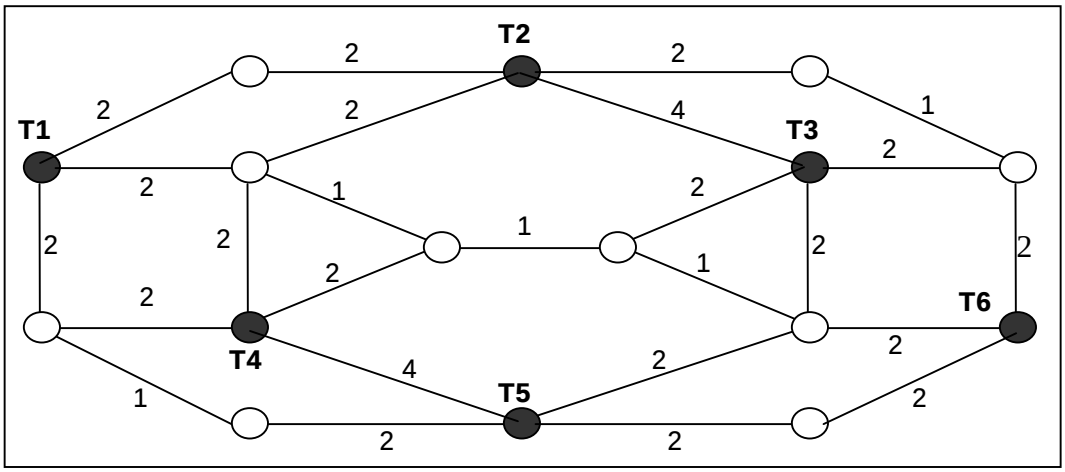


Figura A1.2: Solución encontrada por el algoritmo genético para la instancia de validación grafo_v4.

AI.7. Problema 6

El sexto problema considerado está dado por el grafo_v6 que se presenta en la Figura A1.11. La instancia se compone de veinticinco aristas y dieciséis nodos, de los cuales seis son considerados nodos terminales. Los costos de las aristas son valores enteros variables en el conjunto {1,2,4}. La matriz de requerimientos de conectividad exige la existencia de un número variable de caminos disjuntos entre cada par de nodos terminales del problema (dos o tres caminos).



$$R = \begin{pmatrix} 0 & 2 & 3 & 2 & 3 & 2 \\ 2 & 0 & 2 & 3 & 2 & 3 \\ 3 & 2 & 0 & 2 & 3 & 2 \\ 2 & 3 & 2 & 0 & 2 & 3 \\ 3 & 2 & 3 & 2 & 0 & 2 \\ 2 & 3 & 2 & 3 & 2 & 0 \end{pmatrix}$$

Figura A1.1: Instancia de validación grafo_v1

La solución encontrada por el algoritmo genético se presenta en la Figura A1.12. La solución incluye a ocho de los diez nodos de Steiner y a diecinueve de las veinticinco aristas originales del problema. El costo total de la solución hallada tiene valor 39.

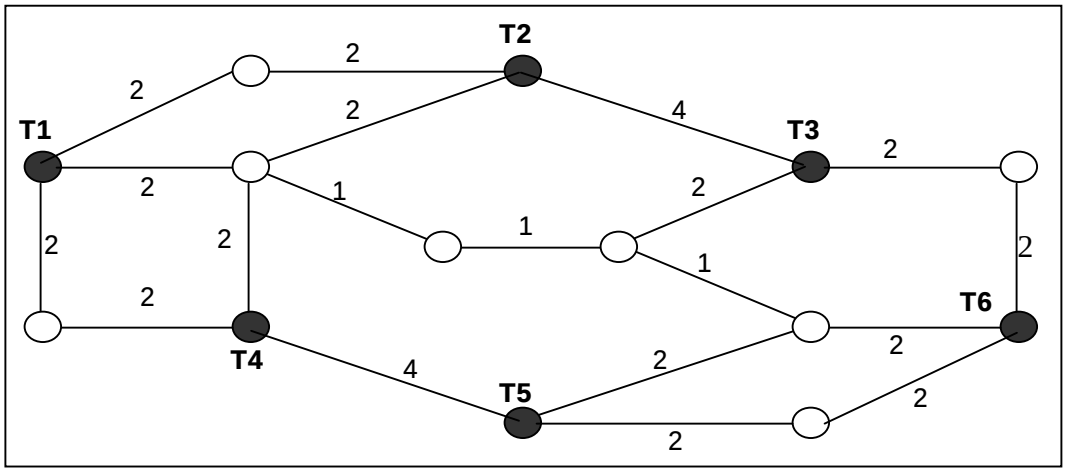
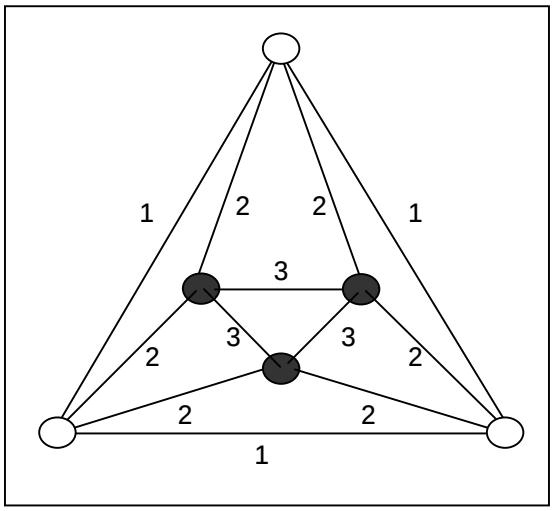


Figura A1.2: Solución encontrada por el algoritmo genético para la instancia de validación grafo_v6.

AI.8. Problema 7

El séptimo problema considerado está dado por el grafo_v7 que se presenta en la Figura A1.13. La instancia se compone de doce aristas y seis nodos, de los cuales tres son considerados nodos terminales. Los costos de las aristas son valores enteros variables entre 1 y 3. La matriz de requerimientos de conectividad exige la existencia de dos caminos disjuntos entre cada par de nodos terminales del problema (caso de problema de biconexión).



$$R = \begin{pmatrix} 0 & 2 & 2 \\ 2 & 0 & 2 \\ 2 & 2 & 0 \end{pmatrix}$$

Figura A1.13: Instancia de validación grafo_v7

La solución encontrada por el algoritmo genético se presenta en la Figura A1.14. La solución no incluye ninguno de los tres nodos de Steiner del problema. Solamente se incluyen los tres nodos terminales y las tres aristas que los unen, entre las doce aristas originales del problema. El costo total de la solución hallada tiene valor 9. Puede apreciarse que la solución optima está compuesta por aquellas aristas del grafo original que tienen el mayor valor de costo utilizado (valor 3).

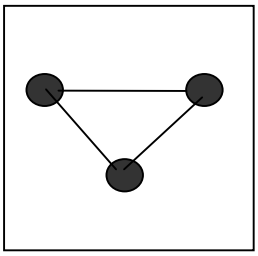


Figura A1.14: Solución encontrada por el algoritmo genético para la instancia de validación grafo_v7.

ANEXO II

INSTANCIAS DE PRUEBA DEL PROBLEMA DE STEINER GENERALIZADO

"Pluralitas non est ponenda sine neccesitate."
(*" Las entidades no deben multiplicarse innecesariamente."*)
WILLIAM DE OCCAM, siglo XIV.

AII.1. Introducción

Esta sección presenta el conjunto de instancias de prueba del Problema de Steiner Generalizado diseñado para la evaluación de los algoritmos implementados en el marco de la Tesis.

Tal como se mencionó en los Capítulos 4 y 5, no existen referencias bibliográficas sobre la aplicación de técnicas heurísticas al caso generalizado del Problema de Steiner. Por este motivo no se han diseñado conjuntos estandarizados de instancias de prueba para evaluar comparativamente los resultados de diferentes algoritmos o métodos de resolución del problema. En general, los investigadores que han abordado el problema generalizado han trabajado con instancias de prueba derivadas de los casos más simples, como el caso del problema del árbol de Steiner, o han generado aleatoriamente sus propios casos de prueba.

Tomando en cuenta las anteriores observaciones, se optó por diseñar un conjunto de instancias de prueba aleatorias, compuesto de tres problemas, para la evaluación de calidad de resultados y desempeño computacional de los algoritmos implementados.

A continuación se presenta el formato utilizado para almacenar los grafos, las características de las instancias de prueba y una descripción del software utilizado para la generación de los problemas.

AII.2. Formato de los grafos

Los grafos que determinan los problemas de prueba se almacenan en archivos de texto, y están representados por el formato que se especifica en la tabla de la Figura A2.1.

```
vertice <id-vertice>
vertice <id-vertice>
.......................
vertice <id-vertice>
arista <id-arista> <id-vertice> <id-vertice> <costo>
arista <id-arista> <id-vertice> <id-vertice> <costo>
.............................................
arista <id-arista> <id-vertice> <id-vertice> <costo>
restriccion <número> <id-vertice> <id-vertice>
restriccion <número> <id-vertice> <id-vertice>
.............................................
restriccion <número> <id-vertice> <id-vertice>
```

Figura A2.1: Formato de los grafos de prueba.

En la Figura A2.1, `<id-vertice>` e `<id-arista>` son palabras que identifican a los nodos y a las aristas del grafo respectivamente, mientras que `<número>` es un entero que representa la cantidad de caminos impuesta como restricción entre el par de nodos terminales indicado y `<costo>` es un real que indica el costo asociado a la arista

Los nodos terminales del grafo no se indican explícitamente, sino que quedan determinados de modo implícito como aquellos nodos para los cuales existen requerimientos de conexión especificados.

AII.3. Características de los casos de prueba

Los tres grafos del conjunto de prueba creado se designan por un nombre que referencia a la cantidad total de nodos y el número de nodos terminales. Por ejemplo, *grafo 50–15* designa al grafo más pequeño del conjunto de problemas de prueba, que tiene 50 nodos de los cuales 15 son terminales, y así sucesivamente para las restantes instancias.

Las tres instancias fueron generadas seleccionando aleatoriamente topologías de grafos, considerado los nodos como puntos geográficos en un plano euclídeo. Los costos asociados a los enlaces fueron considerados proporcionales a las distancias euclideanas entre nodos para las instancias *grafo 75–25* y *grafo 50–15*, y fueron seleccionados aleatoriamente entre los valores enteros en el intervalo [1, …, 20] para la instancia *grafo 100–10*. Los requerimientos de conexión para cada par de nodos terminales fueron seleccionados aleatoriamente de modo uniforme entre los valores enteros en el intervalo [0, …, 4] para las tres instancias consideradas.

Los detalles de las tres instancias de prueba diseñadas para el Problema de Steiner Generalizado se presentan en la tabla de la Figura A2.1, indicando el número total de nodos, número de nodos terminales, número de aristas y el grado de conectividad promedio (cociente entre el número de aristas presentes en el grafo sobre el número de aristas del grafo completo, expresado en aristas por nodo). Se incluyen además los valores de costo total (C_{ORIG}) para cada grafo, correspondiente al caso en que la totalidad del conjunto de aristas del grafo original se incluye en la solución. Los casos de prueba diseñados pueden considerarse "representativos" para redes de comunicaciones de mediano tamaño con requisitos de conexión variables.

	grafo 100-10	*grafo 75-25*	*grafo 50-15*
Nodos	100	75	50
Terminales	10	25	15
Aristas	500	360	249
Grado Conectividad Promedio	0.1	0.13	0.2
Costo total (C_{ORIG})	4925.00	6294.93	10949.98

Figura A2.1: Características de los grafos de prueba para el Problema de Steiner Generalizado.

AII.4. Generador aleatorio de grafos

El conjunto de instancias de prueba del Problema de Steiner Generalizado fue generado mediante un programa diseñado específicamente con tal fin. El programa generador permite crear instancias aleatorias del Problema de Steiner Generalizado recibiendo como entrada los parámetros que definen al problema, que se presentan en la tabla de la Figura A2.2.

Parámetro	*Comentario*
Número de nodos.	Número total de nodos del grafo.
Número de terminales.	Número de nodos terminales del grafo.
Número de aristas.	Número de aristas del grafo.
Máximo costo.	Máximo valor de costo de arista.
Máximo requerimiento.	Máximo valor de requerimiento de caminos.
Máximo valor de coordenada *x*. Máximo valor de coordenada *y*.	Límites de un cuadrado en el cual se suponen ubicados geográficamente los nodos del problema
Opción.	1 – Costo de arista proporcional a distancia euclídea entre nodos. Probabilidad de inclusión de una arista independiente de su costo. 2 – Costo de arista proporcional a distancia euclídea entre nodos. Probabilidad de inclusión de una arista inversamente proporcional a su costo. 3 – Costo de arista aleatorio en (0, Máximo costo). Probabilidad de inclusión de una arista independiente de su costo.
Randconfig.	Archivo de configuración para el generador de números aleatorios utilizado.

Figura A2.2: Parámetros del programa generador de instancias de prueba para el Problema de Steiner Generalizado.

El programa generador crea aleatoriamente topologías de grafos, considerado a los nodos del problema como puntos geográficos en una región del plano euclídeo que corresponde a un cuadrado limitado por las coordenadas (0, Máximo valor de coordenada *x*) y (0, Máximo valor de coordenada *y*).

La existencia de una arista entre un par de nodos se determina de modo probabilístico. Dependiendo de la opción elegida, el costo de esta arista será proporcional a la distancia euclideana entre los nodos que conecta, o se seleccionará aleatoriamente de manera uniforme en el intervalo (0, Máximo costo), y la probabilidad de inclusión de la arista entre un par de nodos puede ser dependiente o no de su valor de costo asociado.

Por último, los requerimientos de conexión para cada par de nodos terminales se seleccionan aleatoriamente de modo uniforme entre los valores enteros en el intervalo [0, …, Máximo requerimiento].

Un seudocódigo del programa generador se presenta en la Figura A2.3.

```
Parámetros de Entrada
numNodos, numTerm, numEdges, maxCost, maxR: enteros
MaxX, maxY: reales
Opcion: 1, 2, 3
Randconfig: string

Costototal = 0.0
NR = Crear Generador Aleatorio(randconfig)
CompEdges = numNodos*(numNodos-1)/2
G = Crear Grafo

// Agregar nodos
para i = 0 ... numNodos-1
   // Sortear coordenadas de nodos
   posx = NR->NrandFloatRank(0,maxX)
   posy=NR->NrandFloatRank(0,maxY)
   G.agregarnodo(i,posx,posy)
   si (opcion = 2)
       // Probabilidad de arista inversamente proporcional al costo
       // Calcular distancias a nodos anteriores
       para j = 0 ... i-1
           costototal += 1.0/distancia(i,j)
   fin
   fin
fin

// Agregar aristas
costototal=1.00/costototal
cantEdges=0
si (opcion = 1) || (opcion = 3)
   // Probabilidad de arista independiente del costo
   probEdge = (2*numEdges)/(numNodos*(numNodos-1))
fin
para i = 0 ... numNodos-1
   para j = i+1 ... numNodos-1
   dist = distancia(u,v);
   si (opcion = 2)
           probEdge = numEdges*costototal/dist;
   fin
   si ( probEdge > NR->uni() ) {
           g.AgregarArista(u,v,dist);
           cantEdges++;
   fin
   fin
fin

// Determinar nodos terminales
para termdef = 0 ... numTerm-1
   Sortear terminal (0, NumNodos)
   Marcar nodo terminal
fin
para i=0 ... numNodos-1
   para j=0 ... i-1
       si (esTerminal(j)) {
           numCaminos = Sortear Entero (0, MaxR)
           AgregarRestriccion(i, numCaminos);
           AgregarRestriccion(j, numCaminos);
   fin
   fin
fin
porcEdges = cantEdges*1.00/CompEdges;
```

Figura A2.3: Seudocódigo del programa generador de instancias.

Luego de su ejecución el programa generador presenta un resumen de la instancia generada, tal como el que se presenta en la Figura A2.4.

```
GENERADOR de CASOS de PRUEBA para el PROBLEMA de STEINER GENERALIZADO
RESUMEN

Grafo generado con numNodos nodos (termdef terminales)
porcEdges=cantEdges*1.00/CompEdges;
Numero de aristas : cantEdges (porcEdges sobre el grafo completo)
Nodos ubicados entre coordenadas (0,maxX) y (0,maxY)
Requerimientos entre 0 y maxR.
caso 1:
  Opcion 1 - Costo de arista proporcional a distancia euclídea entre
  nodos. Probabilidad de inclusión de una arista independiente de su
  costo.
caso 2:
  Opcion 2 - Costo de arista proporcional a distancia euclídea entre
  nodos. Probabilidad de inclusión de una arista inversamente
  proporcional a su costo.
caso 3:
  Opcion 3 - Costo de arista aleatorio en (0, Máximo costo).
  Probabilidad de inclusión de una arista independiente de su costo.
```

Figura A2.4: Reporte del programa generador de instancias.

AII.5. Acceso al software

El conjunto de problemas de prueba diseñado, conjuntamente con la descripción del formato utilizado en la notación que se presentó en la sección AII.2 de este anexo y el programa generador aleatorio de grafos utilizado se encuentran disponibles públicamente en la URL correspondiente al Grupo de Trabajo en Procesamiento Paralelo y sus Aplicaciones del Centro de Cálculo, Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Uruguay: http://www.fing.edu.uy/inco/grupos/cecal/hpc/gsp.

ANEXO III

ESTUDIO EMPÍRICO DE OPERADORES DE CRUZAMIENTO EN UN ALGORITMO GENÉTICO APLICADO AL PROBLEMA DE STEINER GENERALIZADO

"The majority of the new varieties which have shown the ability to expand are a result of crossing-phenomena and not of mutations
…..…
mutation and selection alone, however, proved insufficient to explain evolutionary phenomena."

NILS BARRICELLI
Numerical Testing of Evolution Theories: Part I, 1962

AIII.1. Introducción

Este anexo presenta el estudio realizado sobre la utilización de diferentes familias de operadores de cruzamiento en un algoritmo genético para la resolución del Problema de Steiner Generalizado. Se presentan y comparan los resultados obtenidos utilizando operadores de cruzamiento de N puntos y variantes paramétricas del operador de cruzamiento uniforme en la resolución de los problemas que componen el conjunto de instancias de prueba presentado en el Capítulo 5.

El artículo que se presenta a continuación fue escrito conjuntamente con Martín Pedemonte y publicado en el Workshop de Agentes y Sistemas Inteligentes en el marco del IX Congreso Argentino de Ciencias de la Computación (CACIC 2003) desarrollado en La Plata, Argentina en octubre de 2003.

Estudio Empírico de Operadores de Cruzamiento en un Algoritmo Genético Aplicado al Problema de Steiner Generalizado

Martín Pedemonte
CeCal, InCo
Facultad de Ingeniería
Universidad de la República
Uruguay
mpedemon@fing.edu.uy

Sergio Nesmachnow
CeCal, InCo
Facultad de Ingeniería
Universidad de la República
Uruguay
sergion@fing.edu.uy

Resumen

El Problema de Steiner Generalizado modela el diseño de redes de comunicaciones confiables en las cuales se exigen requisitos de conexión entre nodos distinguidos, que garanticen con alta probabilidad la comunicación entre sí. Es un problema NP difícil, para el cual pocos algoritmos han sido propuestos.

Los algoritmos evolutivos se han utilizado como metaheurísticas alternativas a los métodos exactos para resolver complejos problemas de optimización. Siguiendo este enfoque, hemos propuesto trabajar con algoritmos genéticos para la resolución del Problema de Steiner Generalizado.

La calidad de las soluciones obtenidas al utilizar un algoritmo genético depende de múltiples factores, entre los que se pueden destacar los operadores de recombinación y mutación utilizados. El objetivo de este trabajo es estudiar desde un punto de vista empírico la influencia de diferentes operadores de cruzamiento en un algoritmo genético para la resolución del Problema de Steiner Generalizado. Se presentan y comparan los resultados obtenidos utilizando cruzamientos de N puntos y variantes paramétricas del cruzamiento uniforme.

Palabras clave: Algoritmos genéticos, problema de Steiner generalizado, operadores de cruzamiento.

Destinado al Workshop de Agentes y Sistemas Inteligentes

1. Introducción

Uno de los principales problemas de construcción de redes de comunicaciones plantea el diseño de una topología de interconexión de sus nodos de modo que la red verifique ciertas características de confiabilidad. La confiabilidad de una red es una medida que evalúa la probabilidad de éxito en la comunicación entre pares de nodos.
El continuo crecimiento en el tamaño de los problemas de diseño de redes de comunicaciones ha conducido a proponer alternativas a los enfoques exactos tradicionales para resolver instancias complejas. En este contexto, varias heurísticas se han aplicado para obtener soluciones aproximadas de buena calidad en tiempos razonables. Entre ellas, los algoritmos genéticos (AG) se han manifestado como métodos flexibles y robustos para la solución de los complejos problemas de optimización subyacentes al diseño de redes de comunicaciones.

Dada una red de comunicaciones con nodos distinguidos denominados terminales, el Problema de Steiner Generalizado (GSP, por sus siglas en inglés) refiere al diseño de una subred de mínimo costo que verifique requerimientos prefijados de conexión entre pares de nodos distinguidos denominados terminales.
La minimización del costo total de una red de comunicaciones y la maximización de su confiabilidad son objetivos contrapuestos. Por ello, un modelo que minimice el costo total de la red satisfaciendo los requisitos de conexión sin agregar redundancia de caminos, constituye una solución muy sensible a fallas en los nodos o en los enlaces que afectarían la operatividad de la red.
El GSP incorpora requisitos adicionales de conectividad sobre pares de nodos terminales, aplicándose al diseño de redes de comunicaciones en las cuales la alta confiabilidad queda garantizada por la existencia de caminos alternativos entre nodos terminales. Esta redundancia otorga a la red un funcionamiento más robusto, siendo más resistente a fallas en sus componentes.

Tradicionalmente, los AG que utilizan codificación binaria simple han trabajado con los operadores de cruzamiento de uno y dos puntos, sin prestar especial atención a operadores que utilizan mayor número de puntos de corte. En nuestro entorno de trabajo hemos utilizado el cruzamiento de un punto en un AG para resolver el GSP [2], y en experimentos posteriores hemos utilizado cruzamiento de dos puntos y uniforme alcanzando mejoras en la calidad de los resultados obtenidos.
La propuesta de este trabajo consiste en estudiar desde un punto de vista empírico la influencia de diferentes operadores de cruzamiento en un AG para la resolución del GSP, de modo de determinar el operador más adecuado para el problema.
El resto del artículo se organiza del modo que se detalla a continuación. La sección 2 presenta el problema estudiado y su modelo matemático. La sección 3 describe el algoritmo genético utilizado y la codificación del problema. La sección 4 explica los operadores estudiados y su diseño. Los resultados experimentales sobre las instancias de prueba consideradas y las comparaciones entre variantes se presentan en la sección 5. Por último, la sección 6 ofrece las conclusiones y propuestas de trabajo futuro.

2. El Problema de Steiner Generalizado

Esta sección presenta la formulación y el modelo matemático del GSP, describe variantes del problema y concluye con un breve comentario sobre trabajos relacionados con la aplicación de técnicas evolutivas para la resolución del problema.

2.1 - Formulación y modelo matemático del GSP

La siguiente formulación del GSP se basa en el compendio de Kahn y Crescenzi [7]. Sean los siguientes elementos:

- Un grafo no dirigido $G = (V,E)$, siendo V el conjunto de nodos y E el conjunto de aristas, y con una matriz de costos C asociados a las aristas.
- Un conjunto distinguido de nodos T llamados ***terminales***, de cardinalidad $n_T = |T|$, tal que $2 \leq n_T \leq n_V$, siendo $n_V = |V|$ la cardinalidad del conjunto de vértices del grafo G.
- Una matriz $R=\{r_{ij}\}$ con $i,j \in T$, de dimensión $n_T \times n_T$, cuyos elementos son enteros positivos que indican los requerimientos de conectividad –cantidad de caminos disjuntos– que se exigen entre todo par de nodos terminales.

El problema GSP propone encontrar un subgrafo G_T de G, de costo mínimo, tal que todo par de nodos $i,j \in T$, $i \neq j$ sean r_{ij} arista conexos en G_T, es decir que existan r_{ij} caminos disjuntos, que no comparten aristas, entre los nodos i y j en el grafo G_T.
Sobre los nodos no pertenecientes al conjunto de nodos terminales no se plantean requisitos de conectividad. Estos nodos, llamados ***nodos de Steiner***, pueden o no formar parte de la solución óptima, de acuerdo a la conveniencia de utilizarlos.

Un modelo matemático del problema GSP arista conexo define para cada arista $(i,j) \in E$ una variable binaria x_{ij} y una variable real y_{ij}^{kl} que representa la utilidad de la arista (i,j) en la dirección de i hacia j en un camino que une a los nodos terminales k y l.
El Cuadro 1 presenta el modelo matemático del GSP como problema de programación lineal entera, de acuerdo al trabajo de Robledo [9].

$$Min \sum_{(i,j)\in E} C_{ij} \cdot x_{ij} \quad \text{sujeto a}$$

$$x_{ij} \geq y_{ij}^{kl} + y_{ji}^{kl} \qquad \forall (i,j) \in E, \forall k,l \in T, k \neq l$$

$$\sum_{(k,j)\in E} y_{kj}^{kl} \geq r_{kl} \qquad \forall k,l \in T, k \neq l$$

$$\sum_{(p,j)\in E} y_{pj}^{kl} - \sum_{(i,p)\in E} y_{ip}^{kl} \geq 0 \quad \forall k,l \in T, \forall p \in V \setminus \{k,l\}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in E$$

$$y_{ij}^{kl} \geq 0 \qquad \forall i,j : (i,j) \in E, \forall k,l \in T, k \neq l$$

Cuadro 1: Modelo matemático del GSP como problema de programación lineal entera

Llamando $U=\{u_{ij}\}$ a una solución óptima del problema, el conjunto de aristas determinado por $\{(i,j) \in E / u_{ij}=1\}$ define un subgrafo solución del problema.

2.2 - Variantes y complejidad del problema

La complejidad del problema de Steiner está dada por la generalidad de su planteo, que exige requisitos variables de conexión entre pares de nodos terminales. Ciertas variantes del problema simplifican este requerimiento: los ***problemas de k conexión*** exigen una cantidad fija k de caminos disjuntos entre pares de nodos terminales, mientras que el ***problema del árbol de Steiner*** sólo exige la existencia de un camino entre pares de nodos terminales.

El GSP forma parte de la clase de problemas NP difíciles [7]. Karp probó en 1972 que el propio problema del árbol de Steiner, que plantea la restricción menos general de caminos, es NP completo [8]. La complejidad del problema de Steiner lo hace cada vez menos tratable al aumentar el tamaño del grafo involucrado. El uso de algoritmos exactos queda limitado a pequeñas instancias de complejidad reducida, como consecuencia de los elevados tiempos de ejecución necesarios para su resolución. Por tal motivo se han propuesto soluciones alternativas utilizando heurísticas que permitan encontrar soluciones de buena calidad en tiempos razonables.

2.3 - Técnicas evolutivas aplicadas a la resolución del problema de Steiner

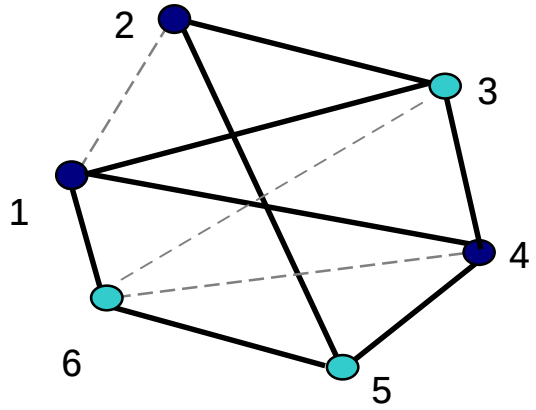
No existen antecedentes sobre la resolución del GSP mediante técnicas evolutivas. Existen trabajos sobre AG aplicados al problema del árbol de Steiner, pero fuera de nuestro entorno de trabajo no hay referencias de aplicación de técnicas evolutivas al problema generalizado. El enfoque de este trabajo corresponde al presentado en un artículo precedente [2] donde se ha seguido una propuesta de Esbensen [4] para el problema del árbol de Steiner, imponiendo el trabajo con soluciones factibles.

3. Implementación mediante un Algoritmo Genético

Esta sección presenta los detalles de codificación e implementación del AG utilizado para la resolución del GSP. Los operadores de cruzamiento constituyen la parte fundamental de nuestro estudio y son presentados en detalle en la sección siguiente.

3.1 - Codificación y operadores

El algoritmo genético implementado se basa en una codificación binaria sencilla. Cada cromosoma consta de un arreglo de bits que representan a las aristas del grafo original, numeradas de 0 a número de aristas - 1. La presencia o ausencia de una arista en un grafo solución queda determinada por el valor 1 o 0 respectivamente, en la posición correspondiente en el cromosoma. La Figura 1 presenta un ejemplo de la codificación binaria utilizada para un grafo pequeño, donde los nodos terminales se colorean en oscuro y los nodos de Steiner en claro. Las aristas marcadas en negro son aquellas presentes en el grafo representado y se notan con el valor 1 en el cromosoma de bits. Las aristas marcadas con la línea punteada gris están presentes en el grafo original pero no en el grafo solución representado, por lo cual se codifican con el valor 0 en el cromosoma. Las etiquetas de las posiciones del cromosoma referencian a los nodos origen y destino de cada arista del grafo original (**e**$_{25}$ = arista entre nodo 2 y nodo 5).



0	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**1**	**0**	**1**
e_{12}	e_{13}	e_{14}	e_{16}	e_{23}	e_{25}	e_{34}	e_{36}	e_{45}	e_{46}	e_{56}

Figura 1 – Ejemplo de la codificación binaria utilizada

Al utilizar la representación binaria los operadores de cruzamiento o mutación resultan sencillos, pero tienen el inconveniente de que los cromosomas resultantes pueden no representar a una solución factible del GSP. Por tal motivo, es necesario verificar la factibilidad de los nuevos cromosomas creados al aplicar un operador evolutivo.

En nuestra implementación hemos adoptado el criterio de descartar individuos no factibles. Esta decisión evita la tarea de cuantificar cuan lejos de una solución factible se encuentra un individuo no factible y determinar un valor adecuado de penalización para su fitness. Esta idea sigue la propuesta de Esbensen en su trabajo sobre el problema del árbol de Steiner [4] y ya ha sido utilizada en el trabajo previo [2].

Los operadores utilizados por el AG propuesto corresponden a:

- Selección por torneo entre tres individuos.
- Mutación de inversión del valor de un alelo, seleccionado probabilísticamente.
- Inicialización aleatoria de la población, mediante un procedimiento de eliminación de aristas a partir del grafo original.

Hemos trabajado con un criterio de parada de esfuerzo prefijado en 2000 generaciones, para realizar una comparación justa entre los resultados de los diferentes operadores de cruzamiento. Usar otros criterios, por ejemplo que analicen la evolución de fitness promedio, podría conducir a comparaciones injustas para operadores que introduzcan menor variabilidad entre generaciones pero evolucionen favorablemente a largo plazo.

3.2 - Función de fitness

La función de fitness utilizada toma en cuenta el costo del grafo representado por un cromosoma. Sustrayendo el valor de costo de un individuo del valor del costo del grafo original es posible mapear el problema de minimización del costo del grafo en el problema de maximización de la función presentada en la Ecuación 1.

$$f = C_{ORIG} - \sum_{0}^{|E|-1} \left[EDGE(i) * C(i) \right]$$

Ecuación 1: Función de fitness para el problema GSP.

En la Ecuación 1, $|E|$ denota la cardinalidad del conjunto de aristas del grafo original, correspondiente al largo de la codificación utilizada. C_{ORIG} representa el costo del grafo original, donde todas las aristas se consideran presentes. La función $C: [0,|E|-1] \rightarrow R$ devuelve el costo de una arista y la función $EDGE: [0,|E|-1] \rightarrow [0,1]$ retorna el valor binario correspondiente a la arista que ocupa la posición *i*-ésima en el cromosoma.

3.3 - Cálculo de factibilidad

La decisión de trabajar con individuos factibles simplifica el diseño de los operadores evolutivos, pero implica utilizar un procedimiento adicional para determinar la factibilidad de individuos luego de aplicar un operador. El cálculo de factibilidad verifica que el grafo representado cumpla las restricciones del GSP.

El algoritmo para cálculo de factibilidad consta de dos etapas:

- Una sencilla heurística se utiliza para detectar soluciones no factibles, verificando que los grados de los nodos terminales sean mayores o iguales al número máximo de caminos impuestos como restricción para cada nodo. La complejidad de esta heurística es de orden cuadrático en la cardinalidad del conjunto de nodos terminales y evita el cálculo de caminos entre pares de nodos terminales en todos los casos.
- Que un grafo pase el chequeo heurístico de grados de nodos terminales no implica que sea solución factible del GSP. En tal caso deben hallarse los caminos entre cada par de nodos terminales. Hemos utilizado una variante del algoritmo de Ford–Fulkerson [5] para cálculo de flujo máximo, que permite hallar caminos entre pares de nodos terminales asignando a un nodo el rol de fuente y al otro el rol de pozo. El algoritmo tradicional de Ford–Fulkerson trabaja sobre grafos dirigidos, por lo cual en el caso del grafo del GSP cada arista del grafo se considera como un par de aristas de sentidos opuestos. Asumiendo capacidad unitaria para las aristas, el flujo máximo entre fuente y pozo coincide con el número máximo de caminos disjuntos entre ellos. Si éste es menor que el valor del requerimiento correspondiente, el grafo es no factible. La variante de Ford–Fulkerson descrita utiliza una función de capacidad de flujo unitaria para cada arista, y tiene orden $O(|E|.r_{MAX}.n_T^2)$, siendo $|E|$ la cardinalidad del conjunto de aristas del grafo original, r_{MAX} el valor máximo del conjunto de requerimientos, y n_T la cardinalidad del conjunto de nodos terminales.

3.4 - Implementación y plataforma de ejecución

El algoritmo genético fue implementado sobre MALLBA [1], una biblioteca para optimización que implementa una gama de algoritmos evolutivos puros e híbridos en sus versiones secuenciales y distribuidas. MALLBA está implementada mediante clases C++ que abstraen a las entidades participantes en la resolución de problemas. Existen *Clases Provistas* que implementan los esqueletos de los métodos de resolución de manera independiente del problema específico a resolver y *Clases Requeridas* que contienen las entidades dependientes del problema y deben ser instanciadas por el usuario. La propia biblioteca resuelve la interacción entre el método de resolución y el problema instanciado en las clases provistas por el usuario.
La biblioteca MALLBA fue desarrollada por las universidades de Málaga, La Laguna y Barcelona y está disponible en `http://neo.lcc.uma.es/mallba/easy-mallba`. En este trabajo hemos extendido las clases provistas de MALLBA implementando los operadores de cruzamiento que se presentan en la siguiente sección.

La ejecución de las pruebas se realizó sobre un cluster de equipos Intel Pentium 4 a 2.4 GHz, cada uno con 512 Mb RAM, con el sistema operativo SuSE Linux 8.0.

4. Operadores de Cruzamiento Utilizados

Esta sección presenta la justificación del enfoque empírico del trabajo y las características de los operadores de cruzamiento considerados en el estudio.

4.1 - Motivación del estudio empírico

En el primer trabajo donde planteamos la resolución del GSP con un AG en versiones serial y paralela [2] no se realizó un análisis de operadores de cruzamiento, limitándose a trabajar con el cruzamiento de un punto. En experimentos posteriores utilizamos el cruzamiento de dos puntos y el uniforme, obteniendo mejoras en los resultados que nos condujeron a formalizar el estudio que se presenta en este artículo.

La complejidad del GSP y los aspectos intrínsecos de la codificación utilizada hacen muy complejo el análisis teórico de la influencia de los operadores de cruzamiento en la calidad de los resultados del AG. Las características de los esquemas que conviene privilegiar dependen de las particularidades del grafo que representa la red para la que se plantea el GSP. Tomando en cuenta la complejidad de diseño de redes de comunicaciones de mediano tamaño, no es sencillo explotar particularidades del problema que permitan determinar a priori un operador de cruzamiento adecuado.

Considerando la dificultad del análisis desde el punto de vista teórico, hemos utilizado un enfoque experimental para abordar el estudio de la influencia de los operadores de cruzamiento sobre la calidad de resultados de un AG para resolver el GSP.

4.2 - Operadores considerados en el estudio

No hay evidencia teórica o experimental que indique la existencia de un operador de cruzamiento universal que obtenga los mejores resultados para todo problema, siendo necesario determinar qué operador se ajusta a las características particulares de cada problema y de cada codificación para obtener buenos resultados.
Si bien existe una amplia gama de operadores de cruzamiento para AG, generalmente se han utilizado tres operadores de implementación sencilla, que trabajan sobre las codificaciones y no utilizan información del problema a resolver: el cruzamiento de un punto (1PX), el cruzamiento de dos puntos (2PX) y el cruzamiento uniforme (UPX). Este artículo presenta el análisis de estos operadores tradicionales y algunas variantes sencillas. Operadores más sofisticados, que utilizan estrategias adaptativas y adquieren cierto conocimiento del problema para mejorar su desempeño [10], no han sido considerados en este trabajo, proponiéndose su estudio para una etapa futura.

4.3 - Familia de operadores de cruzamiento de n puntos

Esta familia de operadores propone definir puntos de corte e intercambiar segmentos entre los individuos padre. Su propuesta se remonta a los trabajos iniciales sobre AG donde se probaron resultados teóricos y empíricos que popularizaron su uso con valores bajos de ***n***, generalmente 1 ó 2 [3,6]. Trabajos posteriores profundizaron el análisis teórico de esta familia de operadores, determinando cotas para la probabilidad de supervivencia para esquemas de cualquier orden. Al analizar las curvas de supervivencia en función del largo de definición para esquemas de un orden dado, Spears y De Jong [11] observaron que cuando ***n*** es par la disrupción crece al aumentar el número de puntos de corte. Cuando ***n*** es impar, para todo par de curvas de supervivencia existe un único punto de intersección, antes del cual, la disrupción crece al aumentar el número de puntos de corte. Luego de ese punto, las curvas modifican sus valores, transformándose en más disruptiva la estrategia que menos lo era y viceversa. Al considerar esquemas de mayor orden, los puntos de intersección de las curvas se desplazan hacia valores mayores, minimizando la zona en la cual se invierte el comportamiento de disrupción. Para esta familia de operadores la probabilidad de supervivencia depende fuertemente del largo de definición del esquema considerado.
En este trabajo limitaremos el estudio de esta familia a los operadores de cruzamiento de un punto (1PX), dos puntos (2PX), tres puntos (3PX) y cuatro puntos (4PX). Para esquemas de bajo largo de definición los operadores pueden ordenarse trivialmente de acuerdo a su nivel de disrupción de menor a mayor: 1PX, 2PX, 3PX y 4PX.

4.4 - Familia de operadores de cruzamiento uniforme parametrizados

Esta familia de operadores propone el intercambio aleatorio de bits entre los individuos padre dependiendo de una probabilidad fija ***p***. Las variantes paramétricas del operador se obtienen considerando ***p*** entre 0 y 0.5, de acuerdo a la simetría del problema. El operador UPX tradicional corresponde a utilizar un valor de $p = 0.5$.
El UPX fue propuesto por Syswerda [13], quien probó resultados teóricos acotando la probabilidad de supervivencia de esquemas y realizó estudios empíricos sobre la calidad de resultados al utilizar el operador para varios problemas de optimización. La probabilidad de supervivencia de esquemas bajo el operador UPX no depende del largo de definición, sino que es constante para todos los largos posibles. Posteriormente, Spears y DeJong plantearon una variante a la mecánica básica del UPX, estudiando el efecto de la parametrización de la probabilidad de intercambio entre los padres [12]. Esta variante mantiene las características del operador original, con probabilidad de supervivencia constante para los esquemas, pero al considerar parametrizaciones con valores menores de ***p***, el cruzamiento resulta menos disruptivo que el UPX tradicional.
En este trabajo limitaremos el estudio a los operadores de cruzamiento uniforme parametrizados con 0.1 (UP1X), 0.2 (UP2X), 0.3 (UP3X), 0.4 (UP4X), y 0.5 (usaremos el nombre UP5X en lugar de UPX, para homogeneizar la nomenclatura).

5. Resultados obtenidos

Esta sección presenta las instancias del GSP sobre las que se evaluaron los operadores de cruzamiento, la parametrización del AG, los resultados obtenidos y su análisis.

5.1 - Conjunto de problemas de prueba

El GSP ha sido poco abordado por la comunidad científica, y como consecuencia no existen conjuntos de problema de prueba estandarizados para comparar la calidad de resultados de los AG. En nuestro entorno de trabajo hemos obtenido resultados sobre un conjunto de tres problemas cuyas topologías, número de nodos terminales y requerimientos consideramos representativos para redes de mediano tamaño.

Los grafos de prueba se generaron seleccionando aleatoriamente puntos en el plano euclídeo para representar los vértices e incluyendo aristas con probabilidad inversamente proporcional a la distancia euclideana entre los nodos que conectan. Las características de los tres grafos utilizados se resumen en la Tabla 1, indicando número total de vértices, número de terminales y número de aristas. Los grafos son designados por un nombre que denota la cantidad total de nodos y la cantidad de nodos terminales.
El conjunto de grafos de prueba, y una descripción del formato utilizado se encuentran disponibles públicamente en `http://www.fing.edu.uy/cecal/hpc/gsp/TestSuite`.
El generador aleatorio utilizado para crear los grafos también se halla disponible públicamente en `http://www.fing.edu.uy/cecal/hpc/gsp/Generator`.

	grafo 100-10	**grafo 75-25**	**grafo 50-15**
Vértices	100	75	50
Terminales	10	25	15
Aristas	500	360	249

Tabla 1: Características del conjunto de grafos de prueba.

5.2 - Parametrización del AG

Los parámetros del AG fueron fijados para estudiar en forma aislada el efecto de los operadores de cruzamiento sobre la calidad de los resultados. Se utilizaron poblaciones de 120 individuos, que evolucionaron durante 2000 generaciones. Las probabilidades de cruzamiento y de mutación fueron fijadas en 0.95 y 0.01 respectivamente.

5.3 - Resultados

Las tablas 2, 3 y 4 presentan los resultados promedio de 30 ejecuciones realizadas para cada grafo y para cada operador. Las tablas se dividen en tres secciones:

- *Mejores resultados obtenidos* detalla los valores para los mejores individuos alcanzados en las ejecuciones.
- *Características de la población final* detalla los resultados para el individuo promedio de la última generación.
- *Esfuerzo requerido para alcanzar el mejor valor obtenido* ofrece el número de generación y el tiempo necesario para obtener el mejor resultado e incluye como referencia el tiempo total de ejecución del algoritmo.

El estudio de eficiencia no es un objetivo de este trabajo, pero hemos incluido los tiempos de ejecución para detectar si eventuales mejoras en la calidad de resultados se logran a costa de un mayor tiempo de cómputo al utilizar una familia de operadores.

	UP1X	***UP2X***	***UP3X***	***UP4X***	***UP5X***	***1PX***	***2PX***	***3PX***	***4PX***
Mejores resultados obtenidos									
Promedio	4441	4474	4485	4497.5	4500	4457	4470	4477.5	4477
Mejor	4491	4569	4540	4548	4573	4502	4532	4521	4520
Desv. Std.	33.71	34.96	32.36	28.77	27.68	40.89	32.27	40.27	30.70
Características de la población final									
Promedio	4415.80	4446.22	4457.67	4470.51	4469.28	4428.22	4441.63	4444.75	4450.15
Mejor	4465.39	4540.07	4511.64	4523.31	4532.43	4477.64	4497.97	4478.97	4491.03
Desv. Std.	34.91	34.98	33.92	29.20	26.57	41.16	30.90	39.00	31.42
Esfuerzo requerido para alcanzar el mejor valor obtenido									
Generación	1756	1484.5	1456	1456	1593.5	1861	1800	1817	1725
Tiempo (min.)	112.93	95.58	93.48	94.95	98.20	114.93	108.68	109.47	106.16
Tiempo total (min.)	123.72	120.14	121.07	117.30	117.08	122.90	121.63	120.89	120.65

Tabla 2: Resultados para el grafo de 100 nodos y 10 terminales.

Las figuras 2 y 3 presentan gráficos con los resultados comparativos para cada familia de operadores para el grafo de 100 nodos y 10 terminales.

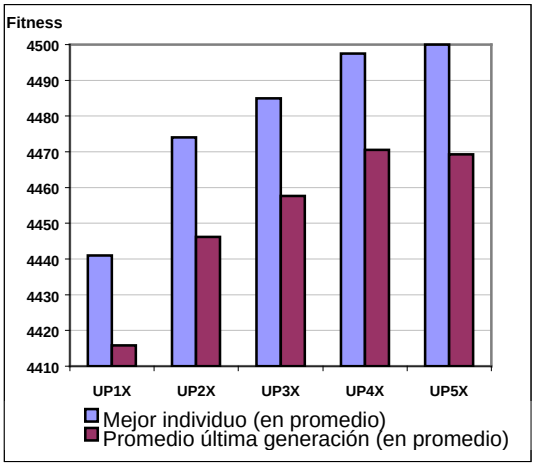


Figura 2: Resultados de operadores uniforme parametrizados, grafo 100-10.

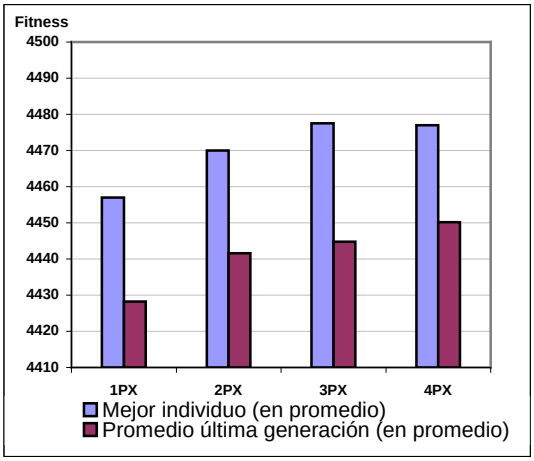


Figura 3: Resultados de operadores de n puntos, grafo 100-10.

En ambas familias de operadores las estrategias más disruptivas obtienen los mejores resultados, existiendo una concordancia entre el orden de operadores al considerar el mejor individuo y el promedio de la última generación. El desempeño de la familia de operadores uniformes parametrizados es notoriamente superior en calidad de los resultados obtenidos a los operadores de *n* puntos. Los mejores resultados para el grafo de 100 nodos y 10 terminales son alcanzados por los operadores UP5X y UP4X.

	UP1X	***UP2X***	***UP3X***	***UP4X***	***UP5X***	***1PX***	***2PX***	***3PX***	***4PX***
Mejores resultados obtenidos									
Promedio	5316.56	5341.36	5327.44	5350.67	5364.58	5290.98	5321.27	5321.5	5334.54
Mejor	5418.02	5415.41	5433.23	5465.85	5393.96	5374.11	5404.72	5388.13	5411.72
Desv. Std.	43.01	39.19	40.29	48.36	27.27	47.66	49.58	34.00	48.93
Características de la población final									
Promedio	5288.1	5304.58	5294.9	5320.78	5331.34	5254.16	5290.9	5286.75	5303.43
Mejor	5378.06	5382.83	5399.43	5441.19	5363.7	5350.71	5376.18	5368.08	5371.98
Desv. Std.	43.90	37.42	39.49	52.43	30.47	50.09	51.00	35.58	48.26
Esfuerzo requerido para alcanzar el mejor valor obtenido									
Generación	1795.5	1770.5	1404.5	1673	1543	1769.5	1792.5	1544	1736.5
Tiempo (min.)	288.07	279.17	228.93	262.96	253.56	279.57	271.27	253.30	273.10
TiempoTotal (min.)	313.12	306.12	308.42	309.42	316.43	314.93	303.00	312.80	313.30

Tabla 3: Resultados para el grafo de 75 nodos y 25 terminales.

Las figuras 4 y 5 presentan gráficos con los resultados comparativos para cada familia de operadores para el grafo de 75 nodos y 25 terminales.

Para el grafo de 75 nodos y 25 terminales, se reitera la superioridad en la calidad de los resultados de la familia de operadores uniformes parametrizados sobre la de *n* puntos. La relación directa entre la disrupción y la calidad de resultados al utilizar operadores de cruzamiento uniforme es violada por el buen desempeño del operador UP2X. En la familia de operadores de *n* puntos, el 2PX muestra buenos resultados, de calidad similar a los de 3PX. El operador 4PX obtiene los mejores resultados respecto al resto, pero las diferencias no son significativas, estando por debajo de la desviación estándar de las ejecuciones realizadas. Se mantiene la concordancia entre el orden de operadores al considerar el mejor individuo y el promedio de la última generación.

Nuevamente, los operadores UP5X y UP4X obtienen los mejores resultados.

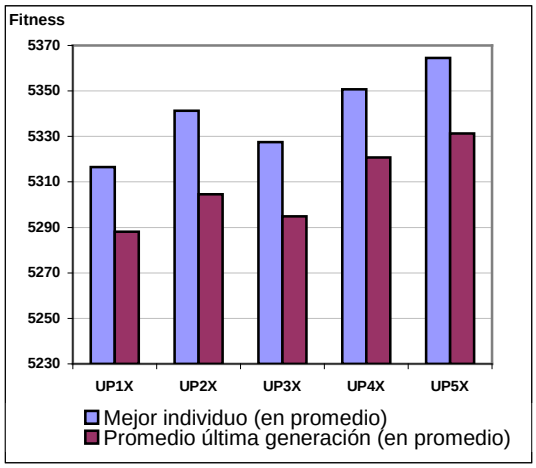


Figura 4: Resultados de operadores uniforme parametrizados, grafo 75-25.

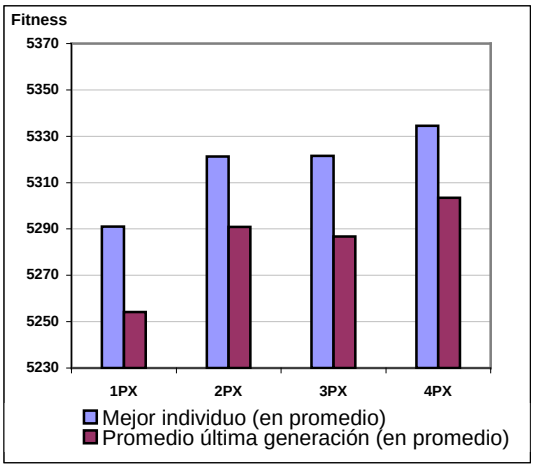


Figura 5: Resultados de operadores de n puntos, grafo 75-25.

	UP1X	***UP2X***	***UP3X***	***UP4X***	***UP5X***	***1PX***	***2PX***	***3PX***	***4PX***
Mejores resultados obtenidos									
Promedio	9301.63	9306.43	9333.71	9313.69	9347.36	9278.61	9283.62	9316.53	9344.58
Mejor	9494.02	9462.86	9454.77	9484.10	9564.75	9435.80	9464.56	9493.84	9520.01
Desv. Std.	97.95	79.66	95.22	66.13	81.96	72.22	77.46	89.49	79.19
Características de la población final									
Promedio	9235.17	9236.39	9262.97	9243.34	9267.19	9182.69	9206.88	9229.37	9249.08
Mejor	9413.57	9343.15	9400.79	9395.70	9466.82	9362.55	9404.81	9422.81	9437.19
Desv. Std.	101.40	75.76	95.02	69.78	82.14	73.62	83.08	90.47	82.06
Esfuerzo requerido para alcanzar el mejor valor obtenido									
Generación	1629.50	1673.50	1785.50	1421.00	1611.50	1570.50	1594.50	1633.00	1652.50
Tiempo (min.)	101.58	106.21	109.59	86.51	101.03	100.91	99.83	101.27	103.19
TiempoTotal (min.)	121.57	121.87	123.21	121.73	120.82	124.29	122.32	123.32	121.80

Tabla 4: Resultados para el grafo de 50 nodos y 15 terminales.

Las figuras 6 y 7 presentan gráficos con los resultados comparativos para cada familia de operadores para el grafo de 50 nodos y 15 terminales.

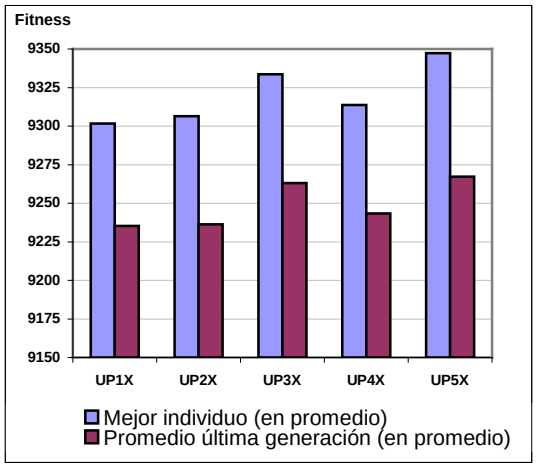


Figura 6: Resultados de operadores uniforme parametrizados, grafo 50-15.

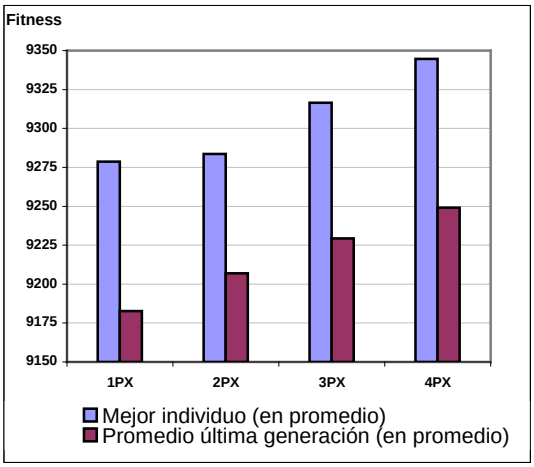


Figura 7: Resultados de operadores de n puntos, grafo 50-15.

Para el grafo de 50 nodos y 15 terminales no se percibe una relación directa entre la disrupción y la calidad de resultados al utilizar operadores de cruzamiento uniforme. Los resultados obtenidos para las diferentes parametrizaciones son muy similares, ubicándose sus diferencias por debajo de la desviación estándar de las ejecuciones realizadas, por lo que no pueden considerarse significativas.
Los operadores de *n* puntos presentan el comportamiento detectado para los restantes grafos, mejorando la calidad de resultados al considerar operadores más disruptivos.
Para este grafo, no se verifica una superioridad en la calidad de los resultados de la familia de uniformes parametrizados sobre la de *n* puntos.

En lo referente a tiempos de ejecución, no se advierten grandes diferencias entre las variantes de cada familia para los grafos estudiados. Se observa que los operadores uniformes en general detienen su evolución en un número de generaciones cercano al 3/4 del tope establecido, luego del cual no se logra mejorar los resultados obtenidos. Los operadores de *n* puntos evolucionan mejorando los resultados hasta un número mayor de generaciones, salvo para el grafo de 50 nodos y 15 terminales, donde exhiben un comportamiento similar a los uniformes.

6. Conclusiones y trabajo futuro

6.1 - Conclusiones

Los resultados obtenidos no pueden considerarse concluyentes debido a que en varios casos no presentan diferencias significativas en la calidad de resultados para los grafos considerados. De todos modos, es posible indicar que para la resolución del problema GSP con la codificación binaria utilizada, en general las estrategias más disruptivas obtienen los mejores resultados. Además, la familia de cruzamiento uniforme parametrizado obtiene resultados de mejor calidad que los obtenidos por la familia de cruzamiento de *n* puntos cuando se utiliza el criterio de parada de esfuerzo prefijado en 2000 generaciones. Dentro de los cruzamientos uniformes parametrizados, en general los mejores resultados se obtienen al trabajar con un parámetro de intercambio alto, entre 0.4 y 0.5. Para los cruzamientos de *n* puntos los mejores resultados se obtuvieron al utilizar la mayor cantidad de puntos de corte.

En lo referente a tiempos de ejecución, no se detectaron diferencias significativas que permitan concluir que el uso de una familia o de un operador en particular involucre un mayor tiempo de uso de recursos computacionales. La evolución de los operadores uniformes se detiene antes que los operadores de *n* puntos, por lo cual podrían existir diferencias en cuanto a los tiempos de ejecución, si se utilizara un criterio de parada basado en la detección de convergencia.

6.2 - Investigación actual y futura

Las líneas de investigación actual y futura se concentran en completar el estudio sobre el efecto de los operadores de cruzamiento sobre el AG aplicado al GSP, incluyendo los operadores de cruzamiento adaptativos no analizados en este trabajo. Estos operadores tienen como idea básica aprender características del problema para mejorar la calidad de los resultados y la propia técnica del cruzamiento. Este camino se presenta como prometedor para mejorar la calidad de los resultados obtenidos.

Complementariamente, debe estudiarse la escalabilidad de los resultados obtenidos para las instancias particulares analizadas en este trabajo. Al considerar grafos representativos

de redes de comunicaciones de mayor tamaño que los casos estudiados, será posible determinar la validez de las conclusiones obtenidas en este trabajo.

El estudio de los operadores de cruzamiento se realizó sobre modelos secuenciales de AG. Teniendo en cuenta el elevado costo computacional de los algoritmos, la extensión del análisis para modelos distribuidos constituye una interesante línea de trabajo. Debido a que el modelo de evolución de los AG distribuidos es diferente al de los secuenciales, verificar la extensión de los resultados obtenidos en este trabajo constituirá en sí mismo un resultado importante.

7. Agradecimientos

Este trabajo fue desarrollado en el marco del proyecto "Algoritmos Genéticos Paralelos y su Aplicación al Diseño de Redes de Comunicaciones Confiables", financiado por la Comisión Sectorial de Investigación Científica de la Universidad de la República. Agradecemos el apoyo del Dr. Héctor Cancela del Departamento de Investigación Operativa, Instituto de Computación de la Facultad de Ingeniería, Universidad de la República, Uruguay y del Dr. Enrique Alba, del Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga, España.

Referencias bibliográficas

[1] E. Alba, F. Almeida, M. Blesa, C. Cotta, M. Díaz, I. Dorta, J. Gabarró, J. González, C. León, L. Moreno, J. Petit, J. Roda, A. Rojas, F. Xhafa, *MALLBA: A Library of Skeletons for Combinatorial Optimisation,* In Proceedings of the EuroPar, pp. 927-932, 2002.

[2] M. Aroztegui, S. Árraga, S. Nesmachnow, *Resolución del Problema de Steiner Generalizado utilizando un Algoritmo Genético Paralelo*, II Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados, pp. 387-394, 2003.

[3] K. De Jong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, PhD Thesis, Dept. Computer & Communication Sciences, Univ. of Michigan, 1975.

[4] H. Esbensen, *Computing Near-Optimal Solutions to the Steiner Problem in a Graph Using a Genetic Algorithm*, Networks 26, pp. 173-185, 1995.

[5] L. Ford, D. Fulkerson, *Flows in Networks*, Princeton University Press, 1962.

[6] J. Holland, *Adaptation in Natural and Artificial Systems*, MIT Press, 1975.

[7] V. Kahn, P. Crescenzi, *A Compendium of Optimization Problems*, Disponible online: http://www.nada.kth.se/theory/problemlist.html. Consultado julio 2003.

[8] R. Karp, *Reducibility among combinatorial problems*, Complexity of Computer Communications, Plenum Press, pp. 85-103, 1972.

[9] F. Robledo, *Diseño Topológico de Redes, Casos de Estudio: The Generalized Steiner Problem y The Steiner 2-Edge-Connected Subgraph Problem*, Master Tesis, PEDECIBA Informática, Universidad de la República, Montevideo, Uruguay, 2001.

[10] W. Spears, *Recombination Parameters*, en The Handbook of Evolutionary Computation, Oxford University Press, 1997.

[11] W. Spears, K. De Jong, *An Analysis of Multi-Point Crossover*, Proceedings of Foundations of Genetic Algorithms Workshop, pp. 301-315, 1991.

[12] W. Spears, K. De Jong, *On the Virtues of Parameterized Uniform Crossover*, Proceedings of the 4th Int. Conference on Genetic Algorithms, pp. 230-236, 1991.

[13] G. Syswerda, *Uniform Crossover in Genetic Algorithms*, Proceedings of the 3rd Int. Conference on Genetic Algorithms, pp. 2-9, 1989.

ANEXO IV

AGRADECIMIENTOS

"Debemos tener bien claro que necesitamos a otras personas
para el descubrimiento y corrección de errores (y ellas a nosotros);
especialmente personas que han crecido con otras ideas en otra atmósfera.
También esto conduce a la tolerancia."

KARL POPPER

Tolerancia y responsabilidad Intelectual

Universidad de Tubinga, Alemania, 1981

AI.1. Introducción

Un importante grupo de personas ha colaborado de manera directa o indirecta en las diferentes etapas del trabajo reportado en esta Tesis. Este anexo intenta brindarles un reconocimiento formal, que complementa el obvio agradecimiento personal por parte del responsable de la Tesis.

AI.2. Agradecimientos académicos

Parte del trabajo de Maestría que se reporta en esta Tesis fue realizada en el marco de un proyecto de Iniciación a la Investigación que tuvo como nombre el título de esta Tesis. Dicho proyecto fue financiado por la Comisión Sectorial de Investigación Científica de la Universidad de la República, Uruguay.

Sin lugar a dudas el apoyo permanente y la orientación de Héctor Cancela debe figurar en el primer lugar de los agradecimientos. Durante el desarrollo del trabajo, Héctor ha cumplido cabalmente con sus funciones de Supervisor y Orientador de Maestría con la calidad académica que lo caracteriza, complementando con sus pertinentes consejos y sugerencias que en variadas ocasiones lograron aportar un punto de vista diferente y en particular la mesura necesaria para tomar decisiones cruciales a lo largo del trabajo desarrollado.

Los compañeros del Grupo de Trabajo en Procesamiento Paralelo y Distribuido y sus Aplicaciones han colaborado directamente en diferentes tareas relacionadas con el proyecto de Maestría. De este modo, Martín Pedemonte, Pablo Ezzatti, Gerardo Ares y Vincent Ho deben recibir el crédito correspondiente, no solo por sus actividades de investigación y aportes al trabajo, sino también por la ardua tarea de coexistir en un mismo entorno con el responsable de este trabajo. Este último agradecimiento se hace extensivo a Eduardo Fernández, compañero del Centro de Cálculo que, quizás sin tenerlo muy en cuenta, ha colaborado de modo importante con tareas relacionadas con este trabajo.

Enrique Alba ha colaborado directamente con una importante parte del trabajo, presentando nuevas ideas y enfoques que han sido destacadas adecuadamente en esta Tesis. Asimismo, los restantes integrantes del grupo de trabajo en Redes y Comunicación Emergente (NEO) de la Universidad de Málaga supieron recibir con hospitalidad y colaborar en variadas actividades desarrolladas en las pasantías realizadas en España. Debe figurar entonces el reconocimiento a Gabriel Luque, Francisco Chicano, Bernabé Dorronsoro y Carlos Cotta. Un especial agradecimiento a Francisco Luna por su continua colaboración prestada tanto a distancia como personalmente.

Aquellos estudiantes que han participado en proyectos de grado relacionados con los temas de investigación de la Tesis han realizado valiosos trabajos que han sido extensamente utilizados a lo largo del trabajo. Debe figurar el agradecimiento entonces para Miguel Aroztegui y Daniel Calegari, y un reconocimiento especial para Santiago Árraga, quien ha continuado trabajando en los temas relacionados con su proyecto de grado y por ende sufriendo el continuo martirio de tratar de poner en práctica aquellas ideas extravagantes que han surgido a lo largo del trabajo y quien escribe no ha tenido oportunidad de abordar.

La tarea de revisión de la Tesis por parte de Graciela Ferreira permitió identificar algunas secciones algo oscuras en el texto, que han sido oportunamente revisadas y algún gazapillo sobreviviente a las innumerables correcciones personales y por parte de Héctor. Para Graciela el agradecimiento correspondiente por la tarea realizada.

Por último, un recuerdo para Carlos López, quien cometió el error de iniciarme en esta tarea del trabajo académico y la investigación, y como si esto no hubiera sido suficiente, reincide esporádicamente.

AI.3. Agradecimiento personal

Por último, esta línea anónima representa el agradecimiento personal a quienes realizaron la tarea más difícil de todas.

REFERENCIAS BIBLIOGRÁFICAS

"En los hábitos literarios también es
todopoderosa la idea de un sujeto único.
Es raro que los libros estén firmados.
No existe el concepto del plagio: se ha
establecido que todas las obras son obra
de un solo autor, que es intemporal y es anónimo."

JORGE LUIS BORGES

Tlön, Uqbar, Orbis Tertius.
El jardín de senderos que se bifurcan, 1941.

Abramson D., Abela, J. (1992). *A Parallel Genetic Algorithm for Solving the School Timetabling Problem*. Proceedings of the 15th Australian Computer Science Conference **14**, pp. 1–11.

Abramson D., Mills, G., Perkins S. (1993). *Parallelisation of a Genetic Algorithm for the Computation of Efficient Train Schedules*. Proceedings of 1993 Parallel Computing and Transputers Conference, pp. 139–149, IOS Press.

Adamidis P. (1994). *Review of Parallel Genetic Algorithms Bibliography*. Technical Report, Automation & Robotics Laboratory, Department of Electrical & Computer Engineering, Aristotle University of Thessaloniki, Grecia.

Adamidis P. (1997). *Improving the performance of Parallel Genetic Algorithms with Co-opearting Populations with Different Evolution behavior*. Ph. D. Tesis, Department of Electrical & Computer Engineering, Aristotle University of Thessaloniki, Grecia.

Adamidis P. (1998). *Parallel Evolutionary Algorithms: A Review*. Fourth Hellenic–European Conference on Computer Mathematics and its Applications. Disponible en http://aetos.it.teithe.gr/~adamidis/Papers/HERCMA98.ps.gz. Consultado octubre 2003.

Adamidis P., Petridis, V. (1996). *Co-operating Populations with Different Evolution Behavior*. Proceedings of the International Conference on Evolutionary Computation, pp. 188–191, IEEE Press.

Adamidis P., Kazarlis S., Petridis, V. (1998). *Advanced methods for evolutionary optimisation*. Symposium on Large Scale Systems: Theory and Applications, University of Patras, Grecia.

Alba E. (2002). *Parallel Evolutionary Algorithms Can Achieve Super-Linear Performance*. Information Processing Letters **82** (1), pp. 7-13.

Alba, E., Almeida F., Blesa M., Cotta C., Diaz M., Dorta I., Gabarró J., González J., León C., Moreno L., Petit J., Roda J., Rojas A., Xhafa F. (2002). *MALLBA: A library of skeletons for combinatorial optimisation.* Proceedings of the Euro–Par, pp. 927–932.

Alba E., Cotta C. (1999). *A Survey of Distributed Genetic Algorithms*. Complexity **4** (4), pp. 31–52.

Alba E., Cotta C. (2003). *Tutorial on Evolutionary Computation*. Disponible en http://www.lcc.uma.es/~ccottap/semEC/ec.html. Consultado octubre 2003.

Alba E., Cotta C., Troya, J. M (1999a). *Numerical and Real Time Analysis of Parallel Distributed GAs with Structured and Panmictic Populations*. Proceedings of the IEEE Conference on Evolutionary Computing **2**, pp. 1019–1026, IEEE Press.

Alba E., Cotta C., Troya, J. M (1999b). *Entropic and Real-Time Analysis of the Search with Panmictic, Structured, and Parallel Distributed Genetic Algorithms*. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), pp. 773, Morgan Kaufmann.

Alba, E., Luque G. (2003). *Parallel LAN/WAN Heuristics for Optimization*. Proceedings of the International Parallel and Distributed Processing Symposium, p. 147.

Alba E., Troya, J. M. (1999). *An Analysis of Synchronous and Asynchronous Parallel Distributed Genetic Algorithms with Structured and Panmictic Islands*. Parallel and Distributed Processing, Lecture Notes in Computer Science **1586**, pp. 248-256, Springer-Verlag.

Alba E., Troya, J. M (2001). *Analyzing Synchronous and Asynchronous Parallel Distributed Genetic Algorithms*. Future Generation Computer Systems **17** (4), pp. 451–465.

Alba E., Tomassini M. (2002). *Parallelism and Genetic Algorithms*. IEEE Transactions on Evolutionary Computation **6** (5), pp. 443–462.

Amdahl, G.M. (1967). *Validity of the single-processor approach to achieving large scale computing capabilities.* American Federation of Information Processing Societies (AFIPS) Conference Proceedings **30**, pp. 483–485, AFIPS Press.

Angeline, P. (1998). *A Historical Perspective on the Evolution of Executable Structures*. Fundamenta Informaticae **35** (1-4), pp. 179–195.

Arraga, S., Aroztegui, M. (2001). *Algoritmos Genéticos Paralelos para la resolución del Problema de Steiner en Grafos*. Proyecto de grado, Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Uruguay.

Árraga S., Aroztegui M., Nesmachnow S. (2003). *Resolución del Problema de Steiner Generalizado utilizando un Algoritmo Genético Paralelo*. Actas del Segundo Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados, pp. 387–394.

Back T., Fogel D. y Michalewicz Z. (1997). *Handbook of Evolutionary Computation*. Oxford University Press, New York.

Balakrishnan, K. (1993). *Parallel Genetic Algorithms, Premature Convergence and the nCUBE*. Coms 625x Term Project, Iowa State University, USA. Disponible en http://citeseer.ist.psu.edu/91170.html. Consultado octubre 2003.

Ball, M.O. (1979). *Computing network reliability*. Operations Research **27**, pp. 823–836.

Baluja, S. (1992). *A Massively Distributed Parallel Genetic Algorithm*. Technical Report CMU–CS–92–196R, Carnegie Mellon University, USA.

Baluja, S. (1993). *Structure and Performance of Fine Grain Parallelism in Genetic Search*. Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 155–162, Morgan Kaufmann.

Baraglia, R. Perego, R. (1999). *Parallel Genetic Algorithms for Hypercube Machine*. Lecture Notes In Computer Science **1573**, pp. 691–703, Springer-Verlag.

Bastos, M., Ribeiro, C. (1999). *Reactive tabu search with path-relinking for the Steiner problem in graphs.* Proceedings of the Third Metaheuristics International Conference, pp. 31–36.

Berger, J., Barkaoui, M. (2001). *A Parallel Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows*. Lecture Notes in Computer Science **2723**, pp. 646–656, Springer-Verlag.

Bertoni A., Dorigo M. (1993). *Implicit Parallelism in Genetic Algorithms*. Artificial Intelligence **61** (2), pp. 307–314.

Bethke, A. D. (1976). *Comparison of genetic algorithms and gradient based optimizers on parallel processor: Efficiency of use of precessing capacity*. Technical Report N° 197, Ann Arbor, University of Michigan. Citado en Cantú-Paz (2001a).

Bianchini, R, Brown C., (1993). *Parallel Genetic Algorithms on Distributed Memory Architectures*. Technical Report 436, Computer Science Department, University of Rochester, USA.

Bianchini, R, Brown C., Cierniak M., Meira W. (1995). *Combining Distributed Populations and Periodic Centralized Selections in Coarse-Grain Parallel Genetic Algorithms.* Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms, pp. 483–486.

Bienstock, D., Brickell, E., Monma, C. (1990). *On the Structure of Minimum-Weight k-Connected Spanning Networks.* SIAM Journal on Discrete Mathematics **3** (3), pp. 320–329.

Bossert, W. (1967). *Mathematical optimization: are there abstract limits on natural selection?* Mathematical Challenges to the Neo-Darwinian intepretation of evolution **5**, pp. 35–46, Wistar Institute Press. Citado en Cantú-Paz (2000c).

Box G. (1957). *Evolutionary Operation: A Method for increasing industrial Productivity*. Applied Statistics **6** (2), pp. 81–101, 1957. Citado en Goldberg (1989a).

Box G. (1998), *Evolutionary Operation: A Statistical Method for Process Improvement*, Wiley.

Boyer, R.; Ballantyne, M., Hines, L. (1996). *Woody Bledsoe: His Life and Legacy*. Artificial Intelligence Magazine **17** (1), pp. 7–20.

Boyer, R., Browne, J., Misra, J. (1998). *In Memoriam Woodrow W. Bledsoe*. Disponible en http://www.utexas.edu/faculty/council/1998-1999/memorials/bledsoe/bledsoe.htm. Consultado octubre 2003.

Braun, H. (1990). *On solving travelling salesman problems by genetic algorithms*. Proceedings of Parallel Problem Solving from Nature, Lecture Notes in Computer Science **496**, pp. 129–133. Springer-Verlag.

Bravo, H., Candia, A. (1997). *A metaheuristic solution to a constrained Steiner tree problem*. 17th International Conference of the Chilean Computer Science Society, pp. 16–20.

Bremermann, H. J., Anderson R. W. (1991). *How the brain adjusts synapses-maybe*. Automated Reasoning: Essays in Honor of Woody Bledsoe, pp. 119–147, R. S. Boyer (Ed.), Kluwer. Citado en Boyer et al. (1998).

Bremermann, H.J., Rogson, M. and Sala, S. (1965). *Search by Evolution*. Biophysics and Cybernetic Systems, pp. 157–167, M Maxfield, A Callahan, L J Fogel (Eds.), Spartan Books, Washington.

Cai, G., Sun, Y. (1989). *The minimum augmentation of any graph to a k-edge-connected graph*. Networks **19**, pp. 151–172.

Calegari, D. (2002). *Algoritmos Genéticos aplicados al diseño de una red de comunicaciones confiable*, Proyecto de grado, Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Uruguay.

Cancela, H., Robledo, F., Rubino G. (2003). *Network design with node connectivity constraints*. IFIP/ACM Latin America Networking Conference, pp 13-20.

Cantu-Paz, E. (1998a). *A Summary of Research in Parallel Genetic Algorithm.* Illinois Genetic Algorithms Laboratory, Technical Report 95007, Urbana-Champaign University, USA.

Cantu-Paz, E. (1998b). *Designing efficient master-slave parallel genetic algorithms*. Proceedings of the Third Annual Conference on Genetic Programming, p. 455, Morgan Kaufmann.

Cantu-Paz, E. (2000a). *Markov chain models of parallel genetic algorithms.* IEEE Transactions on Evolutionary Computation **4** (3), pp. 216–226.

Cantu-Paz, E. (2000b). *Selection intensity in genetic algorithms with generation gaps*. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), pp. 911–918, Morgan Kaufmann.

Cantú-Paz E. (2001a). *Efficient and Accurate Parallel Genetic Algorithms*. Kluwer Academic Publisher.

Cantu-Paz, E. (2001b). *Migration policies, selection pressure, and parallel evolutionary algorithms*. Journal of Heuristics **7** (4), pp. 311–334.

Cantu-Paz, E., Goldberg, D. (1999). *On the scalability of parallel genetic algorithms*. IEEE Transactions on Evolutionary Computation **7** (4), pp. 429–449.

Cayley, A. (1889). *A Theorem on Trees*. Quarterly Journal on Pure and Applied Mathematics **23**, pp. 376–378.

Chou, H., Premkumar G., Chu, C. (2001). *Genetic Algorithm for Communication Network Design An Empirical Study of the Factors that Influence Performance*. IEEE Transactions on Evolutionary Computation **5** (3), pp. 236–249, IEEE Press.

Christofides, N., Whitlock, C. (1981). *Network synthesis with connectivity constraints – a survey*. Operational Research **81**, pp. 705–723.

Chu, C., Chou, H., Premkumar G. (2000). *Digital Data Networks Design Using Genetic Algorithms*. European Journal of Operational Research **127** (1), pp. 140–158.

Cohoon, J., Hedge S., Martin, W., Richards D. (1987). *Punctuated Equilibria: A Parallel Genetic Algorithm*. Proceedings of the Second International Conference on Genetic Algorithms and their Applications, pp. 148–154, Lawrence Erlbaum.

Corne, D., Oates, M., Smith, G. (2000). *Telecommunications Optimization*, Wiley.

Corning P. (1998). *The Synergism Hypothesis, on the Concept of Synergy and it's Role in the Evolution of Complex Systems*. Journal of Social and Evolutionary Systems **21** (2). Disponible en http://www.complexsystems.org/publications/synhypo.html.

Darwin, Ch. (1859). *On the Origin of Species by means of Natural Selection*. John Murray, Londres.

Davis L. (1991). *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York.

Davis, L., Orvosh, D., Cox, A., Qiu, Y. (1993). *A genetic algorithm for survivable network design*. Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 408–415, Morgan Kaufmann.

Davison, B., Rasheed, K. (1999). *Effect of Global Parallelism on the Behavior of a Steady State Genetic Algorithm for Design Optimization.* Proceedings of the Congress of Evolutionary Computation, pp. 534–541, IEEE Press.

De Jong, K. (1975). *An analysis of the behaviour of a class of genetic adaptive systems*. Ph. D. Tesis, University of Michigan, USA. Citado en Cantú-Paz (2000c).

Do, W. Y. (1999). *Multicast routing based on genetic algorithms*. M. Sc. Tesis, Department of Computer Science and Information Engineering, National Chung Cheng University, China.

Dowsland, K. (1991). *Hill-climbing simulated annealing and the Steiner problem in graphs*. Engineering Optimization **17**, pp. 91–107.

Dyson, G. (1998). *Darwin Among the Machines; the Evolution of Global Intelligence*. Perseus.

Esbensen H. (1994). *Computing near-optimal solutions to the Steiner problem in a graph using a genetic algorithm*. Ph. D. Tesis, Department of Computer Science, University of Aarhus, Dinamarca. Resumen publicado en Networks **26**, pp. 173–185, 1995.

Esbensen H., Mazumder P. (1994). *A genetic algorithm for the Steiner problem in a graph*. Proceedings of the European Design and Test Conference, pp. 402–406.

Eshelman L. (1991). *The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination*. Foundations of Genetic Algorithms, pp. 265–283, Morgan Kaufmann.

Eswaran, K., Tarjan, R. (1976). *Augmentation Problems*. SIAM Journal on Computing **5** (4), pp. 653–665. Citado en Robledo (2001).

Flynn, M. (1972). *Some Computer Organizations and Their Effectiveness*. IEEE Transaction on Computing **C-21**, pp. 948–960.

Fogarty, T., Huang, R. (1990). *Implementing the Genetic Algorithm on Transputer Based Parallel Processing Systems.* Proceedings of Parallel Problem Solving from Nature, pp. 145–149.

Fogel D.B. y Anderson R.W. (2000). *Revisiting Bremermann's Genetic Algorithm: I. Simultaneous Mutation of All Parameters*. Proceedings of the Congress of Evolutionary Computation, pp. 1204–1209, IEEE Press.

Fogel L. J., Owens A. J. y Walsh M. J. (1996). *Artificial Intelligence through Simulated Evolution*. John Wiley & Sons, New York.

Fogel, D.B. (1995). *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press, New Jersey.

Ford L., Fulkerson D. (1962). *Flows in Networks*. Princeton University Press, Princeton.

Foster, I. (1995). *Designing and Building Parallel Programs*. Addison-Wesley Publishing.

Frank, A (1990). *Augmenting graphs to meet edge-connectivity requirements*. Technical Report, Institute for Operations Research, University of Bonn, Alemania.

Fraser, A. S. (1957). *Simulation of genetic systems by automatic digital computers*. Australian Journal of Biologial Sciences **10**, pp. 484–491.

Friedberg R. (1958). *A Learning Machine: Part (I)*. IBM Journal of Research and Development, **2** (1), pp. 2–13. Citado en Dyson (1998).

Fujiki, C., Dickinson, J. (1987). *Using the genetic algorithm to generate LISP source code to solve the prisoner's dilemma*. Proceedings of the Second International Conference on Genetic Algorithms and their Applications, pp. 236–240, Lawrence Erlbaum. Citado en Angeline (1998).

Galiasso, P., Wainwright R. (2001). *A hybrid genetic algorithm for the point to multipoint routing problem with single split paths.* Proceedings of the ACM Symposium on Applied Computing, pp. 327–332.

Garey M., Johnson D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company.

Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek R., Sunderman, V. (1994). *PVM: Parallel Virtual Machine - A User's Guide and Tutorial for Networked Parallel Computing*. The MIT Press.

Goldberg D. (1989a). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.

Goldberg, D. (1989b). *Sizing populations for serial and parallel genetic algorithms*. Proceedings of the Third International Conference on Genetic Algorithms, pp. 70–79, Morgan Kaufmann.

Goldberg, D., Deb, K., Kargupta, H., Harik, G. (1993). *Rapid, accurate optimization of difficult problems using fast messy genetic algorithms*. Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 56–64, Morgan Kaufmann.

Goldberg, D., Korb, B., Deb, K. (1990). *Messy genetic algorithms: Motivation, analysis, and first results*. Complex Systems **3**, pp. 493–530.

Gordon, V., Whitley, D. (1993). *Serial and Parallel Genetic Algorithms as Function Optimizers*. Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 177–183, Morgan Kaufmann.

Gorges-Schleuter, M. (1989). *ASPARAGOS: An asynchronous parallel genetic optimization strategy*. Proceedings of the Third International Conference on Genetic Algorithms, pp.422–427, Morgan Kaufmann.

Gorges-Schleuter, M. (1990). *Explicit parallelism of genetic algorithms through population structures*. Proceedings of Parallel Problem Solving from Nature, pp. 150–159.

Gorges-Schleuter, M. (1992). *Comparison of Local Mating Strategies in Massively Parallel Genetic Algorithms*. Proceedings of Parallel Problem Solving from Nature, pp. 559–568.

Gorges-Schleuter, M. (1997). *Asparagos96 and the Traveling Salesman Problem*. Proceedings of the International Conference on Evolutionary Computation, pp. 171–174, IEEE Press.

Gorges-Schleuter, M. (1998). *A Comparative Study of Global and Local Selection in Evolution Strategies*. Proceedings of Parallel Problem Solving from Nature, pp. 367–377.

Grefenstette, J. (1981) *Parallel adaptive algorithms for function optimization*. Technical Report CS-81-19, Computer Science Department, Vanderbilt University, USA. Citado en Goldberg (1989a).

Grefenstette, J. J. (1991). *Conditions for implicit parallelism*. Foundations of Genetic Algorithms, pp. 252–261, Morgan Kaufmann.

Grefenstette, J., Baker, J. (1989). *How genetic algorithms work: A critical look at implicit parallelism*. Proceedings of the Third International Conference on Genetic Algorithms, pp. 20–27, Morgan Kaufmann.

Gropp, W., Lusk, E., Skjellum, A. (1999). *Using MPI*. MIT Press.

Grosso, P. (1985). *Computer simulations of genetic adaptation: Parallel Subcomponent Interaction in a Multilocus Model*. Ph. D. Tesis, University of Michigan, USA. Citado en Cantú-Paz (2000c)

Grötschel, M., Monma, C., Stoer, M. (1991). *Polyhedral Approaches to Network Survivability*. AMS Series in Discrete Mathematics and Theoretical Computer Science **5**, pp. 121–141.

Grötschel, M., Monma, C., Stoer, M. (1995a). *Polyhedral and Computational Investigations for Designing Communication Networks with High Survivability Requirements*. Journal Operations Research, **43** (6), pp. 1012–1024.

Grötschel, M., Monma, C., Stoer, M. (1995b). *Design of Survivable Networks*. Network Models, Handbooks in Operations Research and Management Science, **7**, pp. 617–672, M. Ball, T. Magnanti, C. Monma, G. Nemhauser (Eds.), North-Holland.

Gusfield, D., Martel, Ch., Naor, D. (1990). *A fast algorithm for optimally increasing the edge-connectivity*. Proceedings of Foundation of Computer Science, pp. 698–707.

Gustafson, J., Montry G., Benner, R. (1998). *Development of parallel methods for a 1024-processor hypercube*. SIAM Journal on on Scientific and Statistical Computing **9** (4), pp. 601–638.

Gustafson, L. (1988). *Reevaluating Amdahl's law*. Communications of the ACM **31** (5), pp. 532−533.

Hajba, T. (2002). *Approximation algorithms for finding a k–connected subgraph of a graph with minimum weigh.* Hungarian Electronic Journal. Disponible en http://heja.szif.hu/ANM/ANM-001130-A/anm001130a.pdf. Consultado agosto 2002.

Hart, W., Baden, S., Belew, R., Kohn S. (1996). *Analysis of the Numerical Effects of Parallelism on a Parallel Genetic Algorithm*. Proceedings of the 10th International Parallel Processing Symposium, pp. 606−612.

Hesser, J., Männer, R., Stucky, O. (1989). *Optimization of Steiner trees by genetic algorithms.* Proceedings of the Third International Conference on Genetic Algorithms, pp. 231−236, Morgan Kaufmann.

Hesser, J., Männer, R., Stucky, O. (1991). *On Steiner Trees and Genetic Algorithms*. Lecture Notes in Computer Science **565**, pp. 509−525, Springer-Verlag.

Hicklin, J. F. (1986). *Application of the genetic algorithm to automatic program generation*. Master Tesis, University of Idaho. Citado en Angeline (1998).

Hiramatsu, A., Shimamoto, N., Yamasaki, K. (1993). *A dynamic routing control based on a genetic algorithm.* IEEE International Conference on Neural Networks, pp. 1123−1128.

Hiroyasu, T., Miki, M., Watanabe, S. (1999) *Distributed Genetic Algorithms with a New Sharing Approach in Multiobjective Optimization Problems.* Proceedings of the Congress on Evolutionary Computation, pp. 69−76, IEEE Press.

Hodges, A. (2002). *Alan Turing at The Stanford Encyclopedia of Philosophy*, Edward N. Zalta (Ed.). Disponible en http://plato.stanford.edu/archives/sum2002/entries/turing. Consultado octubre 2003.

Holland, J. (1963). *Outline for a logical theory of adaptive systems*. Journal of the Association for Computer Machinery **3**, pp. 297−314. Citado en Goldberg (1989a).

Holland, J. H. (1975). *Adaption in Natural and Artificial Systems*. Ph. D. Tesis, University of Michigan, USA. Citado en Goldberg (1989a).

Huang, R., Ma, J., Hsu, F. (1997a). *A Genetic Algorithm for Optimal 3-Connected Telecommunication Network Design.* Proceedings of the 1997 International Symposium on Parallel Architectures, Algorithms and Networks, pp. 344−350.

Huang, R., Ma, J., Kunii, T., Tsuboi, E. (1997b). *Parallel Genetic Algorithms for Communication Network Design*, Proceedings of the Second AIZU International Symposium on Parallel Algorithms / Architecture Synthesis, pp. 370−377, IEEE Press.

Hwang K. (1984). *Computer Architecture and Parallel Computing.* McGraw Hill, New York.

Hwang, R., Do, W., Yang, S. (2000). *Multicast Routing Based on Genetic Algorithms*. Journal of Information Science and Engineering, **16** (6), pp. 885−901, 2000.

Jain, K. (1998). *A Factor 2 Approximation Algorithm for the Generalized Steiner Network Problem*. Annual IEEE Symposium on Foundations of Computer Science, pp. 448−457. También disponible en Combinatoria, **21** (1), pp. 39−60, 2001.

Julstrom, B. (1993). *A genetic algorithm for the rectilinear Steiner problem*. Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 474−480, Morgan Kaufmann.

Julstrom, B. (2002). *A Scalable Genetic Algorithm for the Rectilinear Steiner Problem*. Proceedings of the Congress on Evolutionary Computation, pp. 1169−1173, IEEE Press.

Julstrom, B. (2003). *A Hybrid Evolutionary Algorithm for the Rectilinear Steiner Problem*. Genetic and Evolutionary Computation Conference Workshop Program, pp. 49−55.

Kahn V., Crescenzi P. (2003). *A compendium of NP optimization problems*. Disponible en línea http://www.nada.kth.se/theory/problemlist.html. Consultada diciembre 2003.

Kajitani, Y., Ueno, S., Wada, H. (1988). *Minimum augmentation of a tree to a k-edge-connected graph*. Networks **18**, pp. 19–25.

Kapsalis, A., Rayward-Smith, V., Smith, G. (1993). *Solving the graphical Steiner tree problem using genetic algorithms.* Journal of the Operational Research Society **44**, pp. 397–406.

Karp R. (1972). *Reducibility among combinatorial problems*. Complexity of Computer Communications, pp. 85–103, Plenum Press.

Kirkpatrik, S., Gelatt, C., Vecchi, M. (1983). *Optimization by simulated annealing*. Science **220**(4598), pp. 671–680.

Kliewer, G. (2000). *A General Software Library for Parallel Simulated Annealing*. Proceedings of EURO Winter Institute on Metaheuristics in Combinatorial Optimisation. Disponible en http://citeseer.nj.nec.com/article/kliewer00general.html. Consultado noviembre 2003.

Kou, L. Markowsky, G., Berman, L. (1981). *A Fast Algorithm for Steiner Trees.* Acta Informatica **15**, pp.141–145, Springer-Verlag.

Koza J. (1996). *Origins of Genetic Programming*. Disponible en Genetic Algorithms Digest, http://www.aic.nrl.navy.mil/galist/. Consultado octubre 2003.

Lamont, G., Gates, G, Merkle, L. (1997). *A MPI Implementation of the Fast Messy Genetic Algorithm.* Proceedings of the Intel Supercomputer Users Group Conference (ISUG'97).

Lamont, G., Gates, G, Merkle, L. (1998). *Scalability of a MPI-Based Fast Messy Genetic Algorithm.* Proceedings of ACM Symposium on Applied Computing, pp. 386–393.

Levine, D. (1993). *A Genetic Algorithm for the Set Partitioning Problem*. Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 81–487, Morgan Kaufmann.

Levine, D. (1994). *A Parallel Genetic Algorithm for the Set Partitioning Problem*. Ph. D. Tesis, Illinois Institute of Technology, USA.

Lin, S.C., Punch, W.F., Goodman, E.D. (1994). *Coarse-Grain Parallel Genetic Algorithms: Categorization and New Approach.* Proceedings of the Sixth IEEE Parallel & Distributed Processing, pp. 28–37, IEEE Press.

Ljubic, I., Kratica, J. (2000). *A genetic algorithm for biconnectivity augmentation problem*. Proceedings of the IEEE Congress on Evolutionary Computation, pp. 89–96, IEEE Press.

Ljubic, I., Raidl, D., Kratica, J. (2000). *A hybrid GA for the edge-biconnectivity augmentation problem*. Proceedings of Parallel Problem Solving from Nature, pp. 641–650, Springer-Verlag.

Luebke, E., Provan J. (2000). *On the structure and complexity of the 2-connected Steiner network problem in the plane*, Operations Research Letters **26**, pp. 111–116.

Manderick, B., Spiessens, P (1989). *Fine-Grained Parallel Genetic Algorithms*. Proceedings of the Third International Conference on Genetic Algorithms, pp. 428–433, Morgan Kaufmann.

Martins, S., Resende, M., Ribeiro, C., Pardalos, P. (2000). *A Parallel Grasp For The Steiner Tree Problem In Graphs Using Hybrid Local Search Strategy*. Journal of Global Optimization **17**, pp. 267–283.

Maruyama, T., Hirose, T., Konagaya, A. (1993). *A Fine-Grained Parallel Genetic Algorithm for Distributed Parallel Systems*. Proceedings of the Third International Conference on Genetic Algorithms, pp. 184–190, Morgan Kaufmann.

Mc Carthy, J. (1990*). Formalization of common sense, papers by John Mc Carthy*. Ablex.

Mc Carthy, J.(2003). Página personal de John Mc Carthy en Stanford University, http://www-formal.stanford.edu/jmc/. Consultado octubre 2003

McCarthy, J. (2003). *Generality in artificial intelligence*. Communications of the ACM **30** (12), pp. 1030-1035.

Merkle, L., Lamont, G. (1993). *Comparison of Parallel Messy Genetic Algorithms Data Distribution Strategies*. Proceedings of Fifth International Conference on Genetic Algorithms, pp. 191−198, Morgan Kaufmann.

Metropolis, N., Rosenbluth A., Rosenbluth M., Teller, A., Teller, E. (1953). *Equation of state calculations by fast computing machines*. Journal of Chemical Physics **21**, pp. 1087−1092.

Michalewicz Z., (1992). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag.

Minoux, M. (1990). *Efficient greedy heuristics for Steiner tree problems using reoptimization and supermodularity*, INFOR, **28**, pp. 221–233.

Minsky, M. (1971). *Conjetura de Minsky*. Comentario no publicado. Citado en *Parallel Programming in C for the Transputer*. Thiebaut, D. Disponible en http://cs.smith.edu/~thiebaut/transputer. Consultado noviembre 2003.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms.* MIT-Press, Cambridge, Serie Complex Adaptive Systems.

Mitchell, M. (1998). *Computation in Cellular Automata*. Non-standard computation, pp. 95−140 T. Gramss, S. Bornholdt, M. Gross, M. Mitchell, T. Pellizzari (Eds.), Wiley-VCH.

Monma, C., Munson, B., Pulleyblank W. (1990). *Minimum-weight two connected spanning networks*. Mathematical Programming **46**, pp. 153−171.

MPI Forum (2003). *MPI* (*Message Passing Interface*) *Forum Home Page.* Disponible en línea http://www.mpi-forum.org/. Consultada diciembre 2003.

Mühlenbein, H. (1989). *Parallel genetic algorithms, population genetic and combinatorial optimization*. Proceedings of the Third International Conference on Genetic Algorithms, pp. 416−421, Morgan Kaufmann.

Mühlenbein, H. (1991). *Evolution in Time and Space - The Parallel Genetic Algorithm*. Foundations of Genetic Algorithms, pp. 316−337, Morgan Kaufmann.

Mühlenbein, H., Gorges-Schleuter, M., Kramer, O. (1988). *Evolution algorithms in combinatorial optimization*. Parallel Computations **7**, pp. 65−85.

Munetomo, M., Takahashi, M., Takai, Y., Sato, Y. (1993). *A Cooperative Search Strategy Using Hierarchical Genetic Algorithms and Its Implementation on the UNIX-Network.* Information Processing Society of Japan Artifical Intelligence 91. Disponible en http://www.ipsj.or.jp/members/SIGNotes/Eng/02/1993/091/article002.html. Consultado octubre 2003.

Munetomo, M., Takai, Y., Sato, Y. (1993). *An Efficient Migration Scheme for Subpopulation-Based Asynchronously Parallel Genetic Algorithms*. Proceedings of the Fifth International Conference on Genetic Algorithms, p. 649, Morgan Kaufmann.

Nesmachnow S., Cancela H., Alba E. (2003). *Técnicas Evolutivas Aplicadas al Diseño de Redes de Comunicaciones Confiables*. Actas del Tercer Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB '04), pp. 388−395.

Nesmachnow, S. (2002a). Aplicación de las Técnicas de Procesamiento Paralelo a la Implementación de Algoritmos Genéticos. Reporte Técnico INCO 06−02, Facultad de Ingeniería, Universidad de la República, Uruguay.

Nesmachnow, S. (2002b). Evolución en el Diseño e Implementación de Algoritmos Genéticos Paralelos. Actas de la XXVIII Conferencia Latinoamericana de Informática, Montevideo, Uruguay.

Newell, A., Simon H. (1963). *GSP - A Program that Simulates Human Thought*. Computers and Thought, pp. 279–96, McGraw–Hill. Citado en Mc Carthy (1990).

Nowostawski, M. (1998). *Parallel Genetic Algorithms in Geometry Atomic Cluster Optimization and Others Applications.* M.S. Tesis, School of Computer Science, The University of Birmingham, UK.

Nowostawski, M. (2002). *Parallel Genetic Algorithms and Sequencing Optimisation.* University of Birmingham Project Report. Disponible en http://marni.otago.ac.nz/~mariusz/gzipped/rep2r.ps.gz. Consultado junio 2002.

Nowostawski, M., Kwasnicka, H. (1997). *The Search Engine for Information Systems Based on Parallel Genetic Algorithm.* Proceedings of Second International Conference on Parallel Processing & Applied Mathematics, pp. 442–451.

Nowostawski, M., Poli, R. (1999a). *Parallel Genetic Algorithms Taxonomy*. Proceedings of Third International Conference on Knowledge-based Intelligent Information Engineering Systems, pp. 88–92, IEEE Press.

Nowostawski, M., Poli, R. (1999b). *Dynamic Demes Parallel Genetic Algorithm.* Proceedings of Third International Conference on Knowledge-based Intelligent Information Engineering Systems, pp. 93–98, IEEE Press.

O'Reilly, U. M. (1995). *An Analysis of Genetic Programming*. Ph. D. Tesis, Carleton University, Ottawa-Carleton Institute for Computer Science, Canada.

Osborne L.J., Gillet, B.E. (1991). *A comparison of two simulated annealing algorithms applied to the directed Steiner problem on networks*. ORSA Journal on computing **3** (3), pp. 213–225.

Pedemonte, M., Nesmachnow, S. (2003). *Estudio Empírico de Operadores de Cruzamiento en un Algoritmo Genético Aplicado al Problema de Steiner Generalizado*, Actas del IX Congreso Argentino de Ciencias de Computación.

Pedrycz, W., Vasilakos, A. (2001). *Computational intelligence in telecommunications networks.* CRC Press.

Pettey, C., Leuze M. (1989). *A Theoretical Investigation of a Parallel Genetic Algo*rithm. Proceedings of the Third International Conference on Genetic Algorithms, pp. 398–405, Morgan Kaufmann.

Pettey, C., Leuze M., Grefenstette, J. (1987). *Genetic Algorithms and their implementation.* Proceedings of the Second International Conference on Genetic Algorithms and their Applications, pp. 155–161, Lawrence Erlbaum.

Plesnik, J. (1992). *Heuristics for the Steiner Problem in Graphs*. Discrete Applied Mathematics **37**/**38**, pp. 451–463.

Poggi de Aragão, M., Ribeiro, C., Uchoa, E., Werneck, R. (2001). *Hybrid Local Search for the Steiner problem in graphs*. Extended Abstracts of the Fourth Metaheuristics International Conference, pp. 429–433.

Poli R. (1999). *Schema theorems without expectations*. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) **1**, p. 806, Morgan Kaufmann.

Poli R. (2000). *Why the Schema Theorem is Correct also in the Presence of Stochastic Effects*. Proceedings of the Congress on Evolutionary Computation, pp. 487–492, IEEE Press.

Premkumar G., Chu, C., Chou, H. (2000). *Telecommunications Network Design Decision: A Genetic Algorithm Approach*. Decision Sciences **31** (2), pp. 483–506.

Prim, RC, (1957). *Shortest Connection Networks and Some Generalizations*. Bell System Technical Journal, pp. 1389−1401.

Provan, J.S., Ball, M. O. (1983). *The complexity of counting cuts and of computing the probability that a graph is connected*. SIAM Journal on Computing **12**, pp. 777−788.

Prufer. H. (1918). *Neuer beweis eines Satzes uber Permutationen*. Archiv fur Mathematik und Physik **27**, pp.742−744.

Rasheed, K. (1998). *GADO: A genetic algorithm for continuous design optimization.* Technical Report DCS-TR-352, Department of Computer Science, Rutgers University, USA.

Ravi, R. (1994). *Steiner Trees and Beyond: Approximation Algorithms for Network Design*. Ph. D. Tesis, Department of Computer Science, Brown University, USA. Disponible como Technical Report CS-93-41, Brown University Computer Science.

Rechenberg, I. (1965). *Cybernetic Solution Path of an Experimental Problem*. Royal Aircraft Establishment, Library Translation No. 1122, Farnborough, Inglaterra. Citado en Mitchel (1996).

Ribeiro, C., De Souza, M. (2000). *Improved Tabu Search For The Steiner Problem In Graphs*. Journal of Global Optimization, **17** (1-4), pp. 267−283.

Ribeiro, C.C., Uchoa, E., Wernek R. F. (2002). *A hybrid GRASP with perturbations for the Steiner problem in graphs*. INFORMS Journal on Computing, **14** (3), pp. 228−246.

Robertson, G. (1987). *Parallel Implementation of Genetic Algorithms in a Classifier System*. Proceedings of the Second International Conference on Genetic Algorithms and their Applications, pp. 140−147, Lawrence Erlbaum.

Robledo, F. (2001). *Diseño Topológico de Redes, Casos de Estudio: The Generalized Steiner Problem, y The Steiner 2-Edge-Connected Subgraph Problem*. Tesis de Maestría en Informática, PEDECIBA Informática, Uruguay.

Sampat, P., Thuente, D. (1995). *Mathematical programming in a hybrid genetic algorithm for Steiner point problems*. Proceedings of the 1995 ACM Symposium on Applied Computing, pp. 357−363, ACM Press.

Schwefel, H. P. (1975). *Evolutionsstrategie und Numerische Optimierung*. Ph. D. Tesis, Technical University, Berlin, Alemania. Citado en Mitchell (1996).

Schwehm, M. (1996). *Parallel Population Models for Genetic Algorithms*. Universitat Erlangen-Nürnberg. Disponible en http://citeseer.ist.psu.edu/schwehm96parallel.html. Consultado octubre 2003.

Shonkwiler, R. (1993). *Parallel Genetic Algorithms*. Proceedings of the Fifth International Conference on Genetic Algorithms, p. 658, Morgan Kaufmann.

Smith, S. F. (1983). *Flexible learning of problem solving heuristics through adaptive search*. Proceedings of the Eighth International Joint Conference on Artificial Intelligence, pp. 422−425. Citado en Angeline (1998).

Sprave, J. (1999). *A Unified Model of Non-Panmictic Population Structures in Evolutionary Algorithms*. Technical Report ISSN 1433−3325, Universität Dortmund, Alemania.

Starkweather, T., Whitley, D., Mathias, K. (1991). *Optimization Using Distributed Genetic Algorithms*. Proceedings of Parallel Problem Solving from Nature, pp. 176−185, Springer-Verlag.

Stoer, M. (1996). *Design of Survivable Networks*. Lecture Notes in Mathematics **1531**, Springer-Verlag.

Sunderam, V., Dongarra, J., Geist, A., Manchek, R. (1994). *The PVM Concurrent Computing System: Evolution, Experiences and Trends*. Parallel Computing, **20** (4), pp. 531–547. Disponible en http://www.netlib.org/ncwn/pvm-pc-evol.ps. Consultado octubre 2003.

Tanenbaum A. (1998). *Structured Computer Organization*. Prentice Hall.

Tanenbaum, A. (1997). *Redes de Computadoras*. Prentice–Hall.

Tanese, R. (1989a). *Distributed Genetic Algorithms for Function Optimization*. Ph. D. Tesis, Department of Electrical Enineering and Computer Science, University of Michigan, USA.

Tanese, R. (1989b). *Distributed Genetic Algorithms*. Proceedings of the Third International Conference on Genetic Algorithms, pp. 434–439, Morgan Kaufmann.

Thornton C. (1998). *The Building Block Fallacy*. Complexity International, **4**. Disponible en http://journal-ci.csse.monash.edu.au/ci/vol04/thornton/building.htm. Consultado marzo 2003.

Tomassini, M. (1993a). *Massively parallel evolutionary algorithms*. Proceedings of the Second Connection Machine User Meeting, París, Francia.

Tomassini, M. (1993b). *The Parallel Genetic Cellular Automata: Application to Global Function Optimization*. Artificial Neural Nets and Genetic Algorithms, pp. 385–391.

Tomassini, M. (1995). *A Survey of Genetic Algorithms*. Annual Reviews of Computational Physics, pp. 87–118.

Tomassini, M. (1999). *Parallel and Distributed Evolutionary Algorithms: A Review*. Evolutionary Algorithms in Engineering and Computer Science, pp. 113–133.

Top500 (2003). *TOP500 Supercomputer sites*. Disponible en http://www.top500.org. Consultada noviembre 2003.

Tsutsui, S., Fujimoto, Y. (1993). *Forking Genetic Algorithm with Blocking and Shrinking Modes* (*fGA*). Proceedings of the Fifth International Conference on Genetic Algorithm, pp. 206–213, Morgan Kaufmann.

Tsutsui, S., Fujimoto, Y. (1994). *Extended Forking Genetic Algorithm for Order Representation* (*o-fGA*). Proceedings of the International Conference on Evolutionary Computation **1**, pp. 170–175.

Tsutsui, S., Fujimoto, Y., Ghosh, A. (1997). *Forking GAs: GAs with Search Space Division Schemes*. IEEE Transactions on Evolutionary Computation **5** (1), pp. 61–80, IEEE Press.

Vazirani, V. (1998). *The Steiner Tree Problem and Its Generalizations*. International Workshop on Approximation Algorithms for Combinatorial Optimization, pp. 33–38.

Vazirani, V. (2000). *Recent results on approximating the Steiner tree problem and its generalizations*. Theoretical Computer Science **235**, pp. 205–216.

Venkateswaran, R., Obradovic, Z., Raghavendra, C. (1996). *Cooperative Genetic Algorithm for Optimization Problems*. Proceedings of the Second Online Workshop on Evolutionary Computation, pp. 49–52.

Von Neumann, J. (1966). *Theory of Self-Reproducing Automata*. University of Illinois Press, Champaign, Illinois, USA. Citado en Mitchell (1998).

Voss, S. (1990). *Steiner-Probleme in Graphen*. Anton Hain, 1990.

Voss, S. (1992). *Steiner's problem in graphs: Heuristic methods*. Discrete Applied Mathematics **40**, pp. 45–72.

Voss, S., Gutenschwager, K. (1992). *A chunking based genetic algorithm for the Steiner tree problem in graphs*. Network Design: Connectivity and Facilities Location, DIMACS Series in Discrete Mathematics and Theoretical Computer Science **40**, pp. 335–355.

Wakabayashi, S. (2002). *A Genetic Algorithm for the Rectilinear Steiner Tree Problem in VLSI Interconnect Layout*. Journal of the Information Processing Society of Japan, **43** (5), pp. 1315–1322.

Watanabe, T., Nakamura A. (1987). *Edge-connectivity augmentation problems*. Computer and System Sciences, **35**, pp. 96–144.

Watson, R. y Pollack J., (1999). *How symbiosis can guide evolution*. Proceedings of the Fifth European Conference on Artificial Life, pp. 29–38, Springer-Verlag.

Watson, R. y Pollack J., (2000). *Symbiotic Combination as an Alternative to Sexual Recombination in Genetic Algorithms*. Proceedings of Parallel Problem Solving from Nature, pp. 425–434, Springer-Verlag.

Werneck, R. (2001). *Problema de Steiner em Grafos: Algoritmos Primais, Duais e Exatos*; Master Tesis, Departmento de Informatica, Pontifícia Universidade Católica do Rio de Janeiro.

Whitley, D. (1989). *The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best*. Proceedings of Third International Conference on Genetic Algorithms, pp. 116–121, Morgan Kaufmann.

Whitley, D. (1993). *Cellular Genetic Algorithms*. Proceedings of the Fifth International Conference on Genetic Algorithms, p. 658, Morgan Kaufmann.

Whitley, D., Beveridge, R., Guerra C., Graves. C. (1997b). *Messy Genetic Algorithms for Subset Feature Selection*. Proceedings of the Seventh International Conference on Genetic Algorithms, Morgan Kaufmann. Disponible en http://citeseer.nj.nec.com/whitley97messy.html. Consultado setiembre 2002.

Whitley, D., Kauth, J. (1998). *GENITOR: A Different Genetic Algorithm.* Technical Report CS-88-101, Colorado State University. Proceedings of the Rocky Mountain Conference on Artificial Intelligence, pp. 118–130.

Whitley, D., Rana, S., Heckendorn, R. (1997a). *Island Model genetic Algorithms and Linearly Separable Problems*. Evolutionary Computing, AISB Workshop, pp. 109–125.

Whitley, D., Rana, S., Heckendorn, R. (1998). *Representation Issues in Neighborhood Search and Evolutionary Algorithms*. Genetic Algorithms and Evolution Strategies in Engineering and Computer Science, pp. 39–57.

Whitley, D., Rana, S., Heckendorn, R. (1999). *The Island Model Genetic Algorithm: On Separability, Population Size and Convergence*. Journal of Computing and Information Technology, Special Issue on Evolutionary Computing **7** (1), pp. 33–47.

Whitley, D., Starkweather, T. (1990). *GENITOR II: a Distributed Genetic Algorithm*. Journal of Experimental and Theoretical Artificial Intelligence **2**, pp. 189–214.

Williamson, D., Goemans, M., Mihail, M., Vazirani, V. (1995). *A Primal-Dual Approximation Algorithm for the Generalized Steiner Network Problem*. Combinatorica **15**, pp. 435–454.

Wright, A. (2000). *The Exact Schema Theorem*. Disponible en http://www.cs.umt.edu/u/wright/papers/schema.pdf. Consultado octubre 2003.

Xianwei, Z., Changjia, C., Gang, Z. (2000). *A genetic algorithm for multicasting routing problem*, Proceedings of International Conference on Communication Technology. Disponible en http://www.ifip.or.at/con2000/icct2000/icct069.pdf. Consultado noviembre 2003.

Xu, J., Chiu, S., Glover F. (1995). *Tabu Search Heuristics For Designing A Steiner Tree Based Digital Line Network.* Proceedings of Third International Conference on Telecommunication Systems Modeling and Analysis, Telecommunication Systems **8** (1), pp. 55-77.

Ycart, B. (2002). *Modèles et algorithmes markoviens*. Mathematiques et applications **39**, Springer-Verlag, New York.

Zhou, X., Chen, C., Zhu, G. (2000). *A Genetic Algorithm for Multicasting Routing Problem.* Proceedings of the International Conference on Communication Technology, pp. 1248−1253, IEEE Press.

Zhu, L., Schoenefeld, D., Wainwright, R. (1998). *A Genetic Algorithm for the Point to Multipoint Routing Problem with Varying Number of Requests.* Proceedings of the International Conference on Evolutionary Computation, pp. 171−176, IEEE Press.