UNIVERSIDAD DE GRANADA

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Departamento de Ciencias de la Computación e Inteligencia Artificial

Algoritmos Genéticos Locales

Memoria de Tesis presentada por

Carlos García Martínez

como requisito para
optar al grado de Doctor
en Informática

Granada Septiembre de 2008

Editor: Editorial de la Universidad de Granada
Autor: Carlos García Martínez
D.L.: GR. 2581-2008
ISBN: 978-84-691-7826-3

UNIVERSIDAD DE GRANADA

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Departamento de Ciencias de la Computación e Inteligencia Artificial

Algoritmos Genéticos Locales

Memoria de Tesis presentada por

Carlos García Martínez

para optar al grado de Doctor en Informática

Director

Dr. Manuel Lozano Márquez

Granada Septiembre de 2008

La memoria titulada '*Algoritmos Genéticos Locales*', que presenta D. Carlos García Martínez para optar al grado de doctor, ha sido realizada dentro del programa de doctorado '*Diseño, Análisis y Aplicaciones de Sistemas Inteligentes*'del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada, bajo la dirección del doctor D. Manuel Lozano Márquez.

Granada, Septiembre de 2008

El Doctorando

El Director

Fdo. Carlos García Martínez

Fdo. Manuel Lozano Márquez

Tesis Doctoral parcialmente subvencionada por la Comisión Interministerial de Ciencia y Tecnología con los proyectos TIC2005-08386-C05-01 y TIN2008-05854 y una beca de la Secretaría de Educación y Universidades del Gobierno Español.

Agradecimientos

La presente memoria es un importante resultado de un largo camino de trabajo y sobretodo, aprendizaje. Muchas personas han estado a mi lado en este recorrido, aportando lo que cada una de ellas son.

Agradezco a mi director, Manuel Lozano, su esfuerzo y dedicación, su capacidad de trabajo, sus puntos de vista, su pesimismo, sus comentarios sobre los revisores, su casi inagotable búsqueda de la perfección, su hombro en la depresión... Gracias, Manolo.

Agradezco al grupo de investigación *Soft Computing and Intelligent Information Systems* y a sus miembros su ayuda prestada y su sentimiento de grupo. En particular, quiero destacar a Paco Herrera, excelentísimo director del grupo, y a Daniel Molina y Ana Sánchez, compañeros de fatigas.

Agradezco al Departamento de *Informática y Análisis Numérico* de la Universidad de Córdoba, su acogida. Quiero destacar el trato y cercanía que Sebastián Ventura y Antonio Calvo me han ofrecido.

Agradezco a mis *ex*-compañeros de despacho, José Raúl Romero (ahora, `1`), Enrique Yeguas (ahora, `2`) y Rafael Muñoz (ahora `3`) (yo seré el `4`, cuando pongan las puertas), su amistad, disposición y apoyo, sus risas (y formas de reír), sus tutorías (y formas de realizar tutorías) y sus pensamientos (y formas de pensar). No olvido al que todavía echamos de menos, Carlos Porcel. Agradezco también a mis otros compañeros de despacho, Eva y Amelia (¿se puede ser más feliz?), Antonio y María (son papis), Pedro (es único), Luisma (¿una cerveza?) y Sonia (Doctora^2).

Agradezco a Manuel Jesús Marín, Francisco Adarve y Esther Pina, su íntima amistad. En particular, agradezco con el corazón el tiempo que Manolo Marín dedica a enviarnos un email de vez en cuando.

Agradezco a mis padres y a Sonia Romero, todo. Mis padres han hecho lo que realmente soy y Sonia ha tenido que aguantarlo. Ya en serio, agradezco a mi madre el infinito amor por sus hijos e infinita dedicación a nuestro aprendizaje. Gracias, Mamá. A mi padre, los valores transmitidos (que casi he grabado en mi carácter). En particular, exigiendo responsabilidad, siempre se ha preocupado por que seamos libres. Gracias, Papá. A Sonia, su amor (que como matemático, no entiendo), presencia, alegría, desorden, comida, aguante... Gracias, Sonia.

Agradezco también a Dios su atención, en particular, escuchar las peticiones que mi madre realiza por sus hijos.

Y tras todos estos párrafos, agradezco a los no mencionados: hermanos, familiares, conocidos... que ofrecen lo que son.

II

Índice general

Introducción

A Planteamiento

En la industria y la ciencia existen una serie de problemas reales de optimización de difícil solución que se caracterizan por su complejidad computacional (son NP-duros) y porque los algoritmos exactos disponibles para abordarlos son ineficientes o simplemente imposibles de aplicar. Las metaheurísticas (MHs) (Blum y Roli, 2003; Glover y Kochenberger, 2003; Siarry y Michalewicz, 2008) constituyen una familia de métodos aproximados de propósito general consistentes en procedimientos iterativos que guían una heurística subordinada combinando de forma inteligente distintos conceptos para explorar y explotar adecuadamente el espacio de búsqueda asociado a este tipo de problemas. Las MHs se han aplicado sobre problemas de campos muy diversos de la actividad humana, mostrando su capacidad para proporcionar soluciones aceptablemente buenas (no necesariamente óptimas) en un tiempo razonable. Existe un grupo de MHs que siguen paradigmas bien diferenciados y que habitualmente se citan como referentes clásicos, ya que disponen de unos antecedentes históricos muy consolidados. Este grupo incluye Enfriamiento Simulado, Búsqueda Tabú, Búsqueda Local con Multiarranque Aleatorio, Búsqueda Local Iterativa, Búsqueda de Vecindario Variable, Procedimientos de Búsqueda Ávidos Aleatorios y Adaptativos, Algoritmos Evolutivos y Búsqueda Dispersa.

Hay dos factores importantes a la hora de diseñar MHs: *intensificación y diversificación* (Blum y Roli, 2003). La diversificación generalmente se refiere a la habilidad de visitar muchas regiones diferentes del espacio de búsqueda, mientras que la intensificación se refiere a la habilidad de obtener soluciones de alta calidad en esas regiones. Un algoritmo de búsqueda debe alcanzar un equilibrio táctico entre estos dos factores, de alguna forma conflictivos, para resolver exitosamente el problema tratado.

Los *métodos de búsqueda local* (BL) son algoritmos especialmente diseñados para ofrecer una alta intensificación. Dada una solución inicial, son capaces de explorar eficaz y eficientemente la región del espacio asociada, alcanzar la *base de atracción* de un óptimo local y aproximarse a él con un alto grado de precisión, consumiendo pocos recursos computacionales. Para ello, el método de BL empieza desde la solución dada, e iterativamente intenta reemplazar la solución actual por otra cercana, que sea mejor.

En la actualidad, los métodos de BL, reciben una atención especial. De hecho, existe una categoría de MHs, estado del arte de muchos problemas, para las cuales las técnicas de BL son un componente fundamental (Blum y Roli, 2003; Glover y Kochenberger, 2003; Ribeiro y Hansen, 2002; Siarry y Michalewicz, 2008; Voß y otros, 1999). Su modo de operación se resume básicamente en generar, de alguna forma, soluciones que la técnica de BL optimizará. A este tipo de MHs las llamaremos *MHs basadas en BL*.

Los Algoritmos Genéticos (AGs) (Goldberg, 1989b; Holland, 1975) son MHs inspiradas en los procesos genéticos de los organismos naturales y en los principios de la evolución natural de poblaciones. Su idea básica es mantener una *población de cromosomas*, los cuales representan soluciones candidatas a un problema concreto, que evolucionan con el tiempo a través de un proceso de competición y variación controlada. En su formulación inicial, los AGs utilizaban el alfabeto binario para codificar las soluciones, sin embargo, también se han tenido en cuenta otro tipo de codificaciones como la real (Davis, 1991b; Deb y Beyer, 2001; Deb, 2005; Herrera y otros, 1998; Michalewicz, 1992), dando lugar a los *AGs con codificación real* (AGCRs). Uno de los componentes más importantes de los AGs es el operador de cruce, método capaz de combinar la información de los individuos de la población. De hecho, se puede considerar como una de las características que definen y diferencian a los AGs de otros algoritmos basados en la evolución natural. El operador de cruce es un elemento determinante del equilibrio entre intensificación y diversificación mantenido por el AG. Por ello, se ha considerado frecuentemente como pieza clave para hacer AGs más efectivos.

En los últimos años, el atractivo de los AGs como procedimientos de búsqueda ha motivado el diseño de AGs específicos que actúan como métodos de BL (es decir, pretenden proveer una alta intensificación). De hecho, se han presentado varias propuestas de AGs con este propósito (Herrera y Lozano, 2000a; Kazarlis y otros, 2001; Lozano y otros, 2004; Mutoh y otros, 2006; Noman y Iba, 2008; Potts y otros, 1994; Tsutsui y otros, 1997). A este tipo de MHs las denominaremos *Algoritmos Genéticos Locales* (AGLs). Los AGLs son una novedosa categoría de MHs basadas en poblaciones para proveer intensificación. Éstos presentan dos características muy interesantes:

- *Los AGLs pueden producir un rendimiento superior al de los métodos clásicos de BL.* Kazarlis y otros (2001) argumenta que los AGLs pueden *recorrer* caminos complejos que llevan a soluciones de gran calidad, que las técnicas clásicas de BL no pueden seguir.

- *Los AGLs pueden mejorar los resultados de las MHs basadas en BL.* Los AGLs pueden asumir fácilmente el papel de los métodos de BL en las MHs que los usan, obteniendo de esta forma, una nueva clase de MHs que podemos denominar como *MHs basadas en AGLs*. De hecho, en la literatura han aparecido varios ejemplos de Algoritmos Meméticos que usan un AGL en el lugar del método clásico de BL (Kazarlis y otros, 2001; O'Reilly y Oppacher, 1995; Lozano y otros, 2004; Noman y Iba, 2008; Mutoh y otros, 2006; Gang y otros, 2007; Lima y otros, 2006; Sastry y Goldberg, 2004). La importancia actual de las MHs basadas en BL en la resolución de problemas de optimización complejos justifica la investigación de nuevas

propuestas de MHs basadas en AGLs capaces de mejorar su rendimiento. En la actualidad, ésta es una línea de investigación prometedora y atractiva.

En esta memoria, profundizaremos en el estudio de los AGLs y su uso dentro de MHs. En particular, diseñaremos nuevos modelos de AGLs y MHs basadas en éstos, que luego compararemos con MHs basadas en BL y otros algoritmos de optimización presentes en la literatura.

B Objetivos

El principal objetivo de esta memoria es realizar un profundo estudio de los AGLs y MHs basadas en estos modelos, para analizar sus ventajas frente a los métodos clásicos de BL, y las correspondientes MHs basadas en BL. Para ello, nos proponemos:

- Definir un marco común para las diferentes propuestas de AGLs. Para ello, estudiaremos los principios de diseño de los algoritmos existentes y extraeremos sus propiedades. Posteriormente, categorizaremos a esta novedosa clase de MHs.
- Diseñar un nuevo modelo de AGL con codificación binaria, para su comparación con métodos de BL representativos para este tipo de codificación. En este punto, pretendemos:
  - Estudiar diferentes MHs basadas en BL con sus correspondientes variantes que usan el nuevo modelo de AGL sustituyendo al procedimiento de BL.
  - Analizar la relación que pueda generarse entre el nuevo modelo de AGL y los demás componentes de la MH, encargados de suplir diversificación al proceso de búsqueda.
- Diseñar *AGCRs especializados*, que promuevan intensificación fructífera o diversificación útil, es decir, *AGCRs Locales* o *AGCRs Globales*, respectivamente. Además, pretendemos diseñar una MH híbrida que combine un AGCR Local y un AGCR Global y obtenga, de forma simultanea, las ventajas de ambos.

C Resumen

Esta memoria está dividida en cuatro capítulos que describimos brevemente a continuación:

En el Capítulo 1, realizamos una introducción y breve repaso a las MHs más importantes presentadas en la literatura. Después, prestamos atención a dos tipos específicos de MHs, que serán el tema central del resto de la presente memoria:

- *Métodos de BL*, especializados en aportar intensificación refinando soluciones, es decir, obteniendo otras de muy alta calidad consumiendo pocos recursos.

- *AGs*, métodos que imitan los procesos evolutivos de la naturaleza, aplicándolos a un conjunto de soluciones.

En el Capítulo 2, realizamos un profundo estudio y propuesta del concepto de AGL, ofreciendo una visión global que nos permita entender sus fundamentos, propiedades, comportamiento y la forma en que se utilizan para resolver problemas de forma efectiva. Para ello, revisaremos los distintos modelos propuestos en la literatura, con intención de extraer las propiedades comunes. Comentaremos la evolución histórica del concepto. Y por último, los clasificaremos entre las MHs para encontrar sus similitudes y diferencias con otras familias de MHs.

En el Capítulo 3, diseñamos un AGL que realiza un proceso de refinamiento sobre soluciones iniciales, similar al realizado por una técnica de BL. Este nuevo modelo de AGL es un AG *estacionario* que emplea un *método de reemplazo por agrupamiento* (en inglés, *crowding method*) para favorecer la formación de nichos con soluciones de alta calidad en la población. Después, realiza un proceso de BL orientado hacia los nichos más cercanos a la solución a refinar. Posteriormente, estudiaremos las ventajas que introduce el uso del nuevo modelo de AGL en diferentes MHs basadas en BL, y la posible coordinación positiva que se genere entre ellos. Para alcanzar este objetivo, realizaremos un estudio empírico comparando los resultados de varias MHs basadas en BL que usan el AGL diseñado con sus correspondientes versiones clásicas, esto es, usando técnicas clásicas de BL. Nos interesaremos por tres MHs basadas en BL que nunca se han considerado para la aplicación de AGLs: BL con Multiarranque Aleatorio, BL Iterativa y Búsqueda de Vecindario Variable. Además, éstas introducen una coordinación gradual y controlada con el método de refinamiento, que nos permite analizar aspectos muy concretos para el diseño de MHs basadas en AGLs, y formular conclusiones sencillas.

En el Capítulo 4, diseñamos un modelo de *AGCRs especializados* que, mediante el uso de un tipo especial de operadores de cruce para problemas de optimización continua, y el ajuste preciso de sus parámetros de control, permite originar AGCRs Locales o AGCRs Globales. Concretamente, propondremos un *procedimiento de diferenciación sexual* que crea dos grupos de cromosomas a partir de la población del AG: 1) G_H con los N_H cromosomas de la población que pueden ser *progenitores hembra*; y 2) G_M con los N_M individuos que pueden seleccionarse como *progenitores macho* (N_H y N_M son dos parámetros del procedimiento). Después, dos mecanismos diferentes del AGCR se encargan de seleccionar el progenitor hembra y progenitor macho que producirán descendientes. Con la intención de obtener un método robusto frente a problemas con diferentes características, diseñaremos una MH híbrida que combine un AGCR Local y un AGCR Global, obteniendo simultaneamente las ventajas de ambos. Finalmente, compararemos la MH resultante con otras MHs, bien establecidas, para problemas de optimización continua.

Incluimos una sección de *Comentarios Finales* que resume los resultados obtenidos en esta memoria, presentando algunas conclusiones sobre éstos. Además,

se comentarán algunos aspectos sobre trabajos futuros que quedan abiertos en la presente memoria.

Por último, se incluyen tres apéndices donde se definen los problemas de prueba usados en los experimentos realizados (Apéndice A), los resultados numéricos de los algoritmos estudiados (Apéndice B) y los tests estadísticos usados para comparar los resultados de los algoritmos (Apéndice C).

Capítulo 1

Introducción a las Metaheurísticas y a los Algoritmos Genéticos

Muchos *problemas de optimización*, con importancia tanto teórica como práctica, consisten en la búsqueda de la mejor configuración de un conjunto de variables para alcanzar algunos objetivos. Más formalmente, un problema de optimización $P = (S, f)$ se define por:

- Un conjunto de variables $X = \{x_1, \ldots, x_n\}$.
- Los dominios de cada variable $D_1, \ldots, D_n$.
- Un conjunto de restricciones sobre las variables.
- Una *función objetivo* f que debe minimizarse (o maximizarse), donde:

$$f : D_1 \times \cdots \times D_n \rightarrow \Re. \tag{1.1}$$

El *espacio de búsqueda* del problema será el conjunto de todas las posibles asignaciones a las variables: $S = \{X = \{(x_1, v_1), \ldots, (x_n, v_n)\} \mid v_i \in D_i, X$ satisface todas las restricciones$\}$. Para resolver el problema de optimización se debe encontrar la solución $X^* \in S$ con mínimo (o máximo) valor de la función objetivo, esto es, $f(X^*) \leq f(X)\ \forall X \in S$. A X^* se le conoce como óptimo global de (S, f).

Por su naturaleza, estos problemas suelen dividirse en dos categorías: aquéllos en los que las soluciones se codifican con variables discretas y en los que las soluciones se codifican con variables reales. A los primeros, se les conoce como *problemas de optimización combinatora*, y a los segundos, *problemas de optimización continua*.

Los investigadores han propuesto muchos *algoritmos* para tratar los problemas de optimización. Éstos se clasifican en métodos *completos* o *aproximados*. Los primeros garantizan encontrar la solución óptima de un problema de tamaño

finito en un tiempo acotado, sin embargo, para algunos problemas, el tiempo necesario puede ser inviable. Concretamente, para muchos problemas, no se conoce un método completo que sea capaz de resolverlo en tiempo polinomial. Los métodos aproximados sacrifican la garantía de encontrar la solución óptima por obtener buenas soluciones en un tiempo significativamente reducido.

Las *Metaheurísticas* (MHs) (Blum y Roli, 2003; Glover y Kochenberger, 2003; Ribeiro y Hansen, 2002; Siarry y Michalewicz, 2008; Voß y otros, 1999) son métodos que combinan, en un nivel superior de abstracción, métodos aproximados, con intención de explorar eficaz y eficientemente S. Algunas propiedades importantes de las MHs son:

- Las MHs son estrategias para guiar el proceso de búsqueda en S.
- El objetivo es explorar eficientemente S con la intención de encontrar soluciones (casi) óptimas.
- Las MHs son algoritmos aproximados y normalmente no deterministas.
- Suelen incorporar mecanismos para evitar quedar atrapadas en regiones de S.
- Sus conceptos se describen en un nivel superior de abstracción.
- No dependen del problema concreto que se esté tratando.
- Pueden hacer uso de conocimiento específico de los problemas por medio de métodos heurísticos.

Para obtener soluciones de calidad, una MH debe producir un equilibrio apropiado entre los dos siguientes conceptos:

- *Diversificación*: se refiere a la capacidad de la MH de explorar las regiones de S. Su intención es la de hacer que la MH encuentre soluciones *fiables*, es decir, que no dependan de las condiciones de aplicación de la MH como pueden ser las características concretas del problema de optimización a tratar o particularidades del generador de números aleatorios que se utilice, entre otros.
- *Intensificación*: es la capacidad de explotar la información adquirida de las regiones de S. El objetivo es alcanzar soluciones *precisas* en las regiones visitadas por la MH, esto es, soluciones que, dentro de la región, presenten una alta calidad.

Las importancia de las MHs en el campo de los problemas de optimización es inmensa: desde 1995, y cada dos años, se viene celebrando la Conferencia Internacional de Metaheurísticas (en inglés, *Metaheuristics International Conference*); anualmente, y desde 2004, se celebra el taller Internacional sobre Metaheurísticas Híbridas (en inglés, *International Workshop on Hybrid Metaheuristics*); se encuentra en el ámbito de muchas revistas como *Journal of Heuristics* o *International Journal of Metaheuristics* (nueva revista que verá la luz el 1 de Enero de 2009), entre otras; es el tema de muchos números especiales en otras

revistas y se pueden encontrar numerosas aportaciones en congresos y revistas no especializadas.

En este capítulo, realizamos una introducción y breve repaso a las MHs más importantes presentadas en la literatura. Después, prestamos atención a dos tipos específicos de MHs, que serán el tema central del resto de la presente memoria:

- *Métodos de Búsqueda Local* (BL) (Blum y Roli, 2003): Son métodos especializados en refinar soluciones, obteniendo otras de alta calidad consumiendo pocos recursos.
- *Algoritmos Genéticos* (Goldberg, 1989b; Holland, 1975): Son métodos que imitan los procesos evolutivos de la naturaleza, aplicándolos a un conjunto de soluciones.

El capítulo se estructura como sigue: En la Sección 1.1, indicamos los criterios por los que podemos clasificar las distintas MHs. En la Sección 1.2, describimos las MHs más importantes. En la Sección 1.3, analizamos los factores que intervienen en el proceso de búsqueda realizado por la MH. En la Sección 1.4, describimos las técnicas de BL, métodos con gran relevancia práctica en el campo de las MHs. En la Sección 1.5, definimos el concepto *MH basada en BL* como aquélla que utiliza un método de BL para refinar las soluciones que produce. Finalmente, en la Sección 1.6, nos centramos en los Algoritmos Genéticos, principal objeto de estudio de la presente memoria.

1.1. Clasificación de las MHs

Las MHs se pueden clasificar atendiendo a diferentes características. Según la característica seleccionada, obtendremos una clasificación u otra, resultado de un punto de vista específico. A continuación, destacamos tres esquemas de clasificación de las MHs (Blum y Roli, 2003):

- *MHs inspiradas en la naturaleza y no inspiradas en ella*: Una forma intuitiva de clasificar las MHs es de acuerdo a los orígenes de éstas. Encontramos algoritmos inspirados en la naturaleza como los Algoritmos Genéticos o los *Algoritmos basados en Colonias de Hormigas* (Dorigo y otros, 1996; Dorigo y Stützle, 2004), entre otros; y algoritmos no inspirados en la naturaleza como la *Búsqueda Tabú* (Glover y Laguna, 1997) o la *BL Iterativa* (Lourenço y otros, 2003). Este tipo de clasificación presenta dos inconvenientes. Primero, muchos algoritmos recientes no encajan en ninguna de las dos clases (o, en algún sentido, podrían entrar en ambas). Segundo, a veces es difícil clasificar claramente un algoritmo como perteneciente a una de las dos clases. Por ejemplo, podríamos preguntarnos si el uso de mecanismos de memoria, por parte de Búsqueda Tabú, se inspira en la naturaleza.
- *MHs con una sola solución del espacio de búsqueda o poblaciones de soluciones*: Otra característica que podemos utilizar para clasificar las MHs

es el número de soluciones que procesan al mismo tiempo. Los algoritmos que trabajan sobre una sola solución se conocen como *Métodos basados en trayectorias*. Algunos ejemplos son los métodos de BL, Búsqueda Tabú, BL Iterativa o *Búsqueda de Vecindario Variable* (Hansen y Mladenović, 2002; Mladenovic y Hansen, 1997). Todos ellos comparten la propiedad de describir una trayectoria en el espacio de búsqueda a lo largo de la ejecución. Los *Métodos basados en poblaciones*, por contra, realizan procesos de búsqueda que describen la evolución de un conjunto de puntos en el espacio. Los Algoritmos Genéticos y Algoritmos basados en Colonias de Hormigas, entre otros, pertenecen a este grupo.

- *MHs con memoria o sin memoria*: Una característica muy importante para clasificar MHs es el uso que hacen del historial de la búsqueda realizada, esto es, si tienen memoria o no. Los algoritmos sin memoria realizan procesos de *Markov*, es decir, la siguiente acción a realizar viene determinada exclusivamente por la información del estado actual del proceso de búsqueda. Por otro lado, hay varias formas de hacer uso de la memoria. Normalmente, se diferencia entre memoria a *corto plazo* y memoria a *largo plazo*. La primera mantiene apuntados los movimientos más recientes del proceso de búsqueda. La segunda es normalmente una acumulación de parámetros sintéticos sobre la búsqueda. Muchas MHs incluyen, cada vez más, mecanismos de memoria. Podemos destacar Búsqueda Tabú como la primera en la que este concepto adquiere gran relevancia.

1.2. Descripción de las MHs Más Importantes

En esta sección, describiremos las MHs más importantes según la clasificación por número de soluciones que mantienen a la vez. Esta elección se motiva en el hecho de que la categorización de algoritmos es más clara. En la Sección 1.2.1, nos centramos en los *Métodos basados en trayectorias*. En la Sección 1.2.2, describimos brevemente varios *Métodos basados en poblaciones*

1.2.1. Métodos Basados en Trayectorias

En esta sección, describimos brevemente las MHs basadas en *trayectorias*. El término trayectoria proviene del hecho de que el proceso de búsqueda realizado por estos métodos se caracteriza por seguir una trayectoria en el espacio de búsqueda, en la que la siguiente solución puede encontrarse o no, en la vecindad de la solución actual.

El uso de *estructuras de vecindarios* es muy común en los métodos basados en trayectorias. Una estructura de vecindarios es una función $N : S \rightarrow 2^S$ que asigna a cada solución $X \in S$ un conjunto de vecinos $N(X) \subseteq S$. $N(X)$ se conoce como el vecindario de X. Un *óptimo local* con respecto a una estructura de vecindarios N es una solución $\widehat{X}$ tal que es mejor o equivalente a cualquier otra solución X de su vecindario ($\forall X \in N(\widehat{X}) : f(\widehat{X}) \leq f(X)$).

Es interesante destacar que el comportamiento final de un método basado en trayectorias es el resultado de la combinación de algoritmo, representación del

problema y problema concreto. De hecho, la representación del problema, junto con las estructuras de vecindario definen lo que se conoce como *campo de aptitud*; el algoritmo describe la estrategia usada para explorar este terreno; y, por último, las características del espacio de búsqueda concreto vienen determinadas por el problema concreto que se esté tratando.

A continuación, describiremos brevemente los algoritmos de BL. Posteriormente, abordaremos variantes que incorporan componentes diversificadoras para obtener soluciones más fiables.

Métodos de BL

Los métodos de BL son procedimientos que realizan pasos desde una solución actual a otra, en su vecindario, que sea mejor que la actual. El algoritmo para en cuanto encuentra un óptimo local. Estos métodos también se conocen como *procedimientos de mejora iterativa*.

El seudocódigo básico de un método de BL se muestra en la Figura 1.1. La función `mejora(`$N(X^a)$`)` puede realizar bien una mejora por *primer mejor*, o por *mejor*, vecino. En el primer caso, $N(X^a)$ se explora en algún orden determinado, normalmente aleatorio, y se escoge la primera solución que es mejor que X^a. En el segundo, se explora todo $N(X^a)$ y se escoge la mejor solución. Ambos métodos acaban en un óptimo local.

```
BL(f, N){

   X^a ← generaSolucionInicial();

   while(X^a no sea óptimo local){
      X^a ← mejora(N(X^a));
   }

   devolver X^a;
}
```

Figura 1.1: Esquema básico de un método de BL (Blum y Roli, 2003)

La aplicación solamente de métodos de BL sobre los problemas de optimización no suele ser satisfactoria debido a que la solución devuelta depende altamente de la estructura de vecindarios utilizada y la primera solución generada aleatoriamente. Por tanto, son métodos que producen soluciones poco fiables. El resto de MHs basadas en trayectorias pueden verse como variantes de los métodos de BL que añaden mecanismos para evitar quedar atrapados en óptimos locales y aumentar la fiabilidad de las soluciones finales. Estos mecanismos conllevan dar pasos hacia soluciones que pueden ser peores que la actual. Además, esto implica que sus condiciones de terminación no sean tan simples como alcanzar un óptimo local, sino que se tengan que utilizar otros métodos como máximo número de iteraciones, de evaluaciones de f, encontrar una solu-

ción X con $f(X)$ aceptable o sobrepasar un número máximo de iteraciones sin encontrar mejores soluciones, entre otros.

Métodos de BL con Multiarranque

Existen un conjunto de MHs basadas en trayectorias que se distinguen por la aplicación repetida de procesos de BL. La diferencia entre ellos suele ser la forma en que se generan nuevas soluciones iniciales a las que se le aplicará la técnica de BL. Estos métodos se consideran MHs basadas en trayectorias porque siempre realizan un paso desde una solución actual a otra, aunque ésta no tenga que pertenecer al vecindario de la solución actual utilizado por el método de BL. Algunos de los ejemplos más importantes son:

BL con Multiarranque Aleatorio (Boender y otros, 1982; Rinnoy Kan y Timmer, 1989) genera soluciones de forma totalmente aleatoria, del espacio de búsqueda, y les aplica un método de BL. Este método realiza un muestreo aleatorio de los óptimos del problema.

BL Iterativa (Lourenço y otros, 2003) aplica un método de BL a una solución inicial hasta encontrar un óptimo local, después, perturba la solución obtenida y aplica el método de BL a la nueva solución, y vuelve a repetir este proceso. Su idea es la de realizar un recorrido por el espacio de óptimos del problema. La importancia de la perturbación es obvia: perturbaciones demasiado leves pueden ser incapaces de escapar de la base de atracción del óptimo que se acaba de encontrar. Por otro lado, perturbaciones demasiado intensas pueden hacer que BL Iterativa se comporte de forma similar a BL con Multiarranque Aleatorio.

Búsqueda de Vecindario Variable (Hansen y Mladenović, 2002; Mladenovic y Hansen, 1997) recibe una solución inicial, un conjunto ordenado de vecindarios $(N^1, N^2, \ldots, N^{max})$ e inicializa el índice del vecindario actual $(k \leftarrow 1)$. Entonces, se repiten los siguientes pasos: 1) aplica un método de BL a la solución, 2) actualiza el vecindario actual y 3) genera una solución en el vecindario actual de la solución $(X' \in N^k(X))$. En el segundo paso, se realizará la asignación $k \leftarrow 1$ si en el paso anterior se ha encontrado una solución que mejora a la mejor solución hasta el momento; en otro caso, k se incrementa.

Descenso de Vecindario Variable (Hansen y Mladenović, 2002), como Búsqueda de Vecindario Variable, recibe una solución inicial y un conjunto ordenado de vecindarios (N^1, N^2, ..., N^{max}). Entonces, aplica un método de BL a la solución inicial, utilizando la estructura de vecindarios N^1. Al llegar a un óptimo local, aplica el método de BL a la solución obtenida pero usando la estructura de vecindarios N^2. Así hasta llegar a N^{max}. Después se repite el proceso con la última solución obtenida hasta alcanzar algún criterio de parada. Su idea principal es que diferentes estructuras de vecindarios dan lugar a diferentes campos de aptitud, y la solución que es un óptimo local con respecto a una estructura de vecindarios, no tiene por qué serlo para otra. Sin embargo, el óptimo global sí debe serlo para cualquier estructura de vecindarios.

Procedimientos de Búsqueda Ávidos Aleatorios y Adaptativos (más conocidos por su término en inglés *Greedy Randomized Adaptive Search Procedures*, GRASP) (Feo y Resende, 1995; Resende y Ribeiro, 2003) combinan una heurística constructiva con una técnica de BL. La heurística constructiva añade, paso a paso, componentes a la solución en construcción. La elección del siguiente elemento a añadir se escoge de forma aleatoria de una lista de candidatos, la cual está ordenada según algún criterio heurístico que beneficia a los componentes que, se suponen, producirán mejores soluciones. La combinación con un método de BL hace que este método realice un mestreo heurístico de los óptimos del problema.

BL Guiada (Voudouris y Tsang, 1999) pretende ayudar a futuros procesos de BL a alejarse de óptimos locales que se han localizado previamente. Para ello, BL Guiada modifica el campo de aptitud del problema con intención de hacer que el óptimo local actual no sea tan interesante. Para ello, se detectan las características de los óptimos locales visitados, y se utiliza la función f' definida como sigue:

$$f'(X) = f(X) + \lambda \sum_{i=1}^{m} p_i \cdot I_i(X), \tag{1.2}$$

donde $I_i(X)$ es uno si X tiene la característica i, cero en otro caso, p_i es el *parámetro de penalización* de la característica i, λ es el parámetro regulador y m es el número de características tenidas en cuenta.

Enfriamiento Simulado

Enfriamiento Simulado (Kirkpatrick y otros, 1983) es una de las MHs más antiguas que incluyen un mecanismo para evitar los óptimos locales. Su idea es la de permitir movimientos a soluciones peores que la actual, con la intención de escapar del óptimo local. La probabilidad de realizar movimientos a peores soluciones se reduce a lo largo de la ejecución.

En su definición clásica, en cada iteración genera, de forma aleatoria, una solución vecina a la actual ($X' \in N(X^a)$) y la acepta como la nueva solución actual según la distribución de probabilidad *Boltzmann* (se supone que se pretende minimizar f):

$$P(X^a \rightarrow X') = exp\left(-\frac{f(X') - f(X^a)}{T}\right) \tag{1.3}$$

donde T es la *temperatura* actual del sistema, parámetro que decrece a lo largo del proceso de búsqueda. Según esta distribución de probabilidad, al principio, la probabilidad de aceptar movimientos a peores soluciones es alta y gradualmente decrece, llegando a comportarse como una técnica de BL. El proceso es análogo al del enfriamiento de metales y cristales, donde se asume que se pueden alcanzar configuraciones de baja energía realizando un enfriamiento apropiado. Con respecto al proceso de búsqueda, el resultado es la combinación entre una exploración aleatoria y una técnica de BL. En la primera fase, la exploración presta poca atención a mejores soluciones, lo que permite realizar una exploración del espacio de búsqueda; este comportamiento cambia lentamente haciendo

que el proceso acabe en un óptimo (local). La probabilidad de aceptar soluciones peores se controla por dos factores: la diferencia en aptitud de las soluciones y la temperatura. Por un lado, a una temperatura fija, mayores diferencias $f(X') - f(X^a)$ obtienen menores probabilidades de aceptación. Por otro lado, a temperaturas mayores, mayor es la probabilidad de aceptar soluciones peores.

La elección de un *esquema de enfriamiento* apropiado es crucial para el rendimiento final del algoritmo. El esquema de enfriamiento define el valor de T para cada iteración k, $T_{k+1} = Q(T_k, k)$, donde $Q(T_k, k)$ es una función de la temperatura y el número de la iteración. Resultados teóricos indican que bajo condiciones particulares sobre el esquema de enfriamiento, el algoritmo converge en probabilidad a un óptimo global. Un esquema de enfriamiento que cumple la hipótesis de convergencia es la que sigue la ley logarítmica: $T_{k+1} = T_0/log(k + k_0)$, donde k_0 es una constante y T_0 es elevada (Aarts y otros, 1997). Desafortunadamente, los esquemas de enfriamiento que garantizan la convergencia al óptimo global no son viables en aplicaciones reales, porque son demasiado lentos para propósitos prácticos. Por ello, se suelen utilizar esquemas más rápidos. Uno de los más usados sigue la ley geométrica: $T_{k+1} = \alpha T_k$, donde $\alpha \in (0, 1)$, que produce una caída de la temperatura exponencial.

Búsqueda Tabú

La *Búsqueda Tabú* (Glover y Laguna, 1997) usa explícitamente la historia de la búsqueda con un doble objetivo: escapar de los óptimos locales e implementar una estrategia diversificadora. La Búsqueda Tabú más simple selecciona la mejor solución vecina $X' \in N(X^a)$, aunque sea peor que X^a. Para evitar ciclos, utiliza una *memoria a corto plazo*, implementada como una *lista tabú*, que pretende prohibir movimientos hacia soluciones ya visitadas. Por tanto, la lista tabú restringe $N(X^a)$. En cada iteración, se escoge la mejor solución de entre las permitidas y se actualiza la memoria a corto plazo. Normalmente, la lista tabú se implementa como una cola (se suele usar el término en inglés *first-in-first-out*, FIFO) de tamaño limitado. Al añadir un nuevo elemento, si la cola está llena, se elimina el elemento más antiguo. El algoritmo termina cuando se alcanza algún criterio de parada.

La longitud de la lista tabú (l), conocida como *tenencia tabú*, controla la memoria del proceso de búsqueda. Una tenencia tabú pequeña suele concentrar el proceso en una región pequeña del espacio de búsqueda. Por contra, una tenencia tabú mayor fuerza al proceso a explorar regiones más amplias, porque impide volver a visitar un mayor número de soluciones.

Normalmente, la lista tabú no almacena soluciones visitadas, ya que manejar una lista de soluciones es altamente ineficiente. En su lugar, suele almacenar atributos que representan componentes de las soluciones, movimientos, o diferencias entre soluciones. Esto es más eficiente, sin embargo, introduce una perdida de información, ya que prohibir un atributo suele significar evitar más de una solución. Por ello, es posible que soluciones buenas, no visitadas, se excluyan de $N(X^a)$. Para evitar este problema, se define el *criterio de aspiración*, que permite aceptar una solución aunque sea tabú. El criterio de aspiración

más ampliamente usado selecciona las soluciones que son mejores que la mejor visitada hasta el momento.

Además de la memoria a corto plazo, Búsqueda Tabú suele recoger información de todo el proceso y utilizarla para guiar la búsqueda. Esta clase de *memoria a largo plazo* suele referirse a cuatro principios diferentes:

- *Tiempo de los atributos*: En este caso, para cada solución o atributo, se almacena la iteración más reciente en que se visitó.
- *Frecuencia*: se almacena el número de veces que una solución o atributo se ha visitado. Esta información identifica las regiones del espacio que se han explorado más intensamente. Esta clase de información suele usarse para introducir diversificación al proceso de búsqueda.
- *Calidad*: se refiere a la extracción de información de la historia del proceso de búsqueda para identificar las componentes de las mejores soluciones.
- *Influencia*: se pretende detectar las decisiones, realizadas durante el proceso de búsqueda, que han sido las más críticas.

1.2.2. MHs Basadas en Poblaciones

Las *MHs basadas en poblaciones* mantienen, en cada iteración, un conjunto de soluciones, esto es, una población, en vez de una sola solución. Esto es un mecanismo natural para realizar una exploración amplia del espacio de búsqueda. Las MHs basadas en poblaciones se pueden dividir en dos categorías: *Algoritmos Evolutivos* y algoritmos que imitan el comportamiento de especies naturales, a éstos los llamaremos *Algoritmos de Optimización basados en Enjambres*.

Algoritmos Evolutivos

Los *Algoritmos Evolutivos* se inspiran en la capacidad de las especies de evolucionar y adaptarse a su entorno. En cada iteración aplican un conjunto de operadores a individuos de la población actual para generar los individuos de la próxima generación. A los operadores se les conoce normalmente como *operador de cruce* o *de recombinación*, cuando se crean nuevos individuos a partir de la información de más de un individuo de la población actual, y *operador de mutación*, cuando se generan nuevos individuos a partir de sólo uno anterior. La evolución de la población de individuos se guía según los valores de aptitud de cada individuo, que comúnmente es el valor de la función objetivo al evaluar al individuo como una solución al problema. Los individuos con mejor aptitud tendrán mayor probabilidad de producir descendientes. Esta idea se corresponde con el principio de la evolución natural: *supervivencia de los más adecuados*, que permite a la naturaleza adaptarse a entornos cambiantes.

Actualmente, existe una gran variedad de métodos que pertenecen a la clase de Algoritmos Evolutivos. Podemos destacar las siguientes familias: *Programación Evolutiva* (Fogel y otros, 1966; Fogel, 1995), *Estrategias de Evolución* (Beyer y Schwefel, 2002; Rechenberg, 1973, 1994), Algoritmos Genéticos (Goldberg, 1989b; Holland, 1975) y *Evolución Diferencial* (Price y otros, 2005; Storn

```
AlgoritmoEvolutivo(f){

   P ← GenerarPoblaciónInicial();
   Evaluar(P);

   while(no se alcance el criterio de parada){
      P' ← Recombinar(P);
      P'' ← Mutar(P');
      Evaluar(P'');
      P ← SeleccionarIndividuos(P'' ∪ P' ∪ P);
   }

   devolver la mejor solución encontrada;
}
```

Figura 1.2: Esquema básico de un Algoritmo Evolutivo (Blum y Roli, 2003)

y Price, 1997). Todas ellas siguen un proceso de evolución, que se asemeja al seudocódigo mostrado en la Figura 1.2 (Blum y Roli, 2003), existiendo ligeras diferencias entre ellas.

Comúnmente, los individuos de la población de un Algoritmo Evolutivo representan soluciones al problema, aunque también podrían representar soluciones parciales o conjuntos de soluciones, por ejemplo. Generalmente, las representaciones están estrechamente ligadas al problema que se está tratando, siendo las más utilizadas las cadenas de bits, números enteros, números reales o permutaciones de n números enteros. En el contexto de los Algoritmos Genéticos, los individuos mantienen el *genotipo*, la solución asociada se conoce como *fenotipo*. De esta forma, se diferencia entre la solución y su representación. La elección de una representación adecuada es crucial para el éxito del Algoritmo Evolutivo.

En cada iteración, el algoritmo debe decidir qué individuos deben sobrevivir y entrar en la población de la siguiente generación. Esto se realiza a través de un esquema de selección. En el *esquema generacional* los individuos de la siguiente generación se escogen exclusivamente de la población de descendientes (P' y P''). En este caso, se suele mantener la mejor solución encontrada hasta el momento entre diferentes iteraciones, lo cual se conoce como *elitismo*. Si es posible que individuos de la población actual puedan pasar a la siguiente población, entonces el esquema es *estacionario*. Aparte, la mayoría de los Algoritmos Evolutivos trabajan con tamaños de población fijos. Sin embargo, también se pueden utilizar poblaciones de tamaño variable. Una de las implementaciones más conocidas de Algoritmos Evolutivos con poblaciones de tamaño variable utiliza el valor de aptitud de los individuos para asignarle un tiempo de vida, medido en iteraciones (Arabas y otros, 1994).

Finalmente, una de las mayores dificultades en los Algoritmos Evolutivos es la de la *convergencia prematura* a soluciones de baja calidad. Este hecho se relaciona en gran medida con la pérdida de diversidad en la población del

algoritmo: cuando la mayor parte de los individuos del algoritmo se concentra en una región del espacio de búsqueda, resulta complicado explorar otras regiones y avanzar en el proceso de optimización.

Existe una familia especial de Algoritmos Evolutivos que sigue un esquema diferente, son los *Algoritmos de Estimación de Distribuciones* (Larrañaga y Lozano, 2001). Estos algoritmos crean modelos probabilísticos a partir de las soluciones más prometedoras para realizar una estimación sobre todo el espacio de búsqueda. Después, se usa el modelo para generar los individuos de la próxima población, realizando un muestreo del espacio de búsqueda acorde al modelo estimado. En cada iteración, el modelo se vuelve a estimar. La Figura 1.3 muestra el esquema básico de un Algoritmo de Estimación de Distribuciones.

```
AlgoritmoEstimacionDistribuciones(f){

   P ← GenerarPoblaciónInicial();
   Evaluar(P);

   while(no se alcance el criterio de parada){
      P' ← SeleccionarIndividuos(P);
      Modelo ← EstimarModeloDe(P');
      P'' ← MuestrearModelo(Modelo);
      Evaluar(P'');
      P ← SeleccionarIndividuos(P'' ∪ P);
   }

   devolver la mejor solución encontrada;
}
```

Figura 1.3: Esquema básico de un Algoritmo de Estimación de Distribuciones (Blum y Roli, 2003)

Algoritmos de Optimización Basados en Enjambres

La idea de los *Algoritmos de Optimización basados en Enjambres* es la de imitar el comportamiento inteligente que emerge a partir de la colaboración de individuos simples. Un ejemplo de esta inteligencia emergente es la capacidad de las hormigas de encontrar el camino más corto entre las fuentes de comida y sus hormigueros. Actualmente, existen dos principales familias bien diferentes de algoritmos que siguen esta idea, los *Algoritmos basados en Colonias de Hormigas* (Dorigo y otros, 1996; Dorigo y Stützle, 2004) y los *Algoritmos basados en Nubes de Partículas* (Eberhart y Kennedy, 1995; Kennedy y Eberhart, 2001).

Algoritmos Basados en Colonias de Hormigas. Las hormigas son insectos sociales que viven en colonias y que, gracias a su interacción colaboradora, son capaces de mostrar comportamientos complejos y resolver tareas difíciles desde el punto de vista local de una sola hormiga. Un aspecto muy interesante

del comportamiento de varias especies de hormigas es su habilidad para encontrar los caminos más cortos entre el hormiguero y las fuentes de comida.

Mientras caminan, las hormigas depositan una sustancia química olorosa denominada *feromona*. Si no hay presencia de rastros de feromona, las hormigas se mueven básicamente de forma aleatoria, sin embargo, en presencia de rastros de feromona, presentan una tendencia a seguirlos. Cuando existe más de un rastro, las hormigas realizan una elección probabilística condicionada por la cantidad de feromona que presenta cada rastro: los rastros con más feromona son más deseables. Debido a que las hormigas que regresan al hormiguero, vuelven a depositar feromona en el camino que realizaron, se realiza un proceso de autorefuerzo que da lugar a la formación de caminos con alta concentración de feromona. Este comportamiento permite a las hormigas identificar el camino más corto entre el hormiguero y la fuente de comida. Este procedimiento se complementa con el hecho de que la feromona se evapora tras cierto tiempo. De esta forma, los caminos menos prometedores pierden progresivamente su feromona porque cada vez los utilizan menos hormigas.

Los Algoritmos basados en Colonias de Hormigas se basan en una colonia de agentes, hormigas artificiales, que trabajan cooperativamente y se comunican a través de rastros de feromona artificial (Dorigo y Stützle, 2004). En cada iteración, cada hormiga realiza un recorrido por el grafo asociado al problema que se traduce en una solución. Cada arco del grafo representa las opciones que puede tomar una hormiga, los cuales tienen asociados dos tipos de información que guían la elección de la hormiga:

- *Información heurística*: mide la preferencia heurística de trasladarse desde el nodo i al nodo j. Se denota como η_{ij}. Depende del problema tratado y no se modifica a lo largo de la ejecución del algoritmo. Por ejemplo, en el problema del viajante de comercio se suele utilizar el costo de recorrer el arco del nodo i al nodo j.
- *Información del rastro de feromona*: mide la *preferencia aprendida* de realizar ese movimiento e imita la feromona natural que depositan las hormigas reales. Esta información se modifica a lo largo de la ejecución según las soluciones producidas por las hormigas. Mejores soluciones recibirán mayor cantidad de feromona. Esta información se denota por τ_{ij}.

La Figura 1.4 representa el esquema básico de un Algoritmo basado en Colonias de Hormigas. En la función `RealizarAccionesGlobales`, se suelen aplicar heurísticas no inspiradas en las colonias de hormigas naturales, entre otras, se suele aplicar un método de BL a las soluciones en C.

Actualmente, existe una gran variedad de Algoritmos basados en Colonias de Hormigas. Éstos se diferencian básicamente en las fórmulas y decisiones que se utilizan para actualizar los caminos de feromona, o para definir las probabilidades de las diferentes opciones que tiene una hormiga mientras construye una solución.

Algoritmos Basados en Nubes de Partículas. Los *Algoritmos Basados en Nubes de Partículas* (Eberhart y Kennedy, 1995; Kennedy y Eberhart, 2001)

```
AlgoritmoColoniaHormigas(f){

   while(no se alcance el criterio de parada){
      C ← ∅;

      Para cada hormiga h_i {
         C_i ← h_i construye una solución;
         evaluar(C_i)
         C ← añadir C_i;
      }

      ActualizarCaminosFeromona(C);
      RealizarAccionesGlobales(C);
   }

   devolver la mejor solución encontrada;
}
```

Figura 1.4: Esquema básico de un Algoritmo basado en Colonias de Hormigas (Dorigo y Stützle, 2004)

se inspiran en el comportamiento de las bandadas de aves o peces en su desplazamiento por un espacio. Resulta realmente destacable la sincronía y estética que aparece en el comportamiento de estos grupos de animales que, a menudo, cambian repentinamente de dirección, se dispersan y reagrupan. Este comportamiento es el resultado de acciones muy simples que realizan cada uno de los individuos pertenecientes a la bandada. Concretamente, se cree que el resultado se debe al esfuerzo que realiza cada individuo por mantener una distancia óptima con sus vecinos. Este comportamiento individual se repite en muchas otras especies como la humana. La idea subyacente es que el individuo que se desplaza con un grupo de éstos puede beneficiarse de los descubrimientos y experiencia de los otros miembros del grupo a la hora de evitar depredadores, buscar comida o pareja, optimizar parámetros como la temperatura, etc.

Los Algoritmos basados en Nubes de Partículas, como los basados en colonias de hormigas, son métodos en los que un grupo de agentes trabajan cooperativamente y realizan procesos de comunicación. Inicialmente, cada agente se sitúa aleatoriamente en un punto del espacio de búsqueda (una solución) con una velocidad aleatoria. En cada iteración, cada agente se desplaza de un punto del espacio de búsqueda a otro según una velocidad que describe su movimiento. Esta velocidad se modifica en cada iteración teniendo en cuenta información local y global de todo el grupo de agentes, por ejemplo, considerando puntos del espacio deseables que el agente, un agente vecino, o uno global, visitó.

El esquema básico de un Algoritmo basado en Nubes de Partículas se presenta en la Figura 1.5. Los diferentes Algoritmos basados en Nubes de Partículas utilizan diferentes fórmulas y decisiones para actualizar la velocidad de cada agente o la información global y local.

```
AlgoritmoNubesParticulas(f){

   Para cada agente a_i {
      Inicializar posición y velocidad de a_i;
   }

   while(no se alcance el criterio de parada){

      Para cada agente a_i {
         Actualizar velocidad según información local y global;
         a_i se desplaza en el espacio según su velocidad;
         Evaluar la nueva posición de a_i;
         Actualizar la información local y global según a_i;
      }
   }

   devolver la mejor solución encontrada;
}
```

Figura 1.5: Esquema básico de un Algoritmo basado en Nubes de Partículas (Eberhart y Kennedy, 1995)

1.3. Intensificación y Diversificación

La búsqueda realizada por una MH debe ser lo suficientemente *'inteligente'* para explotar intensamente las regiones del espacio de búsqueda con buenas soluciones, y moverse a regiones no exploradas cuando sea necesario. Los conceptos asociados a estos objetivos suelen llamarse *intensificación* y *diversificación*, aunque también se usan otros como *explotación* (relacionado con intensificación) y *exploración* (relacionado con diversificación):

- *Diversificación*: se refiere a la capacidad de la MH para explorar las diferentes regiones de S. Su intención es la de hacer que la MH encuentre soluciones *fiables*, es decir, que no dependan de las condiciones de aplicación de la MH como pueden ser las características concretas del problema de optimización a tratar o particularidades del generador de números aleatorios que se utilice, entre otros.

- *Intensificación*: es la capacidad de explotar la información adquirida de las regiones del espacio de búsqueda visitadas. El objetivo es alcanzar soluciones *precisas* en las regiones visitadas por la MH, esto es, soluciones que, dentro de la región, presenten una alta calidad.

Inicialmente, se consideraba que diversificación e intensificación eran factores opuestos, es decir, aumentar uno de ellos produce una disminución del otro. Actualmente, se reconoce que combinaciones apropiadas de ambos puede produ-

cir una mejora simultánea en ambos. Por tanto, para que una MH tenga éxito, debe establecer un equilibrio apropiado entre los dos.

> *Similarmente, como hemos apreciado, intensificación y diversificación no son nociones opuestas, sino que la mejor forma de cada una contiene aspectos de la otra, existiendo todo un espectro de alternativas* (Glover y Laguna, 1997).

Intensificación y diversificación pueden entenderse como efectos de los componentes que forman una MH. Blum y Roli (2003) introducen una visión unificada de ambas, definiendo como *componentes I&D* cualquier componente algorítmico o funcional que produce un efecto intensificador o diversificador sobre el proceso de búsqueda. En contraste con la visión de que los componentes tienen un efecto intensificador o diversificador, los componentes I&D tienen ambos efectos. Aquellos etiquetados como intensificadores serán componentes I&D con mayor tendencia a ofrecer intensificación que diversificación, y viceversa.

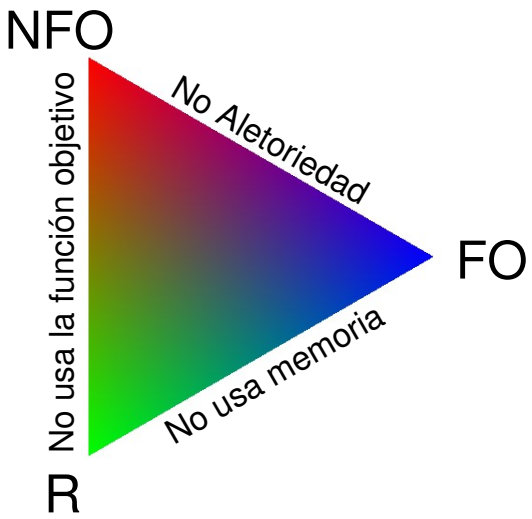


Figura 1.6: Espacio de los componentes I&D (Blum y Roli, 2003)

La Figura 1.6 muestra el espacio de componentes I&D, el cual es un triángulo donde las esquinas se corresponden con ejemplos extremos de componentes I&D. La esquina *FO* corresponde a los componentes I&D que se guían únicamente por la función objetivo del problema considerado. Un ejemplo muy cercano a esta esquina sería la elección del mejor vecino en un método de BL (ver Sección 1.2.1). La esquina *NFO* engloba los componentes I&D guiados por una o más funciones diferentes a la función objetivo, sin uso de aleatoriedad. Un ejemplo sería la creación de una solución, de forma determinista, según una memoria de frecuencias. La tercera esquina (A) es para los componentes I&D que se guían de forma completamente aleatoria. La creación de una solución totalmente aleatoria estaría en esta esquina. Según la descripción de las esquinas, está claro que FO corresponde a componentes I&D con un efecto intensificador muy elevado

y bajo nivel diversificador. Por otro lado, el segmento NFO-R corresponde a componentes I&D con un efecto altamente diversificador y poco intensificador.

Según esta vista unificada, la mayoría de componentes de las MHs pueden clasificarse como componentes I&D que se sitúan en algún lugar del espacio de la Figura 1.6. Algunos ejemplos son:

- *Aceptación de la siguiente solución actual en un método de BL*: Tanto si se escoge el primer mejor vecino, o el mejor de todos, el uso de la función objetivo es significativo. Estos métodos no utilizan memoria y, en primer mejor hay aleatoriedad mientras que en el mejor de todos no. Por tanto, en el caso del primer mejor vecino, el componente I&D se sitúa en algún punto del segmento A-FO, y en el caso del mejor vecino, muy cercano a la esquina FO.

- *Criterio de aceptación de Enfriamiento Simulado*: a una temperatura fija, el criterio de aceptación utiliza la función objetivo con cierto grado de aleatoriedad. Por tanto se trata de un componente I&D que se sitúa en el segmento A-FO. Si la temperatura es alta, este componente se localiza cerca de la esquina A, y si es baja, cerca de FO. Si además consideramos el esquema de enfriamiento, entonces el componente I&D se encuentra en algún punto interior del triángulo.

- *Regla de aceptación en Búsqueda Tabú*: Búsqueda Tabú selecciona el mejor vecino no presente en la lista tabú. Al seleccionar el mejor vecino, se utiliza la función objetivo, lo cual introduce intensificación. Al utilizar un mecanismo de memoria para evitar ciclos, se introduce diversificación. El equilibrio entre ambas dependerá de la longitud de la lista tabú. Por tanto se trata de un componente I&D situado en el segmento NFO-FO. Cuanto menor sea la longitud de la lista tabú, más cercano se encontrará de la esquina FO, y viceversa.

- *El reinicio en BL con Multiarranque Aleatorio*: Se trata de un componente totalmente aleatorio, por tanto se sitúa en la esquina A.

- *El reinicio en los Procedimientos de Búsqueda Ávidos Aleatorios y Adaptativos*: Son métodos que heurísticamente generan soluciones aleatorias. La heurística se guía por una función altamente relacionada con la función objetivo. Por tanto, este componente se sitúa en el segmento A-FO.

- *La selección en los Algoritmos Evolutivos*: Existen muchos tipos de esquemas de selección. La mayoría utilizan al menos, la función objetivo y aleatoriedad. Anteriormente, se catalogaba como un operador que provee intensificación, sin embargo, su aleatoriedad también puede proveer diversificación. Su situación estaría en el segmento A-FO. Por otro lado, existen esquemas de selección que pretenden mantener la diversidad en la población utilizando además otras funciones aparte de la función objetivo. En este caso, les corresponde un lugar dentro del triángulo.

- *El operador de recombinación en los Algoritmos Evolutivos*: En sus inicios, se consideraba como un operador que aporta diversificación, ya que genera

nuevas soluciones. Actualmente se considera que también aporta intensificación, sobre todo cuando utilizan algún tipo de heurística durante la generación de soluciones. Se situaría dentro del triángulo.

1.4. Métodos de BL

Los *métodos de BL* son MHs diseñadas para obtener soluciones precisas a partir de soluciones iniciales dadas, es decir, refinar soluciones. Para ello, el método de BL empieza desde una solución específica, e iterativamente intenta reemplazar la *solución actual* (X^a) por una mejor de un vecindario de ésta ($N(X^a)$), definido apropiadamente. Cuando el método de BL no encuentra una solución $X \in N(X^a)$ mejor que X^a, para y devuelve X^a, la cual se conoce como *óptimo local* con respecto a la estructura de vecindarios N.

Las técnicas de BL son intuitivas, flexibles y generalmente más fáciles de implementar que los algoritmos exactos. Sin embargo, su mayor interés viene por el hecho de que éstas pueden explorar eficaz y eficientemente la *base de atracción* de óptimos locales, encontrando un óptimo con alto grado de precisión y consumiendo pocos recursos computacionales, esto es, tienen una marcada tendencia a promover intensificación. De hecho, las técnicas de BL son un componente fundamental de muchas MHs, estado del arte de muchos problemas (Blum y Roli, 2003; Glover y Kochenberger, 2003; Ribeiro y Hansen, 2002; Siarry y Michalewicz, 2008; Voß y otros, 1999). Por otro lado, en el campo teórico también se ha realizado un progreso considerable (Angel, 2006; Ausiello y Protasi, 1995; Johnson y otros, 1988; Schuurman y Vredeveld, 2007).

En la Sección 1.4.1, describimos los componentes de los métodos de BL. En la Sección 1.4.2, comentamos un tipo de métodos de BL que utilizan una estructura de vecindarios especial, los procedimientos *Kopt*. En la Sección 1.4.3, mencionamos algunas características especiales de los métodos de BL usados para tratar problemas de optimización continua.

1.4.1. Componentes de una Técnica de BL

Un método de BL se define en base a dos elementos, una estructura de vecindarios y un mecanismo de transición (Aarts y van Laarhoven, 1992):

- Una *estructura de vecindarios* es una función $N : S \rightarrow 2^{S}$[1] que asigna a cada solución del espacio de búsqueda ($X \in S$) un conjunto de vecinos ($N(X) \in S$), el cual representa a un conjunto de soluciones que se encuentran *cerca* de X, en algún sentido.
- El *mecanismo de transición* define la estrategia del algoritmo para realizar la búsqueda entre soluciones vecinas.

A continuación, desarrollamos ambos conceptos.

[1] 2^S es el conjunto de todas las particiones de S

Definición de $N(X^a)$

La elección de la estructura de vecindarios a usar es esencial. Ésta depende altamente del problema que se esté tratando. $N(X^a)$ define el conjunto de soluciones que pueden alcanzarse a partir de X^a en un solo paso del método de BL. Hay dos factores importantes a tener en cuenta a la hora de diseñar $N(X^a)$:

- La *cardinalidad de* $N(X^a)$ indica el número de soluciones que pueden alcanzarse desde X^a. Cardinalidades bajas hacen que la iteración del método de BL sea más rápida que con cardinalidades altas. De esta forma, pueden realizar un mayor número de pasos consumiendo pocos recursos. Por otro lado, cardinalidades altas dan mayor libertad a la técnica de BL para elegir su próximo movimiento, lo cual puede resultar en pasos a mejores soluciones que con cardinalidades bajas.
- La *distancia entre dos soluciones*, entendida entre las soluciones del problema, y no entre sus codificaciones, debe ser relativamente pequeña. La idea es que soluciones vecinas deben compartir propiedades. De otra forma, el proceso de búsqueda se vuelve aleatorio.

Típicamente, una estructura de vecindarios no se define enumerando explícitamente el conjunto de posibles soluciones accesibles, sino implícitamente definiendo el conjunto posible de *transformaciones* que pueden aplicarse a X^a. La *inversión de un bit* suele usarse en problemas cuyas soluciones se codifican como cadenas binarias, mientras que existen un enorme conjunto de estructuras de vecindarios cuando las soluciones se codifican con cadenas numéricas o cuando representan secuencias. Charon y Hudry (2001) y Walser y Kautz (1999) revisan varias estructuras de vecindarios, diseñadas según el tipo de codificación que usen las soluciones del problema. Este tipo de vecindarios se conocen como *independientes del contexto*. Por otro lado, $N(X^a)$ también se puede definir acorde al problema concreto, no sólo a la codificación usada, que se esté tratando. En este caso, hablamos de vecindarios *dependientes del contexto*. En Angel (2006), Crauwels y otros (1998), Larrañaga y otros (1999), Ruiz y Maroto (2005) y Valenzuela (2001) se pueden encontrar varios ejemplos.

El Mecanismo de Transición

Por otro lado, el mecanismo de transición especifica la estrategia que se sigue a la hora de explorar las soluciones en $N(X^a)$. Los dos más importantes son:

- *Exploración exhaustiva*: Consiste en examinar todas las posibles transformaciones sobre X^a para seleccionar aquélla que produce la mayor mejora posible. Esto significa que se examinan todas las soluciones de $N(X^a)$ en busca de la mejor. Este mecanismo de transición genera una técnica de BL ampliamente conocida como *BL del mejor*, o también del *máximo gradiente* o *pendiente* (Blum y Roli, 2003; Davis, 1991a; Dunham y otros, 1963).
- *Exploración aleatoria*: Consiste en probar, según un orden aleatorio, las transformaciones posibles sobre X^a, y seleccionar la primera que produce

una solución mejor que X^a. *BL del primer mejor* (Blum y Roli, 2003; Davis, 1991a; Dunham y otros, 1963) es precisamente un método ampliamente conocido que sigue este mecanismo de transición.

Cuando no hay una solución $X \in N(X^a)$ mejor que X^a, ésta se conoce como un *óptimo local* según la estructura de vecindarios N. Esta condición define el criterio de parada del método de BL.

1.4.2. Estructura de Vecindarios Kopt

Existe una estructura de vecindarios especial, conocida como *Kopt* (Katayama y Narihisa, 2001; Kernighan y Lin, 1970; Lin y Kernighan, 1973; Merz, 2000; Merz y Katayama, 2004), que pretende explorar, en tiempo polinomial, vecindarios de muy alta cardinalidad (exponencial). La idea es usar un *procedimiento de exploración iterativa del vecindario* (PEIV) como el operador que genera $N(X^a)$. En vez de explorar un vecindario amplio, PEIV explora inicialmente un vecindario muy reducido de X^a, y selecciona la transformación t^1 que produce la solución X^1; después, explora un vecindario muy reducido de X^1, considerando todas las posibles transformaciones excepto t^1, y obteniendo X^2; y así en adelante, hasta agotar las transformaciones disponibles. Para la siguiente iteración del método de BL, se escoge como X^a a la mejor solución X^i producida, siempre que sea mejor que X^a. Con este tipo de vecindarios, se han propuesto los siguientes dos métodos de BL:

- *BL Kopt* (Kernighan y Lin, 1970; Lin y Kernighan, 1973; Merz, 2000): cuyo PEIV realiza una exploración exhaustiva de $N(X^{i-1})$, eligiendo como X^i la mejor solución, aunque sea peor que X^{i-1}.

- *BL Primer Kopt* (BL-PKopt) (Katayama y Narihisa, 2001; Merz y Katayama, 2004): cuyo PEIV realiza una exploración aleatoria de $N(X^{i-1})$, eligiendo como X^i la primera que sea mejor que X^{i-1}. Si no existiese una solución mejor que X^{i-1} tras haber explorado todo $N(X^{i-1})$, elegiría la que produce el menor detrimento.

1.4.3. Métodos de BL para Problemas de Optimización Continua

Existen dos diferencias importantes entre los métodos de BL utilizados para problemas de optimización combinatora y los utilizados para problemas de optimización continua:

- La cardinalidad de cualquier estructura de vecindarios es infinita en problemas de optimización continua.

- Resulta difícil medir la distancia entre dos soluciones vecinas, ya que, en el peor caso, cualquier estructura de vecindarios puede representar una copia reducida de todo el espacio de búsqueda (Mandelbrot, 1983).

La primera característica, además de afectar a la condición de parada (no se puede comprobar si existe una solución en $N(X^a)$ mejor que X^a), afecta al mecanismo de transición, haciendo inviable realizar una exploración exhaustiva. Por otro lado, algunos métodos de BL para optimización continua pueden hacer uso del *gradiente* de la función objetivo para guiar la exploración aleatoria (Gilbert y Nocedal, 1992). Entonces estaríamos hablando de un mecanismo de transición diferente, la *exploración guiada por grandiente* en contraste a la exploración aleatoria ciega. Cuando el gradiente de la función no se conoce, algunos métodos los sintetizan numéricamente mediante diferenciación finita.

La segunda diferencia hace difícil el diseñar la estructura de vecindarios N debido a que pequeños cambios sobre X^a pueden ser aún demasiado grandes. Para resolver este problema, cada método de BL toma alguna de las siguientes opciones:

- *Técnicas guiadas por gradiente*: La información del gradiente también puede usarse para determinar qué distancia debiera haber entre X^a y las nuevas soluciones a generar. Si el gradiente es pronunciado, se espera obtener mejores soluciones '*lejos*' de X^a; si el gradiente es pequeño, se debe explorar cerca de X^a. Podemos encontrar un gran número de métodos de BL que siguen esta idea (Gilbert y Nocedal, 1992; Press y otros, 1989; Schwefel, 1995; Siddall, 1982; Yuret y de la Maza, 1993).

- *Métodos con pasos adaptativos* (Horner y otros, 1961; Rosenbrock, 1960; Solis y Wets, 1981): X^a se modifica con una *longitud de paso* predeterminada, la cual se reduce cuando la técnica de BL no encuentra mejores soluciones. Schwefel (1995) y Swann (1964) revisan algunas de estas técnicas de BL.

- *Métodos de BL basados en las Estrategias de Evolución (1,λ) y (1+λ)* (Guanqi y Shouyi, 2003; Schwefel, 1995): se genera un conjunto de λ soluciones, copiando X^a y sumando un vector de mutación Z_i, para cada nueva solución (Z_i suele seguir una distribución normal de media cero y desviación típica σ). Una característica importante de estos métodos es su capacidad de autoadaptar los valores de σ, considerándolos como componentes de las soluciones.

- *Técnicas basadas en conjuntos de soluciones* (Nelder y Mead, 1965): Algunos métodos crean un conjunto de soluciones del espacio de búsqueda. Las nuevas soluciones se generan más o menos cerca de ellas según su distribución actual. Después, las nuevas soluciones suelen reemplazar las anteriores, si son mejores. Típicamente, en cada iteración se genera una única solución que reemplaza a la peor del conjunto.

Existe otra familia de métodos de BL para la optimización continua que trabajan sobre una codificación binaria (o *Gray*) de las variables de decisión, de forma que se pueden aplicar métodos de BL para optimización combinatoria. En particular, la técnica de BL *Quad* (Whitley y Rowe, 2005) aprovecha la codificación *Gray* del problema para realizar una búsqueda binaria del óptimo, en cada dimensión.

1.5. Metaheurísticas Basadas en BL

La idea básica de las *MHs basadas en BL* reside en diferenciar bien las necesidades de intensificación y diversificación, y asignarselas a componentes diferentes. Con esta intención, las MHs basadas en BL combinan los altos niveles de intensificación ofrecidos por los procedimientos de BL con componentes encargados de la diversificación, esto es, componentes que hacen énfasis en la exploración global del espacio de búsqueda. Por un lado, los componentes diversificadores aseguran que se muestreen soluciones iniciales en diferentes regiones del espacio de búsqueda y, por el otro, la técnica de BL obtiene las mejores soluciones dentro de esas regiones. El seudocódigo mostrado en la Figura 1.7 representa el esquema básico de una MH basada en BL, donde $parametros_c$ representa los parámetros del componente diversificador, *historia* puede representar mecanismos de memoria que utilice la MH basada en BL y $parametros_{bl}$, los parámetros del método de BL. Básicamente, un *método generador* produce soluciones iniciales que posteriormente se refinan con el método de BL.

```
MH-BL(BL, parametros_c, parametros_bl){

   X' ← generaSolucionInicial();
   X^a ← X';

   while(no se alcance la condición de parada){
      X' ← generaSolucion(X', X^a, parametros_c, historia, ...);
      X' ← BL(X', parametros_bl); //algoritmo de refinamiento;
      actualizar X^a si X' es mejor;
   }

   devolver X^a;
}
```

Figura 1.7: Esquema básico de una MH basada en BL (Marti, 2003)

Podemos encontrar muchos ejemplos de MHs basadas en BL, lo cual pone de manifiesto la relevancia de este tipo de métodos. Además, algúnos constituyen el estado del arte de varios problemas de optimización. Entre estos métodos encontramos a *BL con Multiarranque Aleatorio* (Boender y otros, 1982; Rinnoy Kan y Timmer, 1989), *Procedimientos de Búsqueda Ávidos Aleatorios y Adaptativos* (Feo y Resende, 1995; Resende y Ribeiro, 2003), *Algoritmos basados en Colonias de Hormigas* (Dorigo y otros, 1996; Dorigo y Stützle, 2004), *BL Iterativa* (Lourenço y otros, 2003), *Búsqueda de Vecindad Variable* (Hansen y Mladenović, 2002; Mladenovic y Hansen, 1997), *Algoritmos Meméticos* (Hart y otros, 2004; Moscato, 1999) y *Búsqueda Dispersa* (Glover, 1999; Laguna, 2003).

A continuación, describimos varias MHs basadas en BL. Pondremos especial atención a tres de ellas, en las que claramente se diferencia un componente que aporta, de forma controlada, diversificación al proceso de búsqueda: BL con Multiarranque Aleatorio, BL Iterativa y Búsqueda de Vecindario Variable.

Además, estas tres MHs basadas en BL tendrán un papel importante en el estudio experimental del Capítulo 3. La Sección 1.5.1 describe BL con Multiarranque Aleatorio. La Sección 1.5.2 se centra en BL Iterativa. La Sección 1.5.3 describe el método de Búsqueda de Vecindario Variable. Por último, la Sección 1.5.4 repasa brevemente otras MHs basadas en BL.

1.5.1. BL con Multiarranque Aleatorio

BL con Multiarranque Aleatorio (Boender y otros, 1982; Rinnoy Kan y Timmer, 1989) pretende hacer un muestreo aleatorio de los óptimos de un problema. Para ello, simplemente genera soluciones iniciales aleatoriamente repartidas por el espacio de búsqueda, las cuales se optimizan posteriormente por la técnica de BL. BL con Multiarranque Aleatorio es la MH basada en BL más simple. El método generador de soluciones no interacciona con el método de BL, ya que su comportamiento no depende del rendimiento del método de BL. Esta propiedad es muy interesante a la hora de comparar diferentes algoritmos de refinamiento.

La Figura 1.8 muestra el comportamiento de BL con Multiarranque Aleatorio y la Figura 1.9, su seudocódigo. En cada iteración, se genera una solución inicial X_i, el algoritmo de refinamiento obtiene la solución X_i' (flecha continua); en la siguiente iteración se genera una nueva solución inicial aleatoria X_{i+1}, independiente del estado del proceso de búsqueda, que se refina obteniendo X_{i+1}', y así en adelante. La solución devuelta al final de la ejecución es la mejor solución encontrada en todo el proceso.

1.5.2. BL Iterativa

BL Iterativa (Lourenço y otros, 2003) es una MH basada en BL que pretende realizar un recorrido por el espacio de óptimos de un problema. Para ello, empieza con una solución inicial aleatoria X_0 que se optimiza mediante un método de BL, obteniendo X_0'. Después, se obtiene una nueva solución inicial X_i aplicando una *perturbación* sobre una solución obtenida anteriormente, que se optimiza mediante el mecanismo de BL para obtener X_i'. Este proceso se repite hasta alcanzar la condición de parada. La Figura 1.10 representa la iteración básica de BL Iterativa donde las flechas continuas indican la aplicación del método de BL, y las discontinuas, la perturbación. La Figura 1.11 representa su seudocódigo.

El operador de perturbación es un aspecto clave en BL Iterativa, debido a que le permite escapar de la base de atracción del óptimo local actual, y así alcanzar nuevas soluciones. Éste introduce un equilibrio determinado entre intensificación y diversificación: se obtiene intensificación cuando se aplican perturbaciones leves y diversificación cuando las perturbaciones son más intensas. Como norma general, el operador de perturbación debiera tener una intensidad lo suficientemente alta como para que la siguiente invocación del método de BL no devuelva el mismo óptimo local. Por otro lado, si fuese demasiado intenso, el proceso de búsqueda sería equivalente al de BL con Multiarranque Aleatorio. Según Lourenço y otros (2003): '*Una buena perturbación transforma una excelente solución en un excelente punto de partida para una técnica de BL*'.

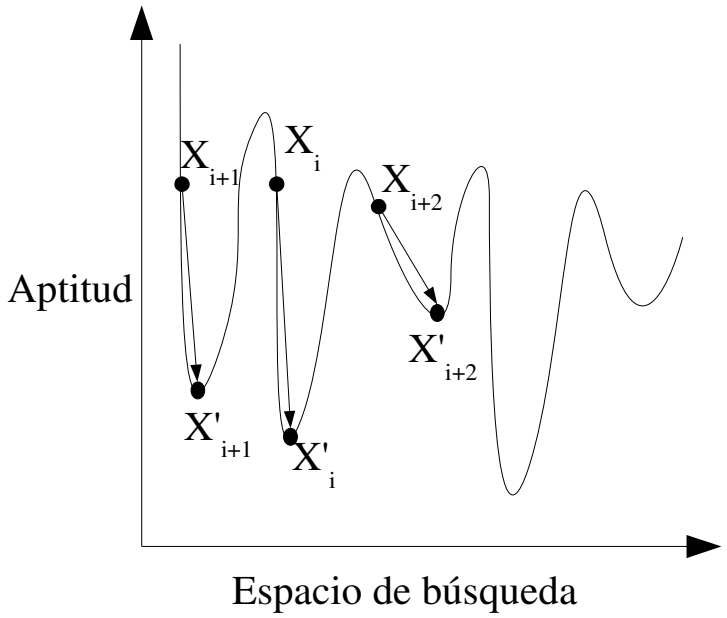


Figura 1.8: Comportamiento de BL con Multiarranque Aleatorio

```
BLMultiarranqueAleatorio(BL, parametros_bl){

   X' ← generaSolucionAleatoria();
   X^a ← BL(X', parametros_bl);

   while(no se alcance la condición de parada){
      X_i ← generaSolucionAleatoria();
      X'_i ← BL(X_i, parametros_bl);
      actualizar X^a si X'_i es mejor;
      i ← i + 1;
   }

   devolver X^a;
}
```

Figura 1.9: Seudocódigo de BL con Multiarranque Aleatorio

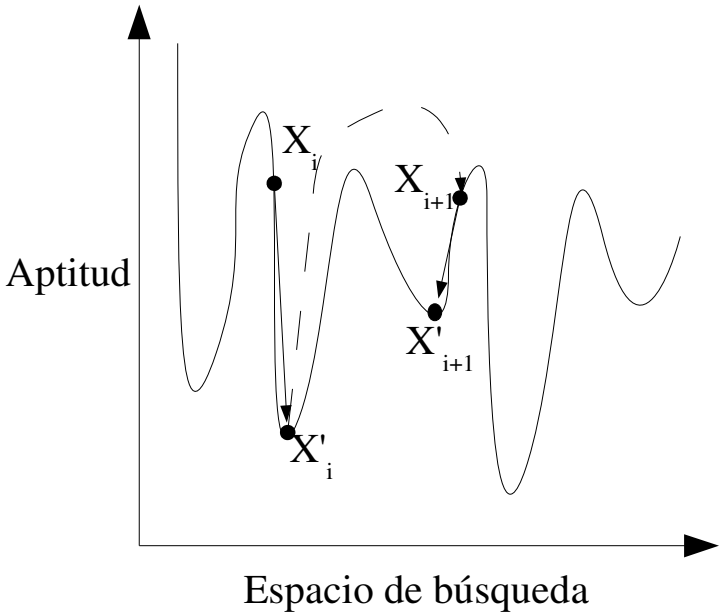


Figura 1.10: Comportamiento de BL Iterativa

```
BLIterativa(BL, parametros_bl, parametros_bli){

   X' ← generaSolucionAleatoria();
   X^a ← BL(X', parametros_bl);

   while(no se alcance la condición de parada){
      X_i ← aplicaPerturbación(X^a,parametros_bli);
      X'_i ← BL(X_i, parametros_bl);
      actualizar X^a si X'_i es mejor;
      i ← i + 1;
   }

   devolver X^a;
}
```

Figura 1.11: Seudocódigo de BL Iterativa (Lourenço y otros, 2003)

En la literatura, se han propuesto diferentes operadores de perturbación (Lourenço y otros, 2003), como son la *perturbación estándar* de un porcentaje dado de componentes de la solución, la aplicación de un método de optimización sobre un subconjunto de las componentes de la solución o la aplicación de una Búsqueda Tabú, entre otros. En el caso de la perturbación estándar, el usuario puede fijar, en cierto modo, la diversificación que se introduce en el proceso de búsqueda, ajustando el parámetro que controla el porcentaje de componentes a perturbar y así, la *intensidad de la perturbación*. Por ejemplo, en codificación binaria, la perturbación estándar cambia el valor de cada componente de una solución de acuerdo a una probabilidad dada.

BL Iterativa, a diferencia de BL con Multiarranque Aleatorio, produce interacciones entre el método generador y el algoritmo de refinamiento. La razón es que las soluciones iniciales, generadas por el operador de perturbación, dependen directamente de las alcanzadas por el algoritmo de refinamiento, mientras que en BL con Multiarranque Aleatorio no.

1.5.3. Búsqueda de Vecindario Variable

En el método de *Búsqueda de Vecindario Variable* (Hansen y Mladenović, 2002) se tiene un conjunto de vecindarios ($\{N^1, \ldots, N^{max}\}$), normalmente anidados (de menor a mayor amplitud), y una variable (k) que indica el vecindario actual (N^k). Cuando comienza el algoritmo, se inicializa $k \leftarrow 1$ y se aplica BL a una solución generada aleatoriamente. Posteriormente, se genera una solución dentro del vecindario actual de la mejor solución encontrada ($N^k(X^a)$), y se le aplica BL. Si la solución obtenida es mejor a la mejor hasta el momento, entonces X^a se actualiza y se inicializa $k \leftarrow 1$; en otro caso, k se incrementa. El proceso se repite hasta alcanzar la condición de parada. La Figura 1.12 representa el comportamiento de este método y la Figura 1.13 su seudocódigo, donde $mod(x, y)$ representa el resto de la división x/y. Dada X_i, el método de BL obtuvo X_i'; después, el operador de generación produjo la solución $X_{i+1} \in N^k(X_i')$, cuyo refinamiento volvió a obtener X_i' (no se mejoró la solución); por tanto, en la siguiente iteración, Búsqueda de Vecindario Variable utiliza el siguiente vecindario ($N^{k+1}(X_i')$) intentando salir de esa base de atracción.

La idea del método de Búsqueda de Vecindario Variable es la de aportar la diversificación justa para que el siguiente refinamiento pueda encontrar una solución mejor a la mejor hasta el momento. Por tanto, se trata de una MH basada en BL que aporta una diversificación adaptativa, a diferencia de BL con Multiarranque Aleatorio y BL Iterativa con perturbación estándar.

Normalmente, los vecindarios están anidados ($N^i(X) \subset N^{i+1}(X)$), sin embargo, Liberti y Dražić (2005) proponen usar *vecindarios de intersección vacía* ($N^{i'}(X) = \{N^i(X) \setminus N^{i-1}(X)\}$). Al generar soluciones en $N^{i'}(X)$, se produce un muestreo sobre los vecindarios originales más equitativo, debido a que, al generar una solución de $N^i(X)$, hay una alta probabilidad de que la solución también pertenezca a $N^{i-1}(X)$.

Por último, los vecindarios utilizados en este método se suelen implementar a través de operadores de perturbación que modifican X^a. Según la intensidad con que los operadores perturben X^a, tendremos vecindarios más o menos am-

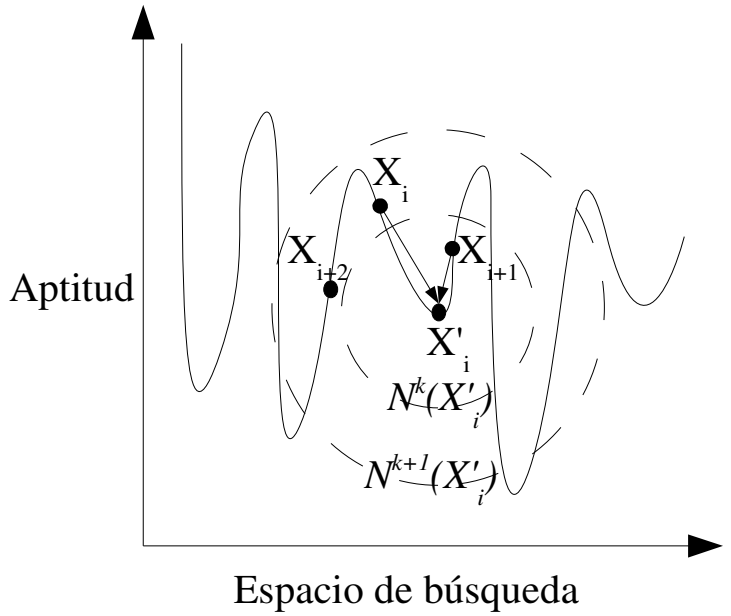


Figura 1.12: Comportamiento de Búsqueda de Vecindario Variable

```
BúsquedaVecindarioVariable(BL, parametros_bl, N^1,...,N^max){

   X' ← generaSolucionAleatoria();
   X^a ← BL(X', parametros_bl);
   k ← 1;

   while(no se alcance la condición de parada){
      X_i ← generaSoluciónEn(N^k(X^a));
      X'_i ← BL(X_i, parametros_bl);
      i ← i + 1;

      if(X'_i es mejor que X^a){
         X^a ← X'_i;
         k ← 1;
      }
      else{
         k ← mod(k, max) + 1;
      }
   }

   devolver X^a;
}
```

Figura 1.13: Seudocódigo de Búsqueda de Vecindario Variable (Hansen y Mladenović, 2001)

plios: vecindarios menores estarán relacionados con intensidades pequeñas, y vecindarios mayores con intensidades grandes. En el caso de soluciones binarias, se puede aplicar el operador de perturbación estándar que cambia el valor de las componentes de la solución (de 0 a 1, y viceversa) según una probabilidad dada (intensidad de la perturbación). Es interesante notar que este operador produce vecindarios que siguen la idea de intersección vacía, esto es, dado un conjunto ordenado de intensidades, habrá un grado bajo de solapamiento entre un muestreo elevado de $N^i(X)$ y otro de $N^{i+1}(X)$.

1.5.4. Otras MHs Basadas en BL

En la Sección 1.2.1, describimos ya, algunas MHs basadas en BL que también se consideran MHs basadas en trayectorias. A continuación, describimos brevemente varias MHs basadas en poblaciones que consideran, desde sus inicios, la aplicación de métodos de BL:

- *Algoritmos basados en Colonias de Hormigas* (Dorigo y otros, 1996; Dorigo y Stützle, 2004): Estos algoritmos se inspiran en la capacidad de varias especies de hormigas para encontrar el camino más corto entre el hormiguero y una fuente de alimento. En estos modelos suele haber dos módulos bien diferenciados: en el primero, se encuentran las hormigas, encargadas de proponer soluciones al problema mediante técnicas constructivas; en el segundo, un *demonio* se encarga de realizar actualizaciones de las estructuras de información de forma global, entre las que suele estar la aplicación de un método de BL a las soluciones obtenidas por las hormigas.

- *Algoritmos Meméticos* (Hart y otros, 2004; Moscato, 1999): Su principal idea es la de combinar, sobre una población de agentes, procesos de cooperación y competición, presentes en los Algoritmos Evolutivos, con procesos de mejora individual. Una amplia representación de los Algoritmos Meméticos son Algoritmos Evolutivos en los que se aplica un proceso de BL para refinar los individuos de la población (Krasnogor y Smith, 2005).

- *Búsqueda Dispersa* (Glover, 1999; Laguna, 2003): es un tipo específico de Algoritmo Memético en que el que las soluciones se construyen por *combinación lineal* de las presentes en un *conjunto de referencia*. Se aplican métodos de BL siempre que se genera una nueva solución.

- *Otros métodos*: métodos como Evolución Diferencial (Price y otros, 2005) y Algoritmos basados en Nubes de Partículas (Kennedy y Eberhart, 2001) pueden usar métodos de BL para refinar sus soluciones. De hecho, en algunos trabajos se aplican estos algoritmos junto a una técnica de BL con este fin (Noman y Iba, 2008; Petalas y otros, 2007).

1.6. Algoritmos Genéticos

Los organismos vivos poseen una consumada destreza en la resolución de problemas, esta habilidad la obtienen a través de la *evolución*. En casi todos los organismos, la evolución se produce mediante dos procesos elementales:

- la *selección natural*, que determina qué miembros de la población sobrevivirán hasta reproducirse, y
- la *reproducción*, que garantiza la mezcla y recombinación de sus genes entre la descendencia.

El efecto conjunto de estos dos procesos marca el éxito de 3 billones de años de vida sobre el planeta: muchos organismos son capaces, en sucesivas generaciones, de *adaptarse* a cambios en su forma de vida, e incluso en su estructura, para combatir cualquier agente que hiciera peligrar su supervivencia.

A través de la historia, la naturaleza ha sido una fuente inagotable de inspiración para el desarrollo técnico y científico, el cual, a cambio, ha contribuido a una mejor comprensión de la propia naturaleza. La idea de diseñar algoritmos, para resolver problemas, basándose en los principios de la evolución de los organismos naturales nace independientemente en ambos lados del atlántico hace aproximadamente tres décadas. En Estados Unidos, J. Holland estudia el fenómeno de la adaptación tal y como ocurre en la naturaleza y desarrolla mecanismos para incorporarla en un sistema computerizado. Holland (1975), en su libro *Adaptación en sistemas naturales y artificiales*, propone una clase de algoritmos adaptativos, ideados como una abstracción de la evolución para resolver problemas prácticos y para servir de modelos computacionales de sistemas naturales evolutivos, métodos que posteriormente recibieron el nombre de *Algoritmos Genéticos* (AGs).

En la naturaleza, los *procesos evolutivos* ocurren cuando se satisfacen las siguientes condiciones:

- Una entidad o individuo tiene la habilidad de reproducirse.
- Hay una población de tales individuos que son capaces de reproducirse.
- Existe alguna variedad, diferencia, entre los individuos que se reproducen.
- Algunas diferencias en la habilidad para sobrevivir en el entorno están asociadas a esa variedad.

Esta diversidad se manifiesta en cambios en los cromosomas de los individuos de la población, y se traslada a la variación de la estructura y el comportamiento de los individuos en su entorno. Estos cambios en la estructura y comportamiento se reflejan en el grado de supervivencia, adaptación, y en el nivel de reproducción. Los individuos que mejor se adaptan a su entorno son los que más sobreviven y más se reproducen. En un periodo de tiempo y con muchas generaciones de individuos, la población llega a contener más individuos cuyos cromosomas se trasladan a estructuras y a comportamientos adecuados a su entorno, sobreviviendo y reproduciéndose en un alto índice, de forma que

con el tiempo, la estructura de individuos en la población cambia debido a la *selección natural* (Darwin, 1859).

J. Holland fue uno de los pioneros en la aplicación de los principios en los que se basan los procesos de la evolución natural como reglas en el diseño de sistemas artificiales para la resolución de problemas. Holland (1975) proporciona un marco general en el que se muestra cómo aplicar los procesos evolutivos a los sistemas artificiales diseñados a partir de los sistemas naturales. Cualquier problema de adaptación puede formularse generalmente en términos genéticos. Una vez formulado en estos términos, tales problemas pueden resolverse mediante AGs.

Paralelamente a los estudios realizados por Holland, la idea de aplicar los principios en los que se basan los procesos de la evolución natural, como reglas en los procedimientos de búsqueda, se investigó con otros enfoques (Fogel y otros, 1966; Rechenberg, 1973). Actualmente, todas estas aportaciones iniciales han cristalizado en lo que se denomina *computación evolutiva* (Bäck y otros, 1997; Eiben y Smith, 2003), y recoge las distintas vertientes en el diseño de los *algoritmos de evolución* como herramientas para el análisis de sistemas no lineales complejos por medio de simulación computacional, y basados en los principios de los procesos de evolución natural.

Los AGs (Goldberg, 1989b; Holland, 1975) son procedimientos adaptativos para la búsqueda de soluciones en espacios complejos, inspirados en los procesos genéticos de los organismos naturales y en los principios de la evolución natural de poblaciones. Su idea básica es mantener una *población de cromosomas*, los cuales representan soluciones candidatas a un problema concreto, que evolucionan con el tiempo a través de un proceso de competición y variación controlada. Cada cromosoma tiene una bondad o adaptación asociada, que describe la adecuación de la solución a la que representa. El proceso de competición, denominado *mecanismo de selección*, utiliza estas adaptaciones para determinar los cromosomas que se usan para crear otros nuevos. Los nuevos cromosomas se generan a través de los operadores genéticos denominados *cruce* y *mutación*.

Los AGs se han aplicado con éxito en problemas de búsqueda y optimización. Gran parte de este éxito se debe a su capacidad para explotar la información acumulada sobre un espacio de búsqueda, y así dirigir las siguientes búsquedas hacia los mejores subespacios. Ésta es su principal ventaja, sobre todo en espacios grandes, complejos y parcialmente definidos donde las técnicas clásicas de búsqueda no son apropiadas. Los AGs ofrecen una aproximación válida a problemas que requieren técnicas de búsqueda efectivas y eficientes.

1.6.1. Estructura de un AG

Un AG comienza con una población de cromosomas generados aleatoriamente y va obteniendo mejores cromosomas gracias a la aplicación de los operadores genéticos, los cuales están basados en los procesos que ocurren en la naturaleza. La evolución se produce sobre la población en forma de selección natural. Durante sucesivas iteraciones, llamadas *generaciones*, se evalúa la adecuación (aptitud o adaptación) de los cromosomas como soluciones y en base a esta evaluación, se forma una nueva población de cromosomas usando un mecanismo de

selección y operadores genéticos específicos, tales como el cruce y la mutación. Para cada problema a resolver, se ha de proporcionar una *función de evaluación o adaptación*, f. Dado un cromosoma, la función de evaluación devuelve un valor numérico que se supone proporcional a la utilidad o adaptación de la solución que representa.

Existe un vocabulario muy extendido en el campo de los AGs, que toma prestado muchos términos de la genética para designar determinados conceptos. Se denominan *genotipos* a los propios cromosomas, *fenotipos* a las soluciones representadas por los cromosomas, *genes* a las unidades de un cromosoma, *loci* a las posiciones de un cromosoma y *alelos* a los posibles estados de un gen.

Aunque existen muchas posibles variantes de AGs, los mecanismo fundamentales bajo los cuales funcionan consisten en operar sobre una población de individuos, que inicialmente se genera de forma aleatoria, y cambiar en cada generación la población atendiendo a los tres pasos siguientes:

1. Evaluación de los individuos de la población.
2. Formación de una población intermedia a través del mecanismo de selección.
3. Formación de una nueva población a partir de los operadores genéticos de cruce y mutación.

Estos tres pasos se repiten hasta que el sistema deja de mejorar, o se alcanza un número máximo (T) de generaciones especificado por el usuario. La Figura 1.14 muestra la estructura de un AG.

```
AG(f){
   t ← 0;
   P(t) ← generarPoblacionInicialAelatoria();
   evaluar(P(t));

   Mientras (no se cumpla la condición de parada) {
      t ← t + 1;
      P'(t) ← seleccionarIndividuos(P(t − 1));
      O(t) ← aplicarCruceSobre(P'(t));
      O'(t) ← aplicarMutacionSobre(O(t));
      P(t) ← O'(t);
      evaluar(P(t));
   }

   devolver la mejor solución encontrada;
}
```

Figura 1.14: Seudocódigo básico de un AG

Debido a la naturaleza aleatoria de los AGs, normalmente todo el proceso se repite más de una vez. Los investigadores de los AGs suelen obtener estadísticas

tales como el mejor elemento encontrado durante una ejecución, la media de la función de evaluación de todos los cromosomas muestreados, generación en la que se encontró el mejor elemento, etc, todas ellas, promediadas sobre el número total de ejecuciones. Además, en la actualidad, se suelen aplicar tests estadísticos avanzados para dar respaldo a las conclusiones formuladas a partir de los resultados.

Representación

La *representación* es uno de los elementos clave en la operatividad de un AG, debido a dos razones: 1) los AGs manipulan una representación codificada del problema y 2) la representación puede limitar en gran medida la visión que tiene un sistema sobre su entorno (Koza, 1992).

Frecuentemente, las soluciones se han representado usando cadenas binarias. Para representar un espacio de búsqueda $S = D_1 \times \ldots \times D_n$ por medio del alfabeto binario se usa un conjunto de funciones $cod_i : D_i \rightarrow \{0,1\}^{L_i}$ con $L_i \in N$ e $i = 1, \ldots, n$, tal que cod_i codifica cada elemento de D_i usando una cadena de ceros y unos. Un elemento $X = (x_1, \ldots, x_n) \in S$ $(x_i \in D_i)$ queda representado a través de la concatenación de la codificación de cada una de sus componentes: $cod(X) = cod_1(x_1) \ldots cod_n(x_n)$. Los AGs basados en este tipo de representación se denominan *AGs con codificación binaria*.

En muchos casos, se han aplicado representaciones no binarias más naturales para los problemas particulares de la aplicación. Algunos ejemplos de estos casos son:

- *Vectores de números reales*: para optimizar los movimientos de un robot (Park y Choi, 2004), selección de características de enfermedades del corazón (Yan y otros, 2008), aprendizaje de movimientos bucales (Brito, 2007), problemas de quimiometría (Lucasius y Kateman, 1989), optimización numérica de funciones (Ballester y Carter, 2004; Davis, 1991b; Deb y otros, 2002a; Michalewicz, 1992; Wright, 1991) (Herrera y otros (1998) presenta una revisión bastante amplia), clasificación (Corcoran y Sen, 1994; Tohka y otros, 2007), diseño de filtros multicapa (Michielssen y otros, 1992), etc.
- *Vectores de números enteros*: para la optimización de funciones (Bramlette, 1991), diseño paramétrico de aviones (Bramlette y Bouchard, 1991), aprendizaje no supervisado de redes neuronales (Ichikawa y Ishii, 1993), planificación (Fowler y otros, 2008), diseño de esquemas de emisión de redes de comunicación (Chiang y otros, 2007), etc.
- *Listas ordenadas*: para problemas de planificación (Fox y McMahon, 1991; Lazzerini y Marcelloni, 2000; Pezzella y otros, 2008; Syswerda, 1991), problema del viajante de comercio (Hutter y Legg, 2006; Whitley y otros, 1989), problema de asignación de facilidades (Ahuja y otros, 2000; Larrañaga y otros, 1999; Tate y Smith, 1995), etc.
- *Expresiones en un lenguaje de programación*: para obtener programas para el control de tareas, planificación de un robot y regresión simbólica (Koza, 1991, 1992; O'Neill y Ryan, 2001).

- *Matrices de dos dimensiones de enteros*: para el problema del transporte (Michalewicz, 1991)

Selección

Consideremos una población P con los cromosomas $C_1, \ldots, C_N$. El mecanismo de selección produce una población intermedia, P', con copias de cromosomas en P. El número de copias recibidas por cada cromosoma depende de su valor de aptitud; aquellos cromosomas con un valor de aptitud alto tienen usualmente mayor probabilidad de contribuir con copias para P'. El mecanismo de selección realiza los siguientes dos pasos:

1. Para cada cromosoma C_i en P se calcula la probabilidad ($p_s(C_i)$) de incluir una copia suya en P'. El modelo más utilizado para calcular esta probabilidad es el modelo *proporcional* (Goldberg, 1989b; Holland, 1975), donde $p_s(C_i),\ i = 1, \ldots, N$ se calcula como

$$p_s(C_i) = \frac{f(C_i)}{\sum_{j=1}^{N} f(C_j)}. \tag{1.4}$$

   De esta forma, los cromosomas con un valor de aptitud por encima de la media reciben más copias que aquéllos con aptitud por debajo de la media.

2. Basándose en las probabilidades de selección calculadas en el paso anterior, se asigna, aleatoriamente, a cada elemento de P el número de copias suyas que aparecerán en P'. Este paso se realiza a través de un método de *muestreo*. El más simple, llamado *mestreo aleatorio simple* (Goldberg, 1989b; Holland, 1975), consiste en simular el comportamiento de una ruleta en la que existe para cada elemento de la población una región proporcional al valor de su probabilidad de selección. Cada vez que se juega, la ruleta determinará un elemento para la población intermedia. Jugando N veces, la población intermedia queda totalmente determinada. Los elementos con mayor probabilidad de selección tendrán, en la ruleta, una mayor proporción de espacio asignado, de esta forma la probabilidad de que la ruleta se detenga en las zonas asignadas a estos elementos es grande.

La Figura 1.15 muestra un ejemplo de la aplicación del mecanismo de selección, mostrándose a la izquierda la ruleta asociada.

Una estrategia adicional, que frecuentemente se utiliza como complemento del mecanismo de selección, es el *elitismo* (De Jong, 1975), el cual asegura que el mejor elemento de una generación t estará presente en la generación $t + 1$. Esto es conveniente ya que es posible que el mejor elemento de la generación t desaparezca debido a la aplicación del propio mecanismo de selección o por los operadores de cruce y mutación.

Operador de Cruce

El operador de cruce es un método para compartir información entre los cromosomas. Generalmente se combinan las características (genes) de dos cro-

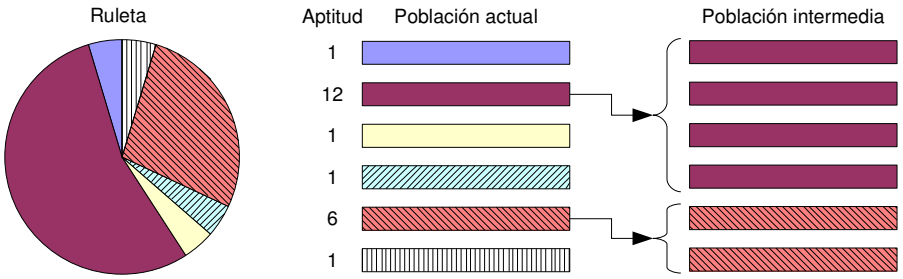


Figura 1.15: Mecanismo de selección por ruleta

mosomas padres para generar dos cromosomas hijos. El operador de cruce juega un papel central en los AGs. De hecho, puede considerarse como una de las características que definen a estos algoritmos y es uno de los componentes a tener en cuenta para mejorar su eficacia (Liepins y Vose, 1992). A veces, al operador de cruce se le denomina operador de *recombinación*.

Las definiciones para los operadores de cruce (y mutación) dependen de la representación particular escogida. Para el caso particular de los AGs con codificación binaria, el operador clásico de cruce es el *cruce simple* (Goldberg, 1989b; Holland, 1975), en el que dados dos cromosomas padres, $C_1 = (c_1^1 \dots c_L^1)$ y $C_2 = (c_1^2 \dots c_L^2)$, se generan los dos hijos, $H_1 = (c_1^1 \dots c_i^1, c_{i+1}^2 \dots c_L^2)$ y $H_2 = (c_1^2 \dots c_i^2, c_{i+1}^1 \dots c_L^1)$, donde i es un número aleatorio perteneciente a $\{1, \dots, L-1\}$. La Figura 1.16 muestra un ejemplo de la aplicación de este operador.

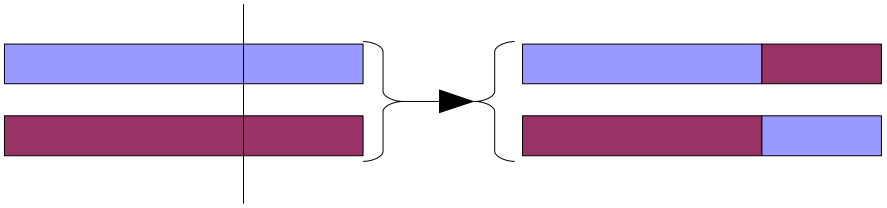


Figura 1.16: Cruce simple

Un parámetro de control del AG, denominado *probabilidad de cruce* (p_c), determina la probabilidad de aplicar el operador de cruce sobre parejas de cromosomas de la población intermedia.

Operador de Mutación

El operador de mutación altera arbitrariamente uno o más componentes (genes) de un cromosoma para aumentar la variabilidad estructural de la población. El papel de la mutación en los AGs es restaurar material genético perdido o no explorado en la población. De esta forma, se asegura que la probabilidad de alcanzar cualquier punto del espacio de búsqueda nunca sea cero. Cada gen en

cada cromosoma sufre una mutación de acuerdo con un parámetro de control, denominado probabilidad de mutación (p_m).

En los AGs con codificación binaria, la mutación de un gen es fácil: se cambia su valor, 1 por 0 y viceversa. La Figura 1.17 muestra un ejemplo de la aplicación del operador de mutación.

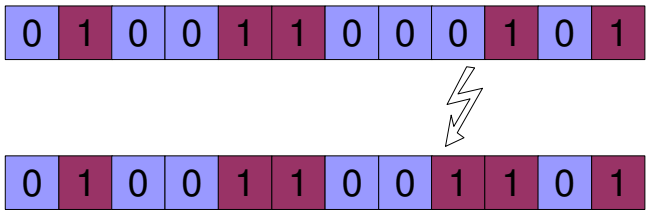


Figura 1.17: Mutación

Ejemplo

La Tabla 1.1 muestra un ejemplo del funcionamiento de un AG durante una generación, para la función de evaluación (valor de aptitud) $f(x) = x^2$. Se ha considerado que $p_c = 0{,}5$ y $p_m = 0{,}001$. El tamaño de la población es cuatro. Destacar que, dado el bajo valor de p_m, no se ha ejecutado ninguna mutación. La representación es binaria; cada número queda representado por su código en binario natural.

La generación mostrada se ha dividido en dos pasos: 1) aplicación del mecanismo de selección y 2) aplicación del cruce y mutación. En la cuarta columna se muestra el número de copias que el mecanismo de selección ha asignado a cada cromosoma. En la quinta, el símbolo '|' indica el punto de cruce elegido para la aplicación del operador de cruce simple.

Una simple generación es suficiente para comprender el poder de los AGs: las medidas peor, media y mejor han mejorado durante esta generación.

1.6.2. Teorema de los Esquemas

Una de las ventajas de los AG con codificación binaria es que, gracias a su simple formulación, se pueden realizar estudios teóricos fáciles. Holland (1975) desarrolló la *teoría de los esquemas*, la cual presenta resultados que explican el buen comportamiento de estos AGs.

Los AGs orientan su búsqueda de forma global dentro del espacio de búsqueda, y el concepto fundamental que describe esta orientación global es el de *esquema*. Un esquema es un patrón de similitud que describe un subconjunto de cadenas con similitudes en ciertas posiciones. Dado un alfabeto $\mathcal{A}$, un esquema es una cadena definida sobre el alfabeto $\mathcal{A} \cup \{\star\}$. El símbolo $\star$ indica que el alelo no está determinado. Por ejemplo, supuesto el alfabeto binario $\mathcal{A} = \{0, 1\}$, el esquema $\epsilon = \star 11 \star 010$ representa los cromosomas:

$$\{(0110010), (1110010), (0111010), (1111010)\}.$$

Tabla 1.1: Ejemplo de iteración de un AG

		Mecanismo de Selección		Cruce y Mutación		
Población C_i	**Func. de Ev.** $f(C_i)$	$\frac{f(C_i)}{\sum_{j=1}^{N} f(C_j)}$	**Copias Obtenidas**	**Población Intermedia**	**Población Nueva**	**Func. de Ev.** $f(C_i)$
0110	36	0,19	1	01|10	0101	25
0010	4	0,02	0	10|01	1010	100
1001	81	0,43	2	1001	1001	81
1000	64	0,34	1	1000	1000	64
Peor	4					25
Media	46,25					67,5
Mejor	81					100

Dos características importantes de un esquema ϵ son:

- *Orden de* ϵ, notado como $o(\epsilon)$: número de símbolos fijos en ϵ.
- *Longitud de* ϵ, notado como $\delta(\epsilon)$: distancia entre el primer y último símbolo fijo de ϵ.

Supongamos que L es la longitud de los cromosomas, $f(\epsilon)$ es la adaptación de un esquema ϵ (valor medio de la función de evaluación de los cromosomas presentes en la población que siguen el esquema ϵ) $\overline{f}$ es la media de la función de evaluación de los elementos en la población y $m(\epsilon, t)$ es el número de cromosomas que siguen ϵ en la población durante una generación t. El número de representantes esperado del esquema ϵ en la siguiente generación, $t+1$, después de la aplicación del mecanismo de selección y los operadores de cruce y mutación, atiende a la siguiente expresión (Holland, 1975):

$$E[m(\epsilon, t+1)] \geq m(\epsilon, t) \cdot \frac{f(\epsilon)}{\overline{f}} \cdot \left[1 - p_c \cdot \frac{\delta(\epsilon)}{L-1}\right] \cdot (1-p_m)^{o(\epsilon)}, \tag{1.5}$$

donde

- $m(\epsilon, t) \cdot \frac{f(\epsilon)}{\overline{f}}$ representa el número esperado de ejemplares obtenidos tras la aplicación del mecanismo de selección,
- $1 - p_c \cdot \frac{\delta(\epsilon)}{L-1}$ representa aproximadamente la probabilidad de supervivencia de ϵ después de la aplicación del operador de cruce. Esta probabilidad es alta cuando $\delta(\epsilon)$ es baja, y
- $(1-p_m)^{o(\epsilon)}$ es la probabilidad de supervivencia de ϵ después de la aplicación del operador de mutación, Esta probabilidad es alta cuando $o(\epsilon)$ es bajo.

Este resultado motiva el teorema de los esquemas (Holland, 1975):

Teorema 1 *Esquemas cortos, de bajo orden, y con adaptación por encima de la media, llamados bloques constructores, reciben un incremento exponencial de cromosomas en las siguientes generaciones.*

Un resultado inmediato de este teorema es la llamada *hipótesis de los bloques constructores* (Goldberg, 1989b):

Resultado 1 *La búsqueda realizada por un AG con codificación binaria es casi óptima a causa de la yuxtaposición de los bloques constructores.*

Esto significa que gracias al cruce de los bloques constructores existentes en la población se podrán localizar cromosomas cercanos al óptimo. Éste es un resultado importante de los AGs con codificación binaria, ya que predice un buen comportamiento para este tipo de AGs. El siguiente ejemplo ilustra el

significado del teorema de los esquemas para el AG del ejemplo de la Sección 1.6.1. La Tabla 1.2 muestra el efecto de la generación t sobre cuatro esquemas interesantes. Estos esquemas son diferentes con respecto a su orden, longitud y adaptación. La segunda columna muestra el número de cromosomas del esquema antes del proceso genético y la tercera muestra la adaptación de los esquemas en el mismo momento. Las siguientes columnas representan la misma información pero después de la aplicación del mecanismo de selección y del operador de cruce. El signo '+' indica que la generación ha causado un aumento en el número de cromosomas, mientras que el signo '−' indica una disminución de cromosomas.

Tabla 1.2: Procesamiento de esquemas

ϵ	$m(\epsilon, t)$	$f(\epsilon)$	$m(\epsilon, t+1)$	$f(\epsilon)$
$1\star\star\star$	2	72,5	3+	81,66
$01\star\star$	1	36	1	25
$1\star\star1$	1	81	1	81
$0\star\star0$	2	20	0−	

El esquema $1\star\star\star$ es corto ($\delta(1\star\star\star) = 0$), de orden bajo ($o(1\star\star\star) = 1$) y con adaptación sobre la media ($f(1\star\star\star) = 72{,}5$ y $\overline{f} = 46{,}25$), por ello ha recibido el incremento exponencial en el número de sus cromosomas predicho por el teorema de los esquemas. El esquema $01\star\star$ es un esquema, con adaptación bajo la media, que ha mantenido el mismo número de cromosomas debido a que es corto y de orden bajo; el efecto del cruce no es significativo sobre los esquemas con estas características. El esquema $1\star\star1$ puede perder cromosomas fácilmente gracias al operador de cruce, sin embargo, ha mantenido el número de cromosomas debido a que es un esquema con adaptación por encima de la media. El caso de $0\star\star0$ es diferente; este esquema ha perdido todos sus cromosomas.

1.6.3. El Problema de la Convergencia Prematura

El comportamiento de los AGs depende estrechamente del equilibrio entre explotar lo que actualmente es mejor y explorar posibilidades que pueden eventualmente convertirse en algo mucho mejor. Los dos siguientes conceptos suelen usarse en el campo de los AGs:

- *Presión selectiva*: Para obtener una búsqueda efectiva debe haber un criterio de búsqueda (la función de evaluación o de aptitud) y una presión selectiva que dé a los individuos con mejor aptitud una mayor oportunidad de seleccionarse para la reproducción, mutación y supervivencia. Sin presión selectiva, el proceso de búsqueda se convierte en aleatorio y las regiones prometedoras del espacio de búsqueda no se favorecen frente a las regiones no prometedoras.
- *Diversidad en la población* (en inglés, *population diversity*): La pérdida de alelos debida a la *presión selectiva*, el ruido de la selección, la ruptura de esquemas por el operador de cruce, y un conjunto de parámetros de control

(p_c, p_m, N, etc.) no adecuado, pueden hacer que este equilibrio entre la exploración y la explotación se desproporcione produciendo la pérdida de diversidad en la población (Li y otros, 1992; Mahfoud, 1995; Potts y otros, 1994).

Presión selectiva y diversidad en la población, conceptos estrechamente ligados con intensificación y diversificación, se relacionan inversamente. Aumentar la presión selectiva suele resultar en una pérdida temprana de la diversidad, mientras que mantener la diversidad en la población reduce el efecto de incrementar la presión selectiva. Estos dos factores deben controlarse para obtener sus ventajas simultáneamente y permitir que las regiones del espacio de búsqueda más prometedoras se alcancen y refinen.

Cuando se pierde la diversidad en la población, en las etapas iniciales del AG, aparece un grave problema: la *convergencia prematura* hacia zonas del espacio de búsqueda que no contienen el óptimo global. Una parte muy importante en la investigación de los AGs se ha concentrado en la propuesta de distintas vías para evitar la convergencia prematura. Entre los mecanismos más destacables están: el aporte de diversidad mediante la representación, modificaciones del mecanismo de selección, modificaciones del operador de cruce, búsqueda de valores óptimos para los parámetros de control, adaptación de determinados elementos del AG a lo largo de la ejecución, ejecuciones múltiples secuenciales y separación espacial. A continuación, pasamos a describir distintas aproximaciones de cada una de estas soluciones.

Representación Redundante

La mayoría de los modelos de representación, usados por los AGs, pretenden realizar una traducción biyectiva entre genotipo y fenotipo, utilizando el mínimo número de variables de decisión que afectan realmente al problema que se está tratando. En la literatura, existen algunas propuestas que atacan el problema de la convergencia prematura introduciendo un mayor número de genes en los individuos (no utilizan una traducción biyectiva), aportando así diversidad en el propio modelo de representación. Para ello, una solución X con L variables de decisión se extiende, añadiendo más variables que en ocasiones adquieren un papel importante en la evaluación de la solución (Raich y Ghaboussi, 1997). En estos modelos de representación, aparece una clara diferencia entre el genotipo de una solución y su fenotipo. Tanto es así que, soluciones con diferente genotipo pueden asociarse al mismo fenotipo. El AG necesita un procedimiento de traducción que seleccione e interprete, los genes que tienen un papel más relevante que el resto y poder evaluar correctamente la solución. Además, estas técnicas reducen la probabilidad de que el operador de cruce o de mutación rompan los esquemas más significativos de una solución, ya que es probable que produzcan cambios en el material genético redundante.

Muhando y otros (2006) proponen un modelo de representación en el cual cada individuo de la población se compone de un gen dominante y uno recesivo. Cuando dos individuos se cruzan, el abanico de soluciones descendientes es más amplio dependiendo de los pares de genes que se seleccionen de los padres. Según los autores, el resultado es un proceso de evolución en el que la apari-

ción de convergencia prematura se retrasa respecto a un AG con un modelo de representación sin genes recesivos.

Xiao y otros (2008) utilizan un modelo de representación muy cercano al presente en las cadenas de *ácido desoxirribonucleico* (ADN). Una solución se representa como una secuencia de cuatro bases: *adenina* (A), *guanina* (G), *citosina* (C) y *timina* (T). El AG utiliza una tabla de traducción que, a partir de cada subsecuencia (o grupo de subsecuencias) de tres elementos (*codón*), obtiene el valor de cada variable de decisión del problema. En esta tabla de traducción, más de un codón puede producir el mismo valor para una variable de decisión, lo que pone de manifiesto la redundancia del modelo. Los autores indican que la redundancia en la codificación propuesta incrementa la diversidad en la población del AG.

Modificaciones del Mecanismo de Selección

Una de las líneas de investigación que ha recibido mayor atención, para solventar el problema de la convergencia prematura, ha sido el diseño de nuevos mecanismos de selección. Hemos de entender que éste repercute intensamente en dos aspectos del proceso evolutivo: la selección de los individuos que reciben copias en la población intermedia y el emparejamiento de individuos que pueden producir descendientes. Los métodos diseñados teniendo en cuenta el primer aspecto pretenden evitar que individuos con aptitud mejor a la media obtengan un número de copias excesivamente superior al de los individuos con aptitud inferior. Los métodos que vigilan el emparejamiento de individuos intentan combinar soluciones que puedan producir descendientes que aporten diversidad a la población. Para ello, seleccionan soluciones que mantienen un cierto nivel de disimilitud.

Métodos que Limitan el Número de Copias. Baker (1987a) propone el siguiente modelo para la obtención de probabilidades de selección, denominado *orden lineal*:

$$p_s(C_i) = \frac{1}{N}\left(\eta_{max} - (\eta_{max} - \eta_{min})\frac{pos(C_i) - 1}{N - 1}\right), \tag{1.6}$$

donde $pos(C_i)$ nota el lugar que ocupa el cromosoma C_i cuando la población se ordena ascendentemente según la función de evaluación. El parámetro η_{min} determina el número de copias esperado para el cromosoma con peor función de evaluación. El mejor cromosoma tendrá un número de copias esperado de η_{max}, el cual se aproxima a $2 - \eta_{min}$. Con el orden lineal, cada individuo recibe un número de copias esperado que depende de la posición de tal cromosoma con respecto a su adaptación, y es independiente de la magnitud de tal adaptación. De esta forma, se evita que los superindividuos dominen la población en pocas generaciones, caso que sí puede ocurrir usando el modelo proporcional. Actualmente, existen varios métodos de selección que utilizan la posición que ocupa cada cromosoma en una ordenación de la población, en vez de el valor de aptitud de éstos. Sokolov y otros (2007) realizan un estudio de diferentes

técnicas de selección de este tipo y evalúan su impacto sobre la diversidad en la población.

Otro modelo es el caso del *muestreo universal estocástico* (Baker, 1987b). Este método garantiza que el número de copias asignado a un elemento esté acotado por la parte entera y la parte entera más uno del número de copias esperado en razón a su probabilidad de selección. Consiste en la simulación de una ruleta con las mismas características que la usada en el muestreo aleatorio simple, con la diferencia de que se incluyen N indicadores igualmente espaciados en la ruleta. Con un solo juego de la ruleta se determina la composición de la población intermedia. El número de indicadores situados dentro de la zona perteneciente a cada cromosoma establece su número de copias asignadas. Este modelo ayuda a prevenir la convergencia prematura ya que evita el dominio de los elementos con alta función de evaluación en las primeras generaciones del AG. La Figura 1.18 muestra un ejemplo de selección mediante el muestreo universal estocástico asociado a los cromosomas de la Figura 1.15.

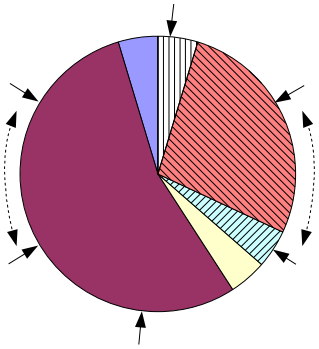


Figura 1.18: Ruleta usada para el muestreo universal estocástico

Métodos que Velan por la Diversidad de los Padres. Existe un grupo importante de mecanismos de selección de padres que se definen usando dos medidas del cromosoma a introducir en la población: su función de evaluación y una medida que evalúa la diversidad introducida en la población (Bandyopadhyay y otros, 1998; Bonham y Parmee, 1999; Craighurst y Martin, 1995; De Jong y otros, 2001; Eshelman y Schaffer, 1991; Fernandes y Rosa, 2001; Ichikawa y Ishii, 1993; Jassadapakorn y Chongstitvatana, 2004; Lee, 2003; Mori y otros, 1995; Shimodaira, 1999; Toffolo y Benini, 2003). Sus propuestas favorecen a los individuos con buenos valores de aptitud y altas contribuciones en la diversidad.

Shimodaira (1999) propone un método en el que los individuos de la población intermedia se ordenan con respecto a su valor de aptitud (de mejor a peor) y en ese orden se seleccionan aleatoriomente según una probabilidad que depende de su distancia con el mejor individuo de la población, a mayor distancia, mayor probabilidad.

En el *emparejamiento variado negativo* (Fernandes y Rosa, 2001), primero, se selecciona un padre por el método de la ruleta y después, otros n_{ass} cromosomas,

por el mismo método. Después, se mide la similitud entre cada uno de los n_{ass} cromosomas y el primer padre. El cromosoma con menor similitud se elige como segundo padre.

Bandyopadhyay y otros (1998), Craighurst y Martin (1995), Eshelman y Schaffer (1991) y Tsai y otros (2004a) proponen métodos en los que dos soluciones padres sólo producen descendientes si cumplen ciertas condiciones como superar un umbral mínimo de diferencia o no tener relaciones de parentesco.

Selección Uniforme en Aptitud. Hutter (2002) y Hutter y Legg (2006) proponen un método que introduce diversidad entre los padres de una forma especial. Sean f_{min} y f_{max} el menor y mayor valor de aptitud, respectivamente, de los individuos de la población actual. El método propuesto genera un valor aleatorio en el intervalo $[f_{min}, f_{max}]$. Después, selecciona el individuo de la población cuya aptitud es más cercana al valor generado. Con este método, se favorece la selección de individuos cuya aptitud no es muy común en la población.

Modificaciones del Operador de Cruce

Se acepta que el operador de cruce juega un papel fundamental en los AGs. De hecho, puede considerarse como una de las características que definen y diferencian a los AGs de otros algoritmos basados en la evolución natural, siendo uno de los componentes que más se ha tenido en consideración a la hora de mejorar el comportamiento de los AGs (Liepins y Vose, 1992). Además, algunos autores lo consideran como una pieza clave para resolver el problema de la convergencia prematura (Booker, 1987). Numerosas investigaciones se han desarrollado tanto para encontrar el valor óptimo del parámetro asociado (p_c) como para diseñar operadores de cruce alternativos y más potentes, entre los que destacan:

- *Cruce n-punto* (De Jong y Spears, 1992; Eshelman y otros, 1989). Es una generalización del cruce simple; se seleccionan n puntos de cruce aleatoriamente, y se intercambian los segmentos, situados entre éstos, de los cromosomas a cruzar.
- *Cruce uniforme* (Syswerda, 1989). El valor de cada uno de los genes de los hijos viene determinado por la elección aleatoria de los valores de estos genes en sus padres. Para ello se utiliza una probabilidad p_f que indica la frecuencia con la que el valor de un gen del primer hijo se toma del primer padre. Cuando un hijo recibe el gen de uno de sus padres, el otro hijo recibirá el gen del otro padre.

Existen dos conceptos muy importantes relacionados con el operador de cruce: la *ruptura de esquemas* y la *tendencia de la representación*:

- La *ruptura de esquemas* se produce cuando el operador de cruce no propaga los esquemas de orden bajo para formar esquemas de orden alto, evitando que los esquemas importantes puedan guiar la búsqueda.

- El efecto denominado *tendencia de la representación* se produce cuando la ruptura producida en un esquema ϵ por un operador de cruce dependen de la distancia entre las posiciones definidas de ϵ, siendo mayor cuanto mayor sea tal distancia. Un operador de cruce con una ruptura de esquemas baja permite que los valores de genes ocupando posiciones contiguas se propaguen juntos durante mucho tiempo en los cromosomas. En estos casos la representación es determinante en el propio proceso de búsqueda.

  Intuitivamente se puede observar que los operadores de cruce simple e incluso el cruce 2-puntos presentarán el efecto tendencia de la representación. Por el contrario, el cruce uniforme y el n-punto, con un valor de n alto, no muestran este fenómeno, ya que con su utilización se favorece la aparición de una gran diversidad de esquemas a lo largo de la búsqueda, debido a que son operadores con alta ruptura de esquemas. Esta propiedad puede resultar perjudicial, ya que produce la pérdida de zonas útiles presentes en la población.

Para que un AG sea efectivo, debe existir un consenso entre la preservación de esquemas y la aparición de nuevos esquemas a través de la ruptura de otros. Eshelman (1991) propone una solución para alcanzar este consenso: combinar un operador de cruce con alta ruptura de esquemas y un mecanismo de selección conservador que mantenga siempre los N (N es el tamaño de la población) mejores individuos obtenidos hasta el momento, con lo que se logra obtener esquemas en nuevos contextos mientras se preservan los mejores descubiertos.

Análisis de Valores Óptimos para los Parámetros

Existen varios trabajos en la línea de estudiar los valores de los parámetros de control (p_c, p_m, N, etc.) que optimizan el comportamiento de los AGs. Los primeros estudios se deben a De Jong (1975). De Jong trabajó sobre cinco funciones de prueba con diferentes propiedades, y definió unas medidas que han sido básicas para ulteriores estudios. Estas medidas son las siguientes:

1. *Mejor valor de la función de evaluación después de T evaluaciones*:

$$f^\star(T) = \ Mejor\ de\ \{f(1), \ldots, f(T)\}. \tag{1.7}$$

2. *Medida Offline después de T evaluaciones*:

$$Offline(T) = \frac{\sum_{t=1}^{T} f^\star(t)}{T}. \tag{1.8}$$

3. *Medida Online después de T evaluaciones*:

$$Online(T) = \frac{\sum_{t=1}^{T} f(t)}{T}. \tag{1.9}$$

donde $f(t)$ es el valor de la función de evaluación correspondiente al cromosoma generado en el tiempo t y $f^\star(t)$ es el mejor valor de la función de evaluación encontrado después de t evaluaciones.

De Jong estudió el efecto de un conjunto de parámetros de control sobre las medidas anteriores, entre ellos p_c, p_m y N. Sus experimentos indicaron que los mejores valores de N estaban entre 50 y 100, para p_c era aproximadamente 0,6 y para p_m era 0,001.

Grefenstette (1986) presenta un AG, denominado *metaAG*, que trata de determinar los valores de los parámetros de control con los que un AG obtendría el mejor comportamiento en las medidas Offline y Online. Cada cromosoma de metaAG codifica valores de los parámetros de control de un AG. La función de evaluación de un cromosoma se describe en función de las medidas Offline y Online obtenidas por un AG, que use los valores de los parámetros de control representados en tal cromosoma, sobre las cinco funciones de prueba de De Jong. Los resultados mostraron que el valor más adecuado para N es 30, para p_c es 0,95 y para p_m es 0,01.

Schaffer y otros (1989) realizan otro estudio sobre la relación entre los parámetros N, p_m y p_c, y la medida Online de un AG con codificación Gray (véase la Sección 1.6.5). Los rangos de valores que permitieron obtener las mejores medidas Online son de 20 a 30 para N, de 0,75 a 0,95 para p_c y de 0,005 a 0,01 para p_m.

AGs Adaptativos

Encontrar operadores genéticos robustos o parámetros de control que permitan evitar la convergencia prematura en cualquier problema no es una tarea fácil, ya que sus repercusiones sobre la operación del AG son complejas y los operadores y parámetros óptimos dependen del problema que se está tratando (Bäck, 1992a). Además, se pueden necesitar distintos operadores o distintos valores para los parámetros de control a lo largo de la ejecución para producir un equilibrio óptimo entre diversificación e intensificación. Por estas razones, uno de los caminos más seguidos en la literatura para evitar la convergencia prematura consiste en la adaptación de determinados elementos de los AGs a lo largo de la ejecución en función de su estado o información disponible sobre el espacio de búsqueda.

Eiben y Schut (2008) realizan una revisión y clasificación taxonómica de los AGs adaptativos propuestos en la literatura, diferenciando:

- *¿Qué se adapta?*: En este punto se encuentran propuestas que adaptan cada una de las diferentes partes del AG como son la representación, función de evaluación, operadores genéticos, mecanismo de selección, población o los parámetros de control de éstos.

- *¿Cómo se adapta?*: Existen básicamente tres formas de realizar la adaptación:

  - El método *determinista* produce el cambio cuando se cumple una condición determinada, independiente del estado del proceso de búsqueda. Normalmente, se sigue un esquema basado en tiempo, número de evaluaciones o generaciones.

  - El método *adaptativo* usa medidas estadísticas que indican el estado del AG para detectar, con posible antelación, la aparición de la convergencia prematura, y actuar apropiadamente.
  - El método *autoadaptativo* codifica los parámetros de adaptación en cada uno de los cromosomas, siendo objeto de la evolución del AG. La idea es realizar una evolución del mecanismo de evolución.

- En el método *adaptativo*, además podemos preguntarnos por *¿qué información se utiliza para aplicar el cambio?*: Podemos encontrar propuestas que calculan medidas estadísticas precisas (*información absoluta*), que aplican cambios según modelos teóricos (Yu y otros, 2007), y otras en las que se mide el rendimiento de diferentes componentes del algoritmo (*información relativa*) y se modifican sus frecuencias de actuación según algún criterio. Mattiussi y otros (2004) y Nsakanda y otros (2007) realizan revisiones y propuestas nuevas de posibles medidas estadísticas a utilizar.

Adaptación en la Función de Evaluación. Algunas propuestas modifican la función de evaluación de los individuos con la intención de disminuir la ventaja que poseen los individuos con mejor valor de aptitud frente a los de peor valor (de la Maza y Tidor, 1992; Goldberg, 1989b; Kuo y Hwang, 1996; Michalewicz, 1992). Por ejemplo, la *ley de la potencia* (Goldberg, 1989b) escala la función de evaluación de la siguiente forma:

$$f'(C) = f(C)^k, \tag{1.10}$$

donde $k \approx 1$

Otros mecanismos modifican la función de evaluación cuando se considera que se ha encontrado un óptimo local, de forma que la región del espacio de búsqueda encontrada no sea atractiva para otros individuos (Beasley y otros, 1993a; Sakanashi y Suzuki, 1994). Por ejemplo, el método propuesto por Sakanashi y Suzuki (1994) asigna peores valores a los elementos cercanos a la zona del óptimo localizado. Este procedimiento se repite cada vez que la población converge a un óptimo local. Para que no se visite el mismo óptimo dos veces, la función de evaluación retiene su efecto de huida de todos los óptimos encontrados durante todo el proceso.

Por último, en este apartado también debemos incluir los métodos de *compartición de recursos* (en inglés *sharing*) (Goldberg y Richardson, 1987; Goldberg, 1989b), o métodos de *limpiado* (en inglés *clearing*) (Pétrowski, 1996) en los que los individuos de la población deben compartir, o competir por, respectivamente, los recursos de la región del espacio de búsqueda asociada. Este tipo de métodos se utilizan normalmente, cuando se pretenden encontrar más de una solución óptima para el problema tratado.

Adaptación de los Parámetros de Control. Desde hace mucho tiempo, se viene reconociendo el significativo impacto de los parámetros de control de los AGs (p_m, p_c, N, etc) sobre su eficacia (Grefenstette, 1986). Si se utilizan valores no adecuados para estos parámetros, puede ocurrir que no se alcance un equilibrio entre diversificación e intensificación apropiado, y consecuentemente,

la eficacia del AG se ve afectada considerablemente (Davis, 1989). Lobo y otros (2007) realizan una revisión de diferentes técnicas para adaptar estos parámetros a lo largo de la ejecución del AG. A continuación se resumen las principales propuestas:

- *Tamaño de la Población*: Baker (1987a) propone un método para aumentar el tamaño de la población cuando existen pocos cromosomas que aportan una copia a la población intermedia. Arabas y otros (1994) adapta el tamaño de la población estableciendo la *edad* que cada cromosoma puede permanecer en la población. Smith (1993) y Yu y otros (2007) utilizan medidas estadísticas para calcular el tamaño de población ideal, y lo adaptan. Lobo y Goldberg (2004) proponen un AG donde varias poblaciones de diferentes tamaños compiten por resolver el problema, resultando en una adaptación del tamaño de la población. Por último, Koumousis y Katsaras (2006) presenta un AG en el que se alternan procesos de intensificación progresiva, mediante una reducción iterativa del tamaño de la población, y procesos de diversificación alta, mediante el aumento instantáneo del tamaño y reinicialización de los individuos, siguiendo un esquema de *dientes de sierra*.

- *Parámetros del Operador de Mutación*: Bäck (1992a), Bramlette (1991), Fogarty (1989) y Holland (1975) proponen planes de reducción de p_m a lo largo de la ejecución del AG. Mauldin (1984), Venugopal y Narendran (1992), Whitley y Hanson (1989) y Whitley (1990) aumentan p_m cuando los nuevos cromosomas reducen la diversidad de la población. Bäck (1992a) y Bäck (1992b) codifican la probabilidad de cada variable de decisión del problema en la representación de la solución, sometiéndola al proceso evolutivo. Herrera y Lozano (2000b) combinan estratégicamente dos técnicas contrarias para adaptar la intensidad con que se mutan las soluciones. Por último, Hrstka y Kučerová (2004) proponen un método donde, cada cierto tiempo, la región ocupada por la mejor solución encontrada se cataloga como área radioactiva, y como tal, las soluciones que entran en ella sufren una mutación con una probabilidad alta.

- *Probabilidad de Cruce*: Wilson (1986) aumenta p_c cuando la entropía de la población disminuye, y viceversa. Booker (1987) propone un método similar pero utilizando la *medida de dificultad* propuesta por Baker (1987a).

- *Probabilidades de Cruce y Mutación*: Davis (1985) propone reducir p_c y aumentar p_m a lo largo de la ejecución del AG. Srinivas y Patnaik (1994) proponen un método por el cual p_c y p_m se calculan de forma separada para cada solución de la población, de forma que las probabilidades son bajas para las soluciones de calidad alta, y elevadas para las soluciones de peor calidad.

Adaptación de los Operadores Genéticos. Existen algunos aspectos en la aplicación de los operadores genéticos que se dejan al azar. Éste es el caso, por ejemplo, de la posición del punto de cruce de un operador de cruce simple. Se han propuesto mecanismos adaptativos para determinar estos aspectos para que

la actividad de los operadores genéticos asociados sea más eficaz. A continuación describimos algunos de ellos.

Schaffer y Morishima (1987) y Hoffmeister y Bäck (1990) describen un proceso por el que el AG realiza un proceso de búsqueda de las posiciones donde se puede realizar un cruce simple. Para ello, a cada individuo se le añade una cadena binaria de igual longitud, donde un '1' indica un posible punto de cruce, y un '0' un punto prohibido de cruce. Esta cadena binaria también sufre la operación del cruce y mutación.

Booker (1987) se da cuenta de que cuando dos cromosomas binarios diferentes se cruzan en un punto tal que se intercambian segmentos idénticos, los cromosomas resultantes serán iguales a los padres. Entonces, propone restringir el cruce a los puntos que pueden producir descendientes con material genético nuevo.

Sebag y Schownauer (1994) proponen un mecanismo adaptativo de cruce basado en un proceso de aprendizaje inductivo. Cada T generaciones, los descendientes producidos se etiquetan como 'malos', 'buenos' o 'inactivos'. Entonces, se generan un conjunto de reglas con intención de caracterizar los puntos de cruce que producen descendientes 'buenos', y se utilizan en las próximas T generaciones.

White (1994) introduce un mecanismo que identifica grupos de genes que deben mantenerse juntos cuando se aplica el cruce. El principio usado es: 1) aquellos genes que están juntos durante el cruce y generan hijos superiores a los padres deben permanecer juntos en futuros cruces, 2) aquellos genes que están juntos durante el cruce y generan hijos peores a los padres, no deben permanecer juntos en el futuro, y 3) aquellos genes que están juntos durante el cruce y generan hijos que no presentan cambios destacables sobre los padres, pueden o no estar juntos en el futuro.

Michalewicz (1992) presenta un operador de mutación adaptativo para AGs basados en codificación real que ajusta, a lo largo del tiempo, la magnitud de los cambios producidos sobre los alelos. En las generaciones iniciales, los cambios son grandes, favoreciendo así la exploración. Conforme avanzan las generaciones, la magnitud de los cambios es menor, proporcionando en las últimas generaciones un ajuste local eficaz (véase la Sección 1.6.5 para más información).

Ichikawa y Ishii (1993) presentan un operador de mutación adaptativo para genes pertenecientes al conjunto de enteros $\{-10, \ldots, 10\}$. La probabilidad de mutación siempre es uno. El posible cambio efectuado sobre un alelo de posición i en un cromosoma depende de la diversidad existente en el gen de posición i en la población. Si esta diversidad es baja, el cambio será grande, mientras que si la diversidad es grande, no se produce cambio o éste será pequeño.

Lin y Yang (2002) proponen un operador de 'cruce/mutación' que, dados dos padres binarios, genera descendientes con alelos que no están presentes en los padres. Para ello, aplica el operador lógico *o exclusivo* y su negación. Cuando un cierto alelo no se presenta frecuentemente en la población, estos operadores lo generan.

Selección Adaptativa de Operadores Genéticos. En esta sección, se discuten algunos mecanismos adaptativos que partiendo de un conjunto de operadores genéticos, deciden durante la ejecución del AG qué operadores genéticos son los más convenientes para cada circunstancia.

Spears (1992) utiliza un AG con dos operadores genéticos: 2-puntos y el cruce uniforme. A cada solución se le añade un gen binario para que el AG decida el tipo de cruce que debe aplicar. Un '1' se puede referir al cruce uniforme y un '0' al 2-puntos. Existen dos técnicas para usar estos genes extras, cuando se ejecuta el cruce sobre la población: la *local* y la *global*. En la primera, durante cada evento de cruce, se observan los genes adicionales de los padres; si ambos son '1' se ejecuta el cruce uniforme, si ambos son '0' se ejecuta el cruce 2-puntos y si uno es '1' y el otro, '0' se elige aleatoriamente uno de ellos. Con la técnica global, la elección del cruce no se hace de forma individual sino considerando la población total. Por ejemplo, supongamos que el 75 % de la población tiene un valor de '1' en su gen adicional, entonces el cruce uniforme se aplicará el 75 % de las veces mientras que el restante 25 % es para el 2-puntos.

Davis (1989) implementa un AG basado en codificación real que usa dos tipos de operadores de cruce y tres tipos de operadores de mutación. La utilización de todos permite disponer de un amplio rango con respecto a los niveles de exploración y explotación que producen. Cada operador tiene una probabilidad de aplicación distinta. Durante cada evento de la aplicación de los operadores genéticos, se selecciona un solo operador de forma probabilista, de acuerdo al conjunto de probabilidades de los operadores. Esto difiere de la técnica usada comúnmente, en la cual, un operador de cruce y un operador de mutación pueden aplicarse a la vez sobre un mismo cromosoma. Se incorpora un proceso adaptativo que altera las probabilidades de aplicación de los operadores según la función de evaluación de los cromosomas generados por cada uno de ellos durante cierto tiempo. Los operadores que permiten la generación de buenos cromosomas tendrán asociadas altas probabilidades de aplicación, y de esta forma se usarán frecuentemente. Por otro lado, los operadores que producen hijos con función de evaluación peor que la de sus padres se aplicarán con menor frecuencia. Julstrom (1995) y Zhang y otros (2006) presentan modelos muy similares.

Schlierkamp-Voosen y Mühlenbein (1994) presentan un mecanismo adaptativo basado en el uso de subpoblaciones. Supongamos que disponemos de un conjunto de operadores de cruce o mutación. Para cada uno de estos operadores, se forma una subpoblación sobre la cual se aplicará tal operador. El número de cromosomas total entre todas las subpoblaciones permanece fijo, aunque el tamaño de las subpoblaciones varía con el tiempo. Cada subpoblación compite con las otras subpoblaciones, de tal forma que, gana o pierde individuos en función de la calidad de la evolución que se produce sobre ella (que depende claramente de su configuración). El tamaño de la subpoblación con mejor calidad aumentará mientras que las del resto disminuye. Schlierkamp-Voosen y Mühlenbein (1994) definen dos ejemplos de este modelo: uno basado en distintos operadores de cruce y otro basado en el mismo operador de mutación pero cambiando su efecto mediante un parámetro de control.

Adaptación en la Representación. Un mecanismo para obtener precisión usando un AG con codificación binaria consiste en realizar ejecuciones iterativas del algoritmo de forma que en una ejecución ζ se utilicen los individuos *recodificados* obtenidos al finalizar la ejecución $\zeta - 1$, así, se parte de una zona identificada como óptima con anterioridad y se obvia el resto del espacio. De esta forma se consigue acotar en cada ejecución el intervalo de actuación de los parámetros por lo que la precisión alcanzada cada vez está más refinada.

El *Codificador Dinámico de Parámetros* (Schraudolph y Belew, 1992) se basa en esta idea. Cada parámetro se trata de forma independiente y se codifica usando una cadena binaria de longitud fija. Durante una generación t cada parámetro i tiene asociado un intervalo de actuación $[a_i^t, b_i^t]$. Este intervalo se divide en cuatro zonas de igual longitud llamadas *cuartos*. Sobre los cuartos se realiza un histograma con los valores del parámetro i disponibles en la población. A continuación se consideran los intervalos intersección de cuartos adyacentes y se calculan los valores del histograma asociados (véase la Figura 1.19).

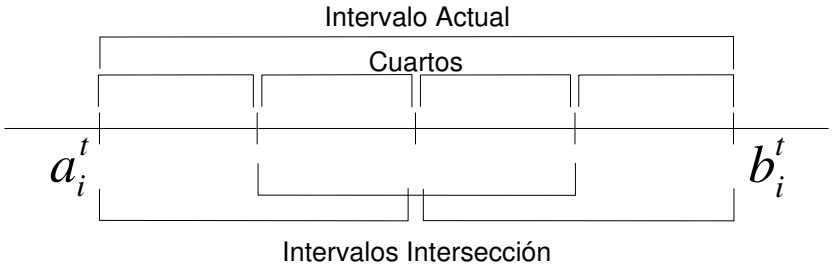


Figura 1.19: Intervalos considerados por el Codificador Dinámico de Parámetros

Con estos intervalos se crean estadísticas. Después de un número de generaciones determinadas por un parámetro de entrada, se observa el intervalo intersección que se ha muestreado más frecuentemente. Si su frecuencia sobrepasa un valor umbral se considera que la población ha convergido en tal intervalo y se dispara un *operador de ampliación*. Este operador restringe los valores del parámetro i en el intervalo intersección anterior. Las codificaciones actuales de este parámetro en la población se cambian para adaptarse a los nuevos límites, de forma que los valores que pertenecen al nuevo intervalo se recodifican y los que no pertenecen se duplican sobre tal intervalo.

El Codificador Dinámico de Parámetros está basado en ARGOT (Shaefer, 1987), algoritmo que utiliza un operador de ampliación bastante parecido al descrito anteriormente, pero que además incluye un *operador de ampliación inverso* que expande el espacio de búsqueda en el caso de existir poca convergencia. ARGOT también manipula el número de bits utilizados en la codificación, de forma que si en la población existe mucha diversidad, este número se reduce, y se eleva cuando se detecta la convergencia.

La *codificación delta* (Whitley y otros, 1991) es otro método que sigue la anterior línea para refinar la precisión. Las cadenas de genes binarios no codifican directamente valores de parámetros, sino que codifican *valores delta*. Estos valores se suman a una solución parcial fija, llamada *solución interina*, para obtener un individuo del espacio de búsqueda. La primera ejecución de un AG basado

en codificación delta es igual que la ejecución de un AG estándar. Después, cada vez que se comienza una nueva ejecución, el AG se inicializa con cromosomas que codifican valores delta sobre la mejor solución encontrada en la ejecución más reciente, que se convierte de este modo en solución interina. Así, la población inicial de una ejecución dada forma un hipercubo alrededor de la mejor solución encontrada en la ejecución anterior. Los hipercubos delta producidos pueden reducirse o alargarse cada vez que el AG reinicializa la población.

Andre y otros (2001) propone otro método en el que, tras un número de soluciones generadas, el espacio de búsqueda se reduce para abarcar las 10 mejores soluciones. Si en las siguientes generaciones se mejora la mejor solución encontrada, entonces se vuelve a reducir el espacio, en otro caso se vuelve al espacio de búsqueda utilizado anteriormente. Además, cada vez que se modifica el espacio de búsqueda se reinicializan todas las soluciones de la población de forma aleatoria.

Finalmente, destacamos que Lin y otros (1994) presentan un modelo de adaptación de la representación basado en el uso de subpoblaciones que representan el espacio de búsqueda con distinta resolución.

Ejecuciones Múltiples Secuenciales

Cuando la población converge, el AG pierde gran parte de su eficacia; los operadores genéticos no pueden producir la adecuada diversidad para acceder a nuevas zonas del espacio de búsqueda y de esta forma el algoritmo reitera su búsqueda sobre la misma zona. En estos momentos, la utilización de los recursos del AG no está justificada por la calidad de los cromosomas que se puedan seguir encontrando en esta zona. Estos recursos se podrían utilizar mejor en la búsqueda de nuevas zonas, con una nueva población (Maresky y otros, 1995).

Goldberg (1989c) sugiere *reinicializar* la ejecución de un AG usando una nueva población cuando su nivel de convergencia es sustancial. La reinicialización se hace con individuos generados aleatoriamente junto con los mejores individuos de la población convergida anterior. Eshelman (1991) utiliza un mecanismo similar. Tras la convergencia, el mejor elemento encontrado se utiliza como plantilla para reinicializar la población al completo. Para formar cada individuo de la nueva población, se muta aleatoriamente una porción del 35 % de este elemento.

Krishnakumar (1989) propone los μAGs, AGs con pequeñas poblaciones basadas en reinicialización. Un μAG utiliza una población de 5 elementos, $p_m = 0$ y $p_c = 1$. La selección se lleva a cabo mediante cuatro competiciones entre elementos adyacentes en la población junto con el modelo elitista.

Maresky y otros (1995) proponen que la reinicialización se produzca sólo sobre un porcentaje de la población. Para ello, cada gen se reinicializa de acuerdo a una determinada probabilidad. Así se mantiene cierta información de la ejecución anterior, a la vez que se introduce nuevo material genético. Para detectar la convergencia se tienen en cuenta las siguientes características de la ejecución del AG: 1) número de evaluaciones desde la última mejora producida sobre el mejor elemento por generación, 2) la magnitud relativa de las mejoras y 3) tamaño de la población. Los experimentos mostraron que este modelo utiliza los

recursos del AG de forma eficaz, sin embargo, se destaca que los porcentajes de reinicialización óptimos dependen de cada problema, por lo que dejan abierto el problema de la determinación de valores robustos para este parámetro de control.

Finalmente, recordar que los modelos que realizan una adaptación de la representación, Codificador Dinámico de Parámetros, ARGOT y la codificación delta, son también ejemplos de reinicialización.

Separación Espacial

No parece que en la naturaleza se haya producido un problema análogo a la convergencia prematura, sobre todo después de tres billones de años de evolución. La necesidad de métodos especiales para evitar la aparición de la convergencia prematura en los AGs sugiere que debe existir un problema básico en la forma en la que éstos modelan la evolución (Collins, 1992). Esta idea ha motivado el estudio de AGs basados en principios evolutivos más complejos.

El siglo pasado ha asistido a una controvertida discusión en el campo de la *genética de poblaciones* sobre dos de las teorías propuestas para explicar el proceso de la *evolución natural* de Fisher (1930) y por otro, la teoría de la balanza cambiante de Wright (1934).

Para Fisher, el motor principal de la evolución es la selección natural que actúa sobre *grandes poblaciones* de organismos, las cuales muestran un comportamiento estocástico bajo. La evolución se produce en una sucesión de pequeños pasos, cada uno de los cuales produce pequeñas diferencias, que al acumularse, conducen eventualmente a grandes cambios en la población. Fisher opina que la evolución actúa de forma idónea sobre grandes poblaciones de organismos, ya que las fluctuaciones que se producen sobre éstas son débiles y cada alelo (valor de un gen) se muestrea de forma equitativa en combinación con otros muchos.

El modelo de Wright es más complejo y sigue una filosofía contraria al anterior. Para Wright, las grandes poblaciones naturales de organismos raramente actúan de forma unificada, sino que están constituidas por subpoblaciones semiaisladas de tamaño relativamente pequeño que se comunican entre sí por medio de migraciones de individuos. Según su modelo, el proceso de evolución consiste en dos fases. Durante la primera fase, la frecuencia de alelos en cada subpoblación se estabiliza en un punto particular. Fortuitamente, una de las subpoblaciones puede estabilizarse en un punto que corresponde con un óptimo local alto. Entonces, comienza la segunda fase; al tener esta subpoblación una adaptación alta, producirá un exceso de individuos que emigrarán hacia las otras subpoblaciones, expandiendo de este modo, sus estructuras genéticas favorables hacia la totalidad de la población. Finalmente, todo el proceso vuelve a comenzar de nuevo. El pequeño tamaño de las subpoblaciones permite que la convergencia de las subpoblaciones juegue un papel importante sin el compromiso de la convergencia total de la población. Incluso si cada una de las subpoblaciones converge, lo hará en un lugar diferente al del resto de ellas, de forma que la diversidad de la totalidad de la población permanece intacta. Con esta teoría, Wright trata de explicar la capacidad de las poblaciones naturales para escapar de óptimos

locales y descubrir nuevas estructuras genéticas, superando así, los efectos no lineales y de desorientación entre los genes.

El modelo de Wright ha jugado un importante papel en el desarrollo de los denominados AGs con *separación espacial* (Collins, 1992; Davidor, 1991; Manderick y Spiessens, 1989; Mühlenbein, 1989, 1991; Mühlenbein y otros, 1991; Yang y otros, 2004). En general, la idea es dividir la población en distintas subpoblaciones (de tamaño relativamente pequeño), cada una de las cuales se procesa, de forma independiente, por medio de un AG. Además, se mantiene cierta clase de comunicación entre las subpoblaciones. Estos modelos, basados en una separación espacial, presentan una ventaja determinante contra la convergencia prematura: *la preservación de la diversidad* producida gracias al semiaislamiento de las subpoblaciones. Otra ventaja de estos modelos es que se pueden implementar con facilidad sobre hardware paralelo, obteniendo de este modo, una mejora sustancial en cuanto al tiempo de computación.

Otra de las teorías naturales sobre la evolución que ha tenido un impacto importante en el desarrollo de AGs basados en separación espacial es la *teoría del equilibrio interrumpido* (Eldredge y Gould, 1972). Según esta teoría, la evolución se caracteriza por largos periodos de relativa estabilidad interrumpidos por periodos de rápidos cambios asociados con eventos de especialización. Cohoon y otros (1987) subrayan que los AGs también tienden hacia la estabilidad o convergencia prematura. Según sus autores, la división de la población en subpoblaciones permite la aparición de distintas '*especies*' de cromosomas. La inclusión de individuos de diferentes especies en una subpoblación donde se ha producido la convergencia, proporciona nuevos bloques constructores, y además, los inmigrantes cambiarán la estructura genética de la subpoblación. Estos dos factores juntos producirán 'un evento de especialización' que se acompañará de un rápido periodo de evolución. Esto explica la capacidad de los AGs con separación espacial para evitar el problema de la convergencia prematura. En esta línea, Mühlenbein (1991) explica que manteniendo distintas subpoblaciones aisladas durante cierto tiempo, la diversidad de la población será alta. Después de producirse migraciones entre las subpoblaciones, el cruce de los inmigrantes y los individuos nativos de cada subpoblación ocasionará el descubrimiento de elementos con buena adaptación, los cuales no podrían haber sido descubiertos por cualquiera de las subpoblaciones de forma aislada. Según Mühlenbein, esto explica que los grandes cambios en las subpoblaciones se produzcan precisamente después de las migraciones, como indica Cohoon y otros (1987).

Estas ideas han dado lugar a un gran número de AGs distribuidos (Affenzeller y Wagner, 2004; Herrera y Lozano, 2000a; Herrera y otros, 1999). Alba y Troya (1999) y Alba y Tomassini (2002) realizan una revisión de diferentes AGs paralelos. En particular, Herrera y Lozano (2000a) presentan modelos de AGs distribuidos donde además, cada subpoblación usa operadores de cruce que proporcionan diferentes niveles de diversificación e intensificación.

1.6.4. Algoritmos Genéticos Estacionarios

El AG generacional crea nuevos descendientes de los miembros de una población antigua usando operadores genéticos y coloca estos individuos en una

```
AGE(f){
   t ← 0;
   P(t) ← generarPoblacionInicialAelatoria();
   evaluar(P(t));

   while (no se cumpla la condición de parada) {
      t ← t + 1;
      padres ← seleccionarPadres(P(t − 1));
      d ← crearDescendiente(padres);
      evaluar(d);
      r ← seleccionarIndividuoAReemplazar(P(t − 1));
      decidir entre P(t) ← P(t − 1) \ {r} + {d} o P(t) ← P(t − 1);
   }
}
```

Figura 1.20: Seudocódigo de un AGE

nueva población que se convierte en la antigua cuando se crea una entera nueva. Existe una variante a este modelo conocida como *AG estacionario* (AGE) (Syswerda, 1989; Whitley, 1989). En un AGE, típicamente se inserta un único nuevo miembro en la población en cada iteración. Una *estrategia de reemplazo* define qué miembro de la población actual debe perecer (dejar una vacante) para tener espacio donde insertar el nuevo descendiente (u ocupar la vacante) de la próxima iteración. Los AGEs son sistemas con solapamiento, dado que padres y descendientes compiten por sobrevivir. La Figura 1.20 muestra el seudocódigo de un AGE.

La *estrategia de reemplazo* selecciona el individuo que puede perecer (por ejemplo, reemplazar el peor, el más antiguo, o un individuo elegido aleatoriamente). Posteriormente, uno puede elegir la *condición de reemplazo* (por ejemplo, reemplazar si el nuevo individuo es mejor o realizar el reemplazo incondicionalmente).

En los AGEs, el equilibrio entre presión selectiva y diversidad en la población se ha tratado poniendo énfasis en los padres que generan el descendiente de cada iteración (selección de padres) y en la fase de reemplazo. De hecho, diferentes estudios han mostrado que el rendimiento superior de los AGEs frente a los AGs generacionales se debe a su alta presión selectiva y a cambios en el equilibrio exploración/explotación producidos por el uso de diferentes técnicas de selección de padres y estrategias de reemplazo, y no por ser sistemas con solapamiento (De Jong y Sarma, 1993). Los mecanismos de selección de padres y estrategias de reemplazo propuestos en la literatura suelen promover:

- sólo diversidad,
- sólo presión selectiva o
- presión selectiva y diversidad en la población simultaneamente.

A continuación describimos varias técnicas de selección de padres y estrategias de reemplazo para AGEs, propuestos en la literatura, haciendo hincapié en si promueven diversidad, presión selectiva o diversidad y presión selectiva simultaneamente.

Mecanismos de Selección de Padres

En esta sección, describimos diferentes mecanismos de selección para AGEs, propuestos en la literatura, que proporcionan sólo diversidad, sólo presión selectiva o ambas.

Sólo Diversidad. *Selección aleatoria* selecciona un individuo aleatorio de la población que el descendiente generado reemplazará.

Sólo Presión Selectiva. *Selección por torneo* es uno de los mecanismos de selección de padres más usados en AGEs, quizás por su simplicidad. La idea básica de este esquema es bastante directa. Se selecciona un grupo de n_T individuos de la población de forma aleatoria. Se comparan los valores de aptitud de los individuos de este grupo, seleccionando el individuo con el mejor valor de aptitud. Típicamente, selección por torneo selecciona sólo dos individuos para la comparación, esto es, $n_T = 2$ (selección por torneo *binario*).

Diversidad y Presión Selectiva. La mayoría de los mecanismos de selección de padres, presentados en la literatura, pueden incluirse en este grupo. Describiremos tres ejemplos:

- *Selección de extremos* (en inglés *Disruptive Selection*) (Kuo y Hwang, 1996). A diferencia de los mecanismos de selección convencionales, Selección de extremos ofrece una mayor probabilidad de selección a los individuos con mejores y peores valores de aptitud, que a las soluciones moderadas. Esto se lleva a cabo modificando la función objetivo de cada cromosoma C como sigue:

$$f'(C) = |f(C) - \overline{f}| \tag{1.11}$$

  donde $\overline{f}$ es el valor medio de la función objetivo de los individuos de la población. Después, se aplica selección por torneo considerando la función objetivo modificada para seleccionar un individuo de la población.

- *Selección uniforme en aptitud* (en inglés *Fitness Uniform Selection Scheme*) (Hutter, 2002), también descrito en la Sección 1.6.3, produce presión selectiva sobre valores de aptitud dispersamente representadas por los miembros de la población. Se define como sigue: sean f_{min} y f_{max} el menor y mayor valor de aptitud de la población actual, respectivamente. Se selecciona un valor aleatorio v de forma uniforme en el intervalo $[f_{min}, f_{max}]$. Después, se selecciona el individuo de la población con el valor de aptitud más cercano a v. Este método produce una alta presión selectiva sobre los mejores valores de aptitud sólo si hay pocos individuos

de alta calidad y la presión selectiva se reduce automáticamente cuando el número de buenos individuos aumenta. En una población típica, evolucionada con selección uniforme en aptitud, hay muchos individuos de baja calidad y sólo unos pocos con alto valor de aptitud. Se favorece a los individuos de buena calidad hasta que la población toma valores de aptitud uniformemente distribuidos. Si en alguna ocasión se genera un nuevo individuo con un valor de aptitud superior, entonces el método vuelve a producir una alta presión selectiva sobre el nuevo individuo.

- *Selección orientada a la diversidad* (Shimodaira, 1999) es un mecanismo, comentado en la Sección 1.6.3, diseñado para su uso en AGs generacionales. Una variante para los AGEs realiza dos torneos para obtener dos cromosomas competitivos, y después se selecciona, el menos similar al mejor cromosoma de la población, de acuerdo a alguna medida de distancia.

Estrategias de Reemplazo

A continuación, revisamos tres estrategias de reemplazo propuestas en la literatura para promover diversidad, producir presión selectiva u ofrecer ambas.

Sólo Diversidad. Estrategia *primero en entrar, primero en salir* (en inglés *first-in-first-out*, FIFO) (De Jong y Sarma, 1993). El descendiente reemplaza al miembro de la población más antiguo.

Sólo Presión Selectiva. *Reemplazar el peor* reemplaza el peor individuo de la población sólo si el nuevo es mejor. Al usar esta estrategia de reemplazo, la población mantiene las mejores soluciones generadas en cualquier momento de la ejecución del AGE. Goldberg y Deb (1991) sugieren que borrar el peor individuo de la población produce una alta presión selectiva, incluso cuando los padres se seleccionan de forma aleatoria.

Presión Selectiva y Diversidad. *Selección por torneo restringido* (De Jong, 1975; Harik, 1995): Supongamos que X^n es el individuo a incluir en la población. Entonces, selección por torneo restringido escoge n_T miembros de la población de forma aleatoria. De entre los miembros seleccionados, se busca el más similar a X^n, sea éste X^r. Entonces, X^n compite con X^r, y si X^n gana, lo reemplaza en la población. Éste es un *método de agrupamiento* (en inglés *crowding*) (Cedeño y otros, 1995; De Jong, 1975; Mahfoud, 1995) que hace que los nuevos individuos tengan mayor probabilidad de reemplazar individuos de la población similares a ellos según su genotipo. De esta manera, en la población no se produce un exceso de soluciones similares.

1.6.5. AGs para Optimización Continua

En el mundo real, muchos de los problemas de optimización presentan variables que tienen un dominio continuo, esto es, sus valores se toman del espacio

real ($\Re$). A este tipo de problemas se les conoce como *problemas de optimización continua*.

Aunque en su formulación inicial, los AGs utilizaban el alfabeto binario para codificar las soluciones, otro tipo de codificaciones, como la real (Davis, 1991b; Deb y Beyer, 2001; Deb, 2005; Herrera y otros, 1998; Michalewicz, 1992), se han tenido en cuenta. El estudio de *AGs con codificación real* (AGCRs) ha recibido especial atención por muchos investigadores (Chelouah y Siarry, 2000; Deb y Tiwari, 2008; Herrera y otros, 2005; Hervás-Martínez y Ortiz-Boyer, 2005; Poojari y Varghese, 2008; Someya y Yamamura, 2005; Winter y otros, 2005; Yang y Kao, 2000), y recientemente hay un creciente interés en resolver problemas del mundo real mediante este tipo de algoritmos.

En esta sección, analizamos los AGCRs, específicos para la optimización continua, como alternativa a la aplicación de AGs con codificación binaria en este tipo de problemas. Expondremos las ventajas y desventajas presentadas en la literatura sobre cada uno de estos tipos de AGs. Posteriormente, describiremos un amplio conjunto de operadores de cruce y de mutación, junto a sus propiedades, diseñados para su uso en AGCRs.

Uso de la Codificación Binaria

En esta sección, exponemos los argumentos teóricos que se han esgrimido para usar la codificación binaria, los cuales han propiciado que esta codificación haya predominado durante buena parte de la historia de los AGs. Además, describimos las desventajas que se presentan con su utilización en problemas con variables continuas y cuando se representan espacios de búsqueda con un cardinal finito que no es potencia de dos. Finalmente, incluimos algunas críticas a los argumentos teóricos presentados en la Sección 1.6.2.

Codificación Binaria. Principalmente, se han utilizado dos tipos de codificación binaria a la hora de representar un parámetro x_i perteneciente a un intervalo continuo $D_i = [a_i, b_i]$: *el código binario* (Goldberg, 1989b; Holland, 1975) y *el código Gray* (Caruana y Schaffer, 1988).

Previamente a la codificación, se realiza una transformación del intervalo $[a_i, b_i]$ en el conjunto $\{0, \ldots, 2^{L_i}\}$ (L_i es el número de bits en la codificación), y son los elementos de este conjunto los que se codifican usando cualquiera de los tipos de códigos anteriormente citados. La transformación realizada implica una *discretización* del intervalo $[a_i, b_i]$, la cual lleva asociada la precisión ρ siguiente:

$$\rho = \frac{b_i - a_i}{2^{L_i} - 1}. \tag{1.12}$$

Una cadena $(c_1, \ldots, c_{L_i})$, $c_i \in \{0, 1\}$, codificada en binario representa el entero $e \in \{0, \ldots, 2^{L_i} - 1\}$ siguiente:

$$e = \sum_{j=1}^{L_i} c_j 2^{(L_i - j)}. \tag{1.13}$$

La distancia Hamming de dos enteros consecutivos codificados bajo el código Gray difiere en uno. La fórmula de conversión de una cadena $(c_1, \ldots, c_{L_i})$ en código binario a una cadena $(g_1, \ldots, g_{L_i})$ en código Gray es:

$$g_k = \begin{cases} c_1 & \text{Si } k = 1 \\ c_{k+1} \oplus c_k, & \text{Si } k > 1 \end{cases} \tag{1.14}$$

donde $\oplus$ nota la suma módulo 2. La conversión inversa viene definida por la siguiente expresión:

$$c_k = \sum_{j=1}^{k} g_j, \tag{1.15}$$

donde el sumatorio se realiza en módulo 2. En la Tabla 1.3 se muestran las codificaciones de los enteros del 0 hasta el 15 usando el código binario y Gray.

Tabla 1.3: Comparación entre el código binario y el código Gray

Entero	**Binario**	**Gray**
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Argumentos para Usar el Alfabeto Binario. Principalmente, existen dos argumentos para usar el alfabeto binario. Primero, este alfabeto maximiza el nivel de paralelismo implícito (una propiedad esencial de los AGs) (Goldberg, 1991). Más tarde veremos que existen algunos argumentos en contra de esto. Segundo, los caracteres alfabéticos asociados con un alfabeto de cardinal alto no se representan adecuadamente en una población finita. Esto fuerza el uso

de poblaciones con tamaños grandes (Reeves, 1993), lo cual afecta a la eficacia del AG. Este inconveniente es más destacado en problemas donde la evaluación de los cromosomas es muy costosa. En estos casos hay que usar alfabetos con cardinal bajo.

Holland (1975) sugiere una importante interpretación del teorema de los esquemas: *los AGs procesan esquemas en vez de cromosomas.* Aunque, superficialmente parece que los AGs con codificación binaria procesan sólo los cromosomas individuales que están presentes actualmente en la población, realmente procesan en paralelo una gran cantidad de información útil relacionada con los esquemas invisibles. Esta importante propiedad se denomina *paralelismo implícito* y es uno de los fundamentos más importantes subyacentes en la búsqueda genética. El siguiente resultado asocia el máximo nivel de paralelismo implícito con el alfabeto binario (Goldberg, 1991):

Teorema 2 *Para un contexto dado de información, las cadenas codificadas usando alfabetos con un cardinal pequeño representarán un número de esquemas mayor que si se codifican usando alfabetos de cardinal más alto.*

Un resultado importante derivado de este teorema es el siguiente (Goldberg, 1991):

Resultado 2 *El alfabeto binario ofrece el máximo número de esquemas por bit de información.*

Debido a que la propiedad de paralelismo implícito significa que los AGs procesan esquemas en vez de cromosomas individuales y de acuerdo al Resultado 2, es posible deducir que el alfabeto binario maximiza el nivel de paralelismo implícito, ya que permite muestrear el máximo número de esquemas por cromosoma en la población. De esta forma, se obtiene el máximo nivel de eficacia.

Por otro lado, existen muchos problemas donde el número de evaluaciones de cromosomas está limitado, por lo que se requiere el uso de poblaciones con un tamaño pequeño. Reeves (1993) estudia la especificación de un tamaño mínimo para garantizar que la población inicial (generada de forma aleatoria) tenga propiedades adecuadas para evitar que la conducta posterior del AG sea ineficaz. Para Reeves esta propiedad radica en el siguiente principio:

> *'Todo punto del espacio de búsqueda debe ser accesible desde la población inicial sólo a través del operador de cruce'.*

Dado un alfabeto $\mathcal{A}$ ($|\mathcal{A}| = q$), un tamaño de población N y una longitud para los cromosomas L, el cumplimiento de este principio es posible sólo si al menos para cada posición i ($i = 1, \ldots, L$) y para cada símbolo $s \in \mathcal{A}$, existe un cromosoma en la población inicial tal que la posición i contiene a s. Reeves calcula la probabilidad ($p^*(q, N, L)$) de que esto ocurra como sigue:

$$p^*(q, N, L) = \left(\frac{q! S(N, q)}{q^N} \right)^L , \tag{1.16}$$

donde $S(N,q)$ es el número de *Stirling* de segundo orden, el cual se genera por la ecuación recursiva

$$S(N+1,q) = S(N,q-1) + q \cdot S(N,q), q \geq 2 \text{ y } S(N,1) = 1, \forall N \geq 1. \quad (1.17)$$

Si se fija un valor mínimo, α, para la probabilidad anterior, entonces podemos estudiar la relación existente entre q y N. Es decir, dado un valor para q, se puede calcular el mínimo valor de N, que conserve la restricción sobre α. De esta forma, para cualquier alfabeto, se puede determinar el tamaño de población mínimo posible que mantenga un determinado nivel de eficacia representado por α. Para diferentes valores de L: $20, 40, \ldots, 200$, y diferentes valores de q: $2, 3, \ldots, 8$, Reeves determinó el valor mínimo de N (N_{min}) tal que $p^*(q, N_{min}, L) > \alpha$, para $\alpha = 0{,}95, 0{,}99$ y $0{,}999$. Los resultados mostraron que, cuanto más alto es q, más alto es el N_{min} que mantiene el mismo límite para p^*.

La principal conclusión de este resultado es que, para un problema dado, el esfuerzo computacional de un AG basado en un alfabeto con $q > 2$ (potencia de dos) para obtener una solución particular, será mayor que usando $q = 2$, ya que se necesita mantener un tamaño de población mayor. Esto es intuitivamente razonable. Por ejemplo, para que queden representados todos los posibles valores de un alfabeto con $q = 8$ en una posición i se necesitan 8 cromosomas, mientras que, si se utiliza $q = 2$, se representa la misma información con las combinaciones de tan solo dos cadenas (000 y 111), ya que con el operador de cruce se pueden generar valores que no están representados de forma explícita en la población inicial.

Tate y Smith (1993) definen una medida, llamada *cubrimiento esperado de alelos*, con la misma función que p^*, es decir, dado un alfabeto $\mathcal{A}$ ($|\mathcal{A}| = q$), determinar el grado de adecuación de un tamaño de población dado. Los estudios realizados utilizando esta medida mostraron que con codificaciones complejas se requieren poblaciones más grandes que con la codificación binaria, para alcanzar un determinado nivel de cubrimiento de alelos. Claramente, este resultado es igual al obtenido por Reeves. Para compensar los problemas asociados con cubrimientos deficientes de alelos, se propone ajustar la probabilidad de mutación. Se mostró que se pueden obtener beneficios sobre determinados problemas que requieren codificaciones no binarias, usando probabilidades de mutación mucho más altas que las que generalmente se usan en los AGs con codificación binaria, las cuales son bastante pequeñas (Goldberg, 1989b).

Problemas en Espacios de Búsqueda Continuos. Con independencia de si se usa el código binario o el Gray, el parámetro L_i determina la magnitud del espacio de búsqueda pero además, delimita la precisión de la solución obtenida (ρ depende de L_i). Esto puede ocasionar dificultades cuando se tratan problemas con altas dimensiones y requerimientos de gran precisión.

La actuación de un AG con codificación binaria frente a estos problemas puede ser ineficaz (Schraudolph y Belew, 1992). En las primeras etapas, el algoritmo consume mucho esfuerzo en evaluar los dígitos menos significativos de la codificación binaria de cada parámetro. Sin embargo, su valor óptimo dependerá de los dígitos más significativos, por lo que hasta que éstos no converjan

a su valor óptimo, su manipulación será de poca utilidad. Una vez que ocurre la convergencia de los dígitos más significativos, no existe necesidad de gastar más esfuerzo con ellos. Este comportamiento ideal no existe en los AGs con codificación binaria, puesto que en ellos la búsqueda se desarrolla poniendo igual atención en todos los dígitos. Con frecuencia, los dígitos menos significativos convergen en las primeras etapas hacia un valor arbitrario. Esto hace que la búsqueda no pueda proseguir de forma eficaz y, consecuentemente, que la precisión alcanzada sea inadecuada.

Un mecanismo para obtener precisión usando un AG con codificación binaria consiste en realizar ejecuciones iterativas del algoritmo de forma que en una ejecución ζ se utilizan los individuos *recodificados* obtenidos al finalizar la ejecución $\zeta - 1$ (véase la Sección de Adaptación en la Representación en 1.6.3).

Cuando se utiliza el código binario puede aparecer el efecto denominado *pendiente Hamming*, consistente en que dos valores adyacentes pueden diferir en cada uno de los bits de su codificación binaria. Por ejemplo, las cadenas 01111 y 10000 representan a los valores 31 y 32, respectivamente, y sin embargo, los valores de cada una de sus posiciones son distintos. Pueden darse situaciones en las que este efecto provoque problemas tales como la convergencia hacia elementos no óptimos, una de ellas se muestra en la Figura 1.21 (Goldberg, 1991).

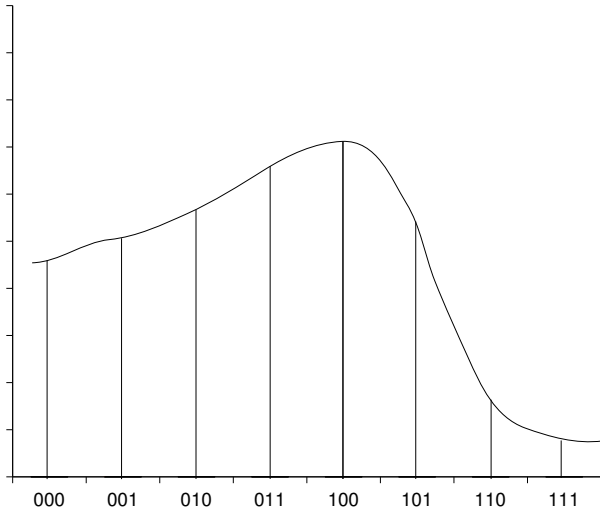


Figura 1.21: Función de evaluación con pendiente Hamming

Como muestra esta figura, los elementos situados a la izquierda poseen una función de evaluación por encima de la media ya que $f(0\star\star) > f(1\star\star)$. Por ello, en un principio, el AG con codificación binaria tiene muchas posibilidades de converger hacia el elemento 011. El óptimo global (100) está muy cercano al elemento 011, sin embargo, la distancia Hamming entre ellos es bastante grande. Se requieren tres cambios para lograr el acceso de uno al otro. Este acceso es improbable usando el operador de mutación. De esta forma, el AG con codificación binaria convergerá probablemente hacia el elemento 011. Este problema se

soluciona empleando el *código Gray* (Caruana y Schaffer, 1988), sin embargo, el uso de esta codificación introduce características no lineales con respecto a la recombinación (Goldberg, 1989a). Otra solución es usar un operador de mutación con una actuación más inteligente que mute con más frecuencia cromosomas prometedores, aunque de este modo se imita el comportamiento de un modelo de ascensión de colinas (Goldberg, 1991).

Además, cuando se utiliza el alfabeto binario para codificar un parámetro perteneciente a un conjunto finito discreto con cardinal distinto a una potencia de dos, algunos códigos pueden ser *redundantes*, es decir, al decodificarse darán valores no pertenecientes al dominio del parámetro. Consideremos, por ejemplo, un parámetro $x_i \in D$ donde $|D_i| = 10$. Se necesita un mínimo de 4 bits para codificar los elementos de D_i usando el alfabeto binario. Si se eligen los códigos 0000 hasta 1001 para codificar sus elementos, entonces ¿qué ocurre con los códigos 1010 hasta 1111?

La existencia de códigos redundantes plantea un problema ya que los operadores de cruce y mutación no garantizan que los códigos resultantes, después de su aplicación, no sean *redundates*. Se necesitan mecanismos que limiten la existencia de estos códigos en la población. Beasley y otros (1993b) citan tres posibilidades:

1. Descartar los cromosomas que contienen códigos redundantes.
2. Asignar a los cromosomas con códigos redundantes un valor bajo para la función de evaluación.
3. Asociar a los códigos redundantes un código válido.

Las dos primeras soluciones pueden ocasionar pérdidas de material genético importante, simplemente por el hecho de ir acompañado de códigos redundantes, lo cual podría provocar un comportamiento ineficaz en el AG. Existen distintos mecanismos para llevar a cabo la última posibilidad, entre los que destacan *la asociación fija* y *la asociación aleatoria*. En la primera, cada código redundante tiene asociado un código válido específico. Es un mecanismo simple, aunque tiene el inconveniente de que algunos elementos del dominio estarán representados por dos cadenas distintas, mientras que otros lo estarán por sólo una. Con la técnica aleatoria, cada código redundante se asocia de forma aleatoria con un código válido. De esta forma, se evita el problema de *la tendencia de la representación* (Sección 1.6.3), aunque supone que la información transmitida de padres a hijos es menos significativa. Existe una alternativa híbrida denominada *asociación probabilista*, consistente en que cada cadena (redundante o no) al decodificarse tiene asociada dos elementos del dominio, en cada momento se escoge cualquiera de ellos aleatoriamente. Así se asegura que todos los elementos del dominio están igualmente representados.

Críticas en Contra de que la Codificación Binaria Favorezca la Propiedad de Paralelismo Implícito. A continuación, presentamos tres críticas en contra de los resultados que apuntaban al alfabeto binario como el alfabeto que más favorece la propiedad de paralelismo implícito de los AGs.

- *Los esquemas pueden no tener sentido*: Sea S un espacio de búsqueda con $|S| = 2^n$ y B^n el conjunto de las cadenas binarias de longitud n. Entonces existen $2^n!$ posibles representaciones de S usando B^n asociadas al conjunto

$$R = \{\theta | \theta : S \rightarrow B^n, \theta \text{ es inyectiva}\}. \tag{1.18}$$

  Liepins y Vose (1990) presenta resultados teóricos que muestran la importancia de seleccionar buenas representaciones (funciones θ), las cuales además dependen de cada problema. Conocer la mejor representación para un problema es equivalente a conocer la forma de resolver el propio problema. En general, el usuario del AG no puede suministrar la mejor respresentación. Si se selecciona una representación de R de forma aleatoria, quizás los esquemas no tengan significado (Radcliffe, 1992), es decir, los esquemas no relacionaran cromosomas con adaptaciones correladas. Bajo estas circunstancias, no está claro que la recolección de información sobre la eficacia de un subconjunto de cromosomas (esquemas) proporcione información sobre la eficacia del resto de ellos. Todos los esquemas del mismo orden tenderán a tener aproximadamente la misma adaptación. De esta forma, el mecanismo evolutivo del AG no podrá hacer distinciones entre ellos (Tate y Smith, 1993), produciéndose una búsqueda poco efectiva.

- *Aparecen esquemas inapropiados*: Un espacio con 2^n elementos tienen 2^{2^n} subconjuntos. Si se utiliza una representación basada en un alfabeto de k elementos existirán un máximo de $(k+1)^n$ subconjuntos que se consideran esquemas. Pueden existir subconjuntos del espacio que agrupen cromosomas prometedores, compartiendo propiedades de las que se derive su buen comportamiento. Sin embargo, es posible que estos subconjuntos no formen un esquema (excepto el más general $\star \ldots \star$) o incluso que no exista un esquema que los contenga (Radcliffe, 1992). Por ejemplo, si codificamos los enteros 0 hasta 15 usando 4 dígitos, los representantes del 7 y 8 (0111 y 1000) no comparten ningún esquema (excepto el más general $\star \ldots \star$).

- *El teorema de los esquemas es válido con otras definiciones de esquema*: El máximo grado de paralelismo implícito logrado por el alfabeto binario está soportado bajo la definición clásica de esquema. Esta restricción es inapropiada ya que pueden encontrarse otras definiciones de esquema para diferentes tipos de codificaciones que permitan que esta propiedad también se cumpla (Antonisse, 1989; Radcliffe, 1991; Vose, 1991).

La Codificación Real

Una de las alternativas más importantes a la codificación binaria es la codificación real. Este tipo de codificación parece particularmente natural para resolver problemas con variables en espacios de búsqueda continuos. Un cromosoma es un vector de números reales cuyo tamaño es igual a la longitud del vector solución del problema; de esta forma, *cada gen representa una variable*

del problema. Los valores de los genes están forzados a permanecer en el intervalo establecido para las variables a las cuales representan, de modo que los operadores genéticos tendrán que preservar este requerimiento.

La utilización de la codificación real aparece inicialmente en aplicaciones particulares, tales como en Lucasius y Kateman (1989) para problemas de quimiometría. Posteriormente, los AGCRs se han usado principalmente en problemas de optimización numérica sobre dominios continuos (Davis, 1991b; Eshelman y Schaffer, 1993; Herrera y otros, 1995; Janikow y Michalewicz, 1991; Michalewicz, 1992; Mühlenbein y Schlierkamp-Voosen, 1993; Wright, 1991). Hasta 1991 no se realizó ningún estudio teórico sobre estos algoritmos; su utilización provocaba cierta controversia: los investigadores familiarizados con la teoría fundamental de los AGs no comprendían lo buenos resultados empíricos mostrados por los AGCRs, ya que esta teoría sugería, como se ha visto anteriormente, que los alfabetos de cardinal bajo debían ser más efectivos que aquéllos con cardinales altos. Más tarde, se presentaron distintas herramientas para el tratamiento teórico de los AGCRs (Eshelman y Schaffer, 1993; Goldberg, 1991; Radcliffe, 1991; Wright, 1991), lo que permitió corroborar su buen comportamiento.

Ventajas de la Codificación Real

El uso de parámetros reales posibilita manejar grandes dominios (incluso dominios desconocidos) para las variables, tarea difícil en las implementaciones binarias, donde extender el dominio significaría sacrificar precisión, supuesta una longitud fija para los cromosomas. Otra ventaja es la capacidad para explotar la *gradualidad* de las funciones con variables continuas, donde el concepto gradualidad se refiere a que pequeños cambios en las variables se corresponden con pequeños cambios en la función. En esta línea, se destaca la habilidad de los AGCRs para el *ajuste local* de las soluciones, existiendo operadores de mutación (mutación no uniforme, de Michalewicz (1992)) que permiten que ésta se realice de forma más rápida y eficaz que en los AGs con codificación binaria, donde el ajuste es difícil debido al efecto pendiente Hamming (Sección 1.6.5)

Usando codificación real, la representación de las soluciones está muy cercana a la formulación natural de muchos problemas, es decir, no existen diferencias entre el *genotipo* y el *fenotipo*. Así se evita el proceso de codificación y decodificación binaria que se debe producir en los AGs con este tipo de codificación, de manera que aumenta la rapidez del AG. Radcliffe (1992) deduce que no es necesario que existan diferencias entre el genotipo y el fenotipo para que se produzca evolución. Por ello, no se justifica que los operadores genéticos deban definirse en razón a la representación escogida. Muy al contrario, el autor aboga por *definir tanto los operadores genéticos como los conceptos de esquema directamente sobre el espacio de los fenotipos, siempre y cuando esto sea posible*. Los operadores genéticos y los conceptos de esquema presentados para los AGCRs en la literatura, concuerdan con la idea de Radcliffe.

Antonisse (1989) explica que mientras gran parte del trabajo realizado por la comunidad de la inteligencia artificial ha consistido en desarrollar representaciones sumamente expresivas y relativamente complejas, la codificación binaria es resueltamente simple. Este autor presenta una nueva interpretación de esquema

que anula la superioridad teórica del alfabeto binario en favor de alfabetos con cardinal alto. Además, destaca la importancia de la expresividad de la codificación:

> *... Esta interpretación hace que la teoría concuerde con la intuición de que cuanto más expresivo sea un lenguaje, el aparato que proporciona para la adaptación será más potente, y estimula la exploración de esquemas de codificación alternativos en la investigación de los AGs.*

Claramente, debido a que con el uso de la codificación real el genotipo y el fenotipo son la misma cosa, el nivel de expresividad alcanzado es muy alto.

A estas ideas hay que unir el tema de la relación entre los AGs y el *conocimiento específico del problema*. Para Davis (1989), la mayoría de los problemas del mundo real no pueden manejarse usando la codificación binaria, ya que muchos de estos problemas poseen un conocimiento específico útil a la hora de considerar transformaciones de las soluciones en el dominio. Los AGs con codificación binaria son indiferentes ante el conocimiento específico del problema, por ello, las técnicas de optimización que sí lo consideran pueden superarlos. Davis opina que *se debería incluir conocimiento específico del problema en los AGs añadiéndolo en la fase de decodificación o expandiendo el conjunto de operadores genéticos a utilizar*. La utilización de la codificación real permite integrar fácilmente conocimiento específico del problema en los AGCRs para manejar problemas con restricciones no triviales.

Operadores de Cruce para Codificación Real

En la literatura, podemos encontrar muchas propuestas de operadores de cruce para AGCRs. Aquí describimos algunos de ellos. Supongamos que los cromosomas $C_1 = (c_1^1, \ldots, c_n^1)$ y $C_2 = (c_1^2, \ldots, c_n^2)$ se han seleccionado para aplicarles el operador de cruce. A continuación, se muestran los efectos de diferentes operadores de cruce para AGCRs sobre estos cromosomas. Hay que notar que cada operador produce un número distinto de hijos, por lo que a veces, se hace necesaria una selección para obtener los cromosomas que formarán parte de la población. Esta selección se denominará *selección de descendientes*.

Cruce Plano (Radcliffe, 1991). Se genera el hijo, $H = (h_1, \ldots, h_i, \ldots, h_n)$, donde h_i es un valor escogido de forma aleatoria (uniformemente) del intervalo $[c_i^1, c_i^2]$.

Cruce Simple (Michalewicz, 1992; Wright, 1991). Aleatoriamente se elige una posición $i \in \{1, 2, \ldots, n-1\}$ y se generan

$$H_1 = (c_1^1, c_2^1, \ldots, c_i^1, c_{i+1}^2, \ldots, c_n^2), \text{ y } H_2 = (c_1^2, c_2^2, \ldots, c_i^2, c_{i+1}^1, \ldots, c_n^1). \quad (1.19)$$

Cruce Aritmético (Michalewicz, 1992). Se generan dos descendientes, $H_k = (h_1^k, \ldots, h_i^k, \ldots, h_n^k)$ $k = 1, 2$, donde $h_i^1 = \lambda c_i^1 + (1-\lambda)c_i^2$ y $h_i^2 = \lambda c_i^2 + (1-\lambda)c_i^1$. λ es una constante (cruce aritmético uniforme) o varía en función del número de generaciones realizadas (cruce aritmético no uniforme).

Cruce BLX-α (Eshelman y Schaffer, 1993). Se genera un solo descendiente, $H = (h_1, \ldots, h_i, \ldots, h_n)$, donde h_i es un número escogido de forma aleatoria (uniformemente) del intervalo $[c_{min} - I\alpha, c_{max} + I\alpha]$, donde $c_{max} = max(c_i^1, c_i^2)$, $c_{min} = min(c_i^1, c_i^2)$ e $I = c_{max} - c_{min}$. El cruce BLX-0,0 es igual al cruce plano.

Cruce Lineal (Wright, 1991). Se generan tres hijos, $H_k = (h_1^k, \ldots, h_i^k, \ldots, h_n^k)$ con $k = 1, 2, 3$, donde $h_i^1 = \frac{1}{2}c_i^1 + \frac{1}{2}c_i^2$, $h_i^2 = \frac{3}{2}c_i^1 - \frac{1}{2}c_i^2$ y $h_i^3 = -\frac{1}{2}c_i^1 + \frac{3}{2}c_i^2$.

Cruce Discreto (Mühlenbein y Schlierkamp-Voosen, 1993). h_i se genera de forma aleatoria (uniformemente) escogiendo un elemento del conjunto $\{c_i^1, c_i^2\}$.

Cruce Lineal Extendido (Mühlenbein y Schlierkamp-Voosen, 1993). $h_i = c_i^1 + \alpha(c_i^2 - c_i^1)$ y α es un valor escogido de forma aleatoria (uniformemente) en el intervalo $[-0{,}25, 1{,}25]$.

Cruce Intermedio Extendido (Mühlenbein y Schlierkamp-Voosen, 1993). $h_i = c_i^1 + \alpha_i(c_i^2 - c_i^1)$ y α_i es un valor escogido de forma aleatoria (uniformemente) en el intervalo $[-0{,}25, 1{,}25]$. Este operador es igual a BLX-0,25.

Cruce Heurístico (Wright, 1991). Supongamos que la función de evaluación de C_1 es mejor que la de C_2. Entonces $h_i = r \cdot (c_i^1 - c_i^2) + c_i^1$, donde r es un número aleatorio perteneciente al intervalo $[0, 1]$.

Cruce Lineal BGA (Schlierkamp-Voosen y Mühlenbein, 1994). Bajo las mismas consideraciones anteriores, $h_i = c_i^1 \pm rang_i \cdot \gamma \cdot \Lambda$, donde $\Lambda = \frac{c_i^2 - c_i^1}{\|C_1 - C_2\|}$, el signo '$-$' se elige con probabilidad 0,9, $rang_i$ normalmente se toma $0{,}5 \cdot (b_i - a_i)$ y $\gamma = \sum_{k=0}^{15} \alpha_k 2^{-k}$, donde $\alpha_i \in \{0, 1\}$, $i = 1, \ldots, 15$, se genera aleatoriamente con $p(\alpha_i = 1) = \frac{1}{16}$.

Cruce Difuso (Voigt y otros, 1995). Este operador se inspira en la teoría de los conjuntos difusos (Zimmermann, 1990). La probabilidad de que h_i tenga un valor v_i está dada por la distribución binomial $p(v_i) \in \{\phi(c_i^1), \phi(c_i^2)\}$, donde $\phi(c_i^1)$ y $\phi(c_i^2)$ son dos distribuciones de probabilidad triangulares con modas c_i^1 y c_i^2, respectivamente. Suponiendo que $c_i^1 \leq c_i^2$, entonces los posibles valores para v_i cumplen:

$$c_i^1 - d \cdot |c_i^2 - c_i^1| \leq v_i \leq c_i^1 + d \cdot |c_i^2 - c_i^1|, \tag{1.20}$$

$$c_i^2 - d \cdot |c_i^2 - c_i^1| \leq v_i \leq c_i^2 + d \cdot |c_i^2 - c_i^1|, \tag{1.21}$$

donde $d \geq 0{,}5$.

Esquema de Emparejamiento No Aleatorio (Karlin, 1968). La idea básica consiste en combinar los dos padres de forma determinista; un hijo se obtiene a través de la combinación convexa de los padres, usando dos matrices de pesos cuyos elementos deben cumplir determinadas propiedades. Los operadores de cruce aritmético y lineal son dos ejemplos de este tipo de operadores de cruce. Qi y Palmieri (1994) generaliza este modelo al permitir que los pesos de las matrices usadas para la combinación convexa se generen de forma aleatoria. Este nuevo esquema engloba los siguientes tipos de cruces (Qi y Palmieri, 1994):

1. El discreto, cuando las matrices de pesos son diagonales, contienen sólo ceros y unos y deben sumar la matriz identidad.
2. BLX-α y los dos extendidos, cuando las matrices son diagonales y pueden contener elementos distintos del cero y el uno, y además no se requiere que las dos matrices sumen la matriz identidad.

Cruce PBX-α (Lozano y otros, 2004). Es una variante del cruce BLX-α que crea un nuevo descendiente en la región apuntada por uno solo de los padres. Se genera un hijo, $H = (h_1, \ldots, h_i, \ldots, h_n)$, donde h_i es un valor escogido de forma aleatoria (uniformemente) del intervalo $[c_i^1 - I\alpha, c_i^1 + I\alpha]$, donde $I = c_{max} - c_{min}$.

Cruce UNDX (Kita y otros, 1999). Este cruce utiliza μ padres $(C^1, \ldots, C^\mu)$. Se calcula la media $X^m = (x_1, \ldots, x_n)$ de todos los padres, excepto el último, y los vectores $D^i = C^i - X^m$, para $i = 1, \ldots, \mu - 1$. Sean los cosenos $E^i = D^i/|D^i|$, para $i = 1, \ldots, \mu - 1$. Se calcula la longitud L del vector $C^\mu - X^m$ ortogonal a los E^i calculados. Se crean los vectores E^j, para $j = \mu, \ldots, n$, con n la dimensión del problema, que forman una base ortonormal al subespacio que generan los vectores E^i, con $i = 1, \ldots, \mu - 1$. Entonces, el descendiente se obtiene con la siguiente ecuación:

$$H = X^m + \sum_{i=1}^{\mu-1} w_i |D^i| E^i + \sum_{j=\mu}^{n} v_j D E^j, \tag{1.22}$$

donde w_i y v_i son variables aleatorias que siguen una distribución normal con media cero y varianza σ_ζ^2 y σ_η^2, respectivamente (parámetros del operador).

Cruce PCX (Deb y otros, 2002a). Dado un conjunto de padres $(C^1, \ldots, C^\mu)$, se calcula la media X^m. Después, para cada descendiente a crear, se selecciona aleatoriamente uno de los padres del conjunto (C^s) con igual probabilidad para todos. Se calcula el vector $D^s = C^s - X^m$. Para cada uno de los $(\mu - 1)$ padres restantes, se obtienen las distancias L^i perpendiculares a la recta D^s y se calcula su media $\overline{L}$. Entonces el descendiente se calcula como muestra la siguiente fórmula:

$$H = C^s + w_\zeta D^s + \sum_{i=1, i \neq s}^{\mu} w_\eta \overline{L} E^i \tag{1.23}$$

donde E^i son un conjunto de $(\mu-1)$ bases ortonormales que generan el subespacio perpendicular a D^s, y w_ζ y w_η son variables aleatorias que siguen una distribución normal con media cero y varianza σ_ζ^2 y σ_η^2, respectivamente (parámetros del operador).

Efectos de los Operadores de Cruce

El efecto de los operadores de cruce se puede estudiar desde dos puntos de vista: 1) a nivel de gen y 2) a nivel de cromosoma. A nivel de gen pueden considerarse distintos tipos de intervalos donde el cruce puede generar genes:

- Un intervalo de *explotación*, cuyos extremos son los genes a combinar. Se denomina así porque sus puntos contienen información común a los dos genes padres.
- Dos intervalos de *exploración*, que se definen a partir de cada extremo del intervalo de definición y el gen padre más cercano al mismo. No existe garantía de que los puntos pertenecientes a estos intervalos tengan propiedades comunes a los padres.
- Intervalos de *explotación relajada*. Son extensiones del intervalo de *explotación* hacia los intervalos de *exploración*.

Puede comprobarse que muchos de los cruces utilizan el intervalo de *explotación* para generar genes, lo cual, intuitivamente, parece adecuado. Sin embargo, estudios realizados sobre BLX-α (Eshelman y Schaffer, 1993) confirman la adecuación de utilizar $\alpha > 0{,}0$, lo que implica utilizar intervalos de *explotación relajada*. En ausencia de presión selectiva, valores de α menores que 0,5 producen la convergencia de los genes de la población hacia un valor próximo al centro de sus respectivos rangos, propiciando poca *diversidad* en la población y dando lugar a una posible convergencia prematura hacia lugares no óptimos. Sólo cuando $\alpha = 0{,}5$ se consigue un equilibrio entre las tendencias de convergencia (*explotación*) y divergencia (*exploración*), ya que la probabilidad de que un gen hijo caiga entre sus padres es igual a que lo haga fuera del intervalo formado por ellos. Además de BLX-α ($\alpha > 0{,}0$), otros tipos de cruces utilizan valores no pertenecientes al intervalo formado por los genes padres, como son el cruce lineal y los dos extendidos. En la Figura 1.22, se representa gráficamente el efecto de cada uno de los tipos de cruce enumerados y se supone, sin falta de generalidad, que $c_i^1 \leq c_i^2$ y que la adaptación de C^1 es mejor que la de C^2.

Al considerar el caso de los operadores heurístico y lineal BGA, se observa que ambos incluyen la bondad de los cromosomas padres para generar el hijo. El cruce heurístico produce hijos en el intervalo de exploración localizado en el mismo lado que el mejor padre. El cruce lineal BGA genera hijos en los intervalos de explotación o exploración cercanos al mejor padre, sin embargo, la probabilidad de visitar el intervalo de exploración es mayor.

A nivel de cromosoma, los efectos del cruce se pueden considerar geométricamente. Supuestos dos padres (X e Y), notaremos H_{xy} al hiperplano definido por los componentes de X e Y. Se puede comprobar que el cruce discreto y

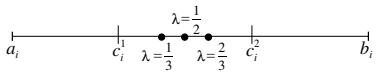
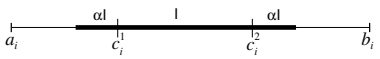
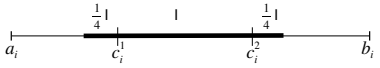
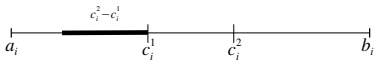
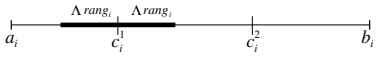
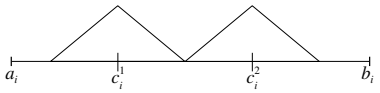
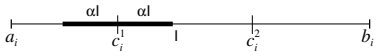
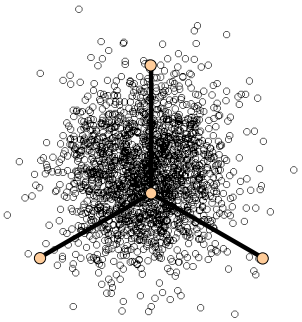
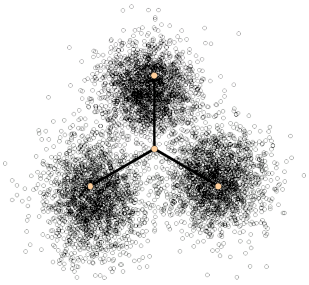
Tipo de Cruce	**Gráfica**
Aritmético	
BLX-α	
Lineal	
Discreto	
Extendido	
Heurístico	
Lineal BGA	
Difuso	
PBX-α	
UNDX	
PCX	

Figura 1.22: Operadores de cruce para AGCRs

el cruce simple generan esquinas de H_{xy}, el cruce intermedio extendido puede generar cualquier punto dentro de un hiperplano levemente mayor a H_{xy}, y el cruce lineal extendido y lineal generan un punto de la línea que une X e Y.

Con respecto a UNDX, es interesante ver que los descendientes se crean alrededor del centro de los padres, siendo mayor la probabilidad cuanto más cerca del centro se generan. Mientras que PBX-α y PCX generan el hijo alrededor de uno solo de los padres, aunque éste no tiene por qué ser el más apto.

Operadores de Mutación para Codificación Real

Supongamos que $C = (c_1, \ldots, c_i, \ldots, c_n)$ es un cromosoma y c_i es un elemento a mutar, a continuación se muestra el gen c_i' resultante de aplicar distintos operadores de mutación.

Mutación Aleatoria (Radcliffe, 1991). c_i' se toma de forma aleatoria (uniformemente) del dominio $[a_i, b_i]$.

Mutación No Uniforme (Michalewicz, 1992). Si se supone que este operador se aplica durante una generación t y que T es el máximo número de generaciones del AGCR, entonces

$$c_i' = \begin{cases} c_i + \Delta(t, b_i - c_i), & \text{Si } \beta = 0 \\ c_i - \Delta(t, c_i - a_i), & \text{Si } \beta = 1 \end{cases} \tag{1.24}$$

donde β es un número aleatorio que puede valer cero o uno, y

$$\Delta(t, y) = y\left(1 - r^{\left(1 - \frac{5}{T}\right)^b}\right), \tag{1.25}$$

donde r es un número aleatorio del intervalo $[0, 1]$ y b es un parámetro elegido por el usuario, que determina el grado de dependencia con el número de iteraciones. La función $\Delta(\cdot, \cdot)$ devuelve un valor en el rango $[0, y]$, de forma que la probabilidad de devolver un número cercano a cero crece conforme t es mayor. De esta forma, el intervalo de generación de genes es más pequeño conforme pasan las generaciones. Esta propiedad permite que este operador produzca una búsqueda uniforme del espacio de búsqueda cuando t es pequeño, y una búsqueda local en las etapas finales, favoreciendo el ajuste local.

Mutación Usando Desplazamiento (Davis, 1991b).

Cuando se optimiza una función de evaluación continua con máximos y mínimos locales, y en un determinado momento se obtiene un cromosoma situado en un buen máximo local, sería interesante generar cromosomas alrededor de éste para aproximarnos al punto cumbre de tal máximo. Para ello se puede deslizar el cromosoma en un valor que lo aumente o disminuya en una pequeña cantidad aleatoria. El máximo deslizamiento permitido está determinado por un parámetro definido por el usuario. Se han presentado distintos ejemplos de este

tipo de mutación, tales como: las mutaciones con *desplazamiento garantizado grande y pequeño* (Davis, 1989) y las mutaciones con *desplazamiento grande y pequeño* (Kelly y Davis, 1991). La diferencia entre estos operadores está en el valor máximo de desplazamiento permitido.

Mutación BGA (Mühlenbein y Schlierkamp-Voosen, 1993).

$$c_i' = c_i \pm rang_i \cdot \gamma, \tag{1.26}$$

donde $rang_i$ define el rango de mutación y normalmente se toma $0{,}1 \cdot (b_i - a_i)$, el signo $\pm$ representa $+$ o $-$ con probabilidad 0,5, y

$$\gamma = \sum_{k=0}^{15} \alpha_k 2^{-k}, \tag{1.27}$$

$\alpha_i \in \{0, 1\},\ i = 1, \ldots, 15$, se genera aleatoriamente con $p(\alpha_i = 1) = \frac{1}{16}$. Con este operador se generan valores del intervalo $[c_i - rang_i, c_i + rang_i]$. La probabilidad de generar vecinos al gen mutado es muy elevada. El mínimo acercamiento se realiza con una precisión de $rang_i \cdot 2^{-15}$.

Las siguientes mutaciones son generalizaciones de la mutación BGA. Se diferencian de ésta en la forma de calcular γ.

Mutación Modal Discreta (Voigt y Anheyer, 1994).

$$\gamma = \sum_{k=0}^{\pi} \alpha_i B_m^k, \tag{1.28}$$

con $\pi = \left\lfloor \frac{log(rang_{min})}{log(B_m)} \right\rfloor$. $B_m > 1$ es un parámetro denominado base de mutación (en el caso de la mutación anterior esta base es 2), y $rang_{min}$ es un límite inferior para el rango de mutación.

Mutación Modal Continua (Voigt y Anheyer, 1994).

$$\gamma = \sum_{k=0}^{\pi} \alpha_k \phi(B_m^k), \tag{1.29}$$

donde $\phi(z_k)$ es una distribución de probabilidad triangular con:

$$\frac{B_m^k - B_m^{k-1}}{2} \leq z_k \leq \frac{B_m^{k+1} - B_m^k}{2}. \tag{1.30}$$

Capítulo 2

Algoritmos Genéticos Locales: Caracterización

En el Capítulo 1, realizamos una introducción y breve repaso a las MHs más importantes presentadas en la literatura. Hemos prestado especial atención a dos tipos específicos de MHs:

- Los *métodos de BL*, especializados en aportar intensificación refinando soluciones, es decir, obteniendo otras de muy alta calidad consumiendo pocos recursos.
- Los *AGs*, métodos que imitan los procesos evolutivos de la naturaleza, aplicándolos a un conjunto de soluciones. Los AGs se han aplicado con éxito en problemas de búsqueda y optimización. Gran parte de este éxito se debe a su capacidad para explotar la información acumulada sobre un espacio de búsqueda, y así dirigir las siguientes búsquedas hacia los mejores subespacios.

Actualmente, el atractivo de los AGs como procedimientos de búsqueda ha motivado el diseño de AGs específicos que actúan como métodos de BL (es decir, pretenden proveer una alta intensificación). De hecho, se han presentado varias propuestas de AGs con este propósito (Herrera y Lozano, 2000a; Kazarlis y otros, 2001; Lozano y otros, 2004; Mutoh y otros, 2006; Noman y Iba, 2008; Potts y otros, 1994; Tsutsui y otros, 1997). En este capítulo, proponemos la denominación de *Algoritmos Genéticos Locales* (AGLs) para esta nueva familia de MHs.

Los AGLs son una novedosa categoría de MHs basadas en poblaciones para proveer intensificación. Éstos presentan ventajas sobre las técnicas clásicas de BL. En espacios de búsqueda complejos, muchos procedimientos de BL pierden la habilidad para seguir un camino hacia el óptimo. Esta dificultad es más evidente cuando el espacio de búsqueda contiene caminos muy estrechos de dirección arbitraria (conocidos como *crestas*, véase la Figura 2.1). Esto se debe a que estos métodos de BL intentan dar pasos a lo largo de direcciones ortogonales que no coinciden necesariamente con la dirección de la cresta. Sin embargo,

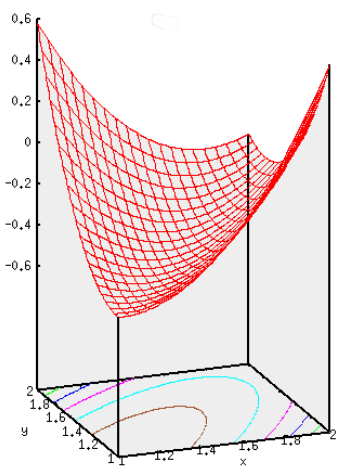


Figura 2.1: Ejemplo de cresta no alineada con los ejes de coordenadas

el estudio realizado por Kazarlis y otros (2001), apunta que los AGLs tienen capacidad para seguir crestas de dirección arbitraria sin importar su anchura, o incluso, discontinuidades.

El propósito de este capítulo es ofrecer una visión global de los AGLs que nos permita entender sus fundamentos, propiedades, comportamiento y la forma en que se utilizan para resolver problemas de forma efectiva. Para ello, comentaremos la evolución histórica del concepto AGL, el cual parte de la idea de aumentar la intensificación en los AGs, y acaba en la necesidad de crear una nueva categoría de MHs. Analizaremos las diferentes propuestas de AGLs presentadas en la literatura, con intención de extraer las propiedades comunes. Por último, los clasificaremos entre las MHs para encontrar sus similitudes y diferencias con otras familias de MHs.

El capítulo se estructura de la siguiente forma: en la Sección 2.1 destacamos las novedosas ideas de los AGLs frente a las de los AGs clásicos. En la Sección 2.2 estudiamos la evolución histórica del concepto de AGL realizando una revisión cronológica de los modelos propuestos en la literatura. En la Sección 2.3 describimos las propuestas de AGLs en profundidad. En la Sección 2.4 destacamos las propiedades comunes de los AGLs. En la Sección 2.5 determinamos el lugar que ocupan los AGLs dentro de una taxonomía ampliamente aceptada de las MHs. Por último, presentamos las conclusiones en la Sección 2.6.

2.1. Definición de AGLs: AGs para Intensificar

Tradicionalmente, los AGs se han aplicado de dos maneras diferentes:

- Como MHs completas y autónomas, es decir, diseñando cuidadosamente sus componentes para proveer, por ellas solas, diversificación e intensifi-

cación simultaneamente con la intención de obtener soluciones robustas y precisas al mismo tiempo.

- Como métodos de diversificación que se combinan con una técnica de BL encargada de las necesidades de intensificación. Un ejemplo de éstos, son los *Algoritmos Meméticos* (Krasnogor y Smith, 2005; Moscato, 1999).

Sin embargo, debido a la flexibilidad de la arquitectura de los AGs, se pueden diseñar modelos para proveer sólo intensificación. De hecho, varios estudios usan modelos AGs, cuidadosamente diseñados, con el único propósito de obtener soluciones precisas (Gang y otros, 2004; Herrera y Lozano, 2000a; Kazarlis y otros, 2001; Lozano y otros, 2004; Mutoh y otros, 2006; Noman y Iba, 2008; Potts y otros, 1994; Tsutsui y otros, 1997). A estos modelos de AG que, como las técnicas de BL, promueven la intensificación, los llamamos Algoritmos Genéticos Locales.

Es conveniente indicar que en la literatura podemos encontrar propuestas con nombres similares, que no son AGLs. Por ejemplo, el término *AGs Localizados* suele hacer referencia a los AGs Distribuidos (Tanese, 1987). También, algunos autores nombran a los Algoritmos Meméticos como *Búsqueda Local Genética* (en inglés *Genetic Local Search*) (Krasnogor y Smith, 2005; Moscato, 1999). Debemos remarcar que, en ninguno de estos modelos se pretende diseñar AGs especializados en intensificar.

2.2. Evolución Histórica del Concepto de AGL

En esta sección, describimos la evolución del concepto AGL. Referenciaremos las propuestas de AGLs que aparecen en la literatura, en orden cronológico, observando que aparece una cierta relación entre la percepción cada vez más clara de sus fronteras, y la forma en que se aplicaron.

Los primeros trabajos sobre AGLs nacieron por la dificultad de diseñar un único AG que proveyese un equilibrio apropiado entre diversificación e intensificación. Aparecieron entonces los trabajos de Potts y otros (1994), Tsutsui y otros (1997) y Herrera y Lozano (2000a). Sus soluciones consistían en la combinación de varios AGs con diferentes propósitos en un *marco distribuido* (Tanese, 1987), esto es, varias *subpoblaciones* que se procesan en paralelo por AGs independientes. La clave de los trabajos mencionados fue la de *especializar* los AGs de cada subpoblación para ofrecer diferentes grados de diversificación e intensificación (estos modelos se describen más profundamente en la Sección 2.3.1):

- Potts y otros (1994) usan cuatro subpoblaciones, nombradas como *especies I–IV*, las cuales aplican diferentes probabilidades de mutación. En este caso, especie IV intenta obtener una alta intensificación usando una probabilidad de mutación baja.

- Tsutsui y otros (1997) combinan dos subpoblaciones, una *exploradora* y otra *explotadora*. Mientras la primera explora el espacio de búsqueda, la

segunda se encarga de inspeccionar la región donde se localiza la mejor solución encontrada hasta el momento.

- Herrera y Lozano (2000a) proponen los *AGs Distribuidos Graduales con codificación real*. Estos modelos se basan en una topología hipercúbica con tres dimensiones donde la *cara frontal* se compone de cuatro subpoblaciones especializadas en diversificación y la *cara trasera*, de otras cuatro especializadas en intensificación.

Más adelante, aparecen los trabajos de Kazarlis y otros (2001), Lozano y otros (2004), Mutoh y otros (2006) y Gang y otros (2007), en los que se marca una diferencia importante en el uso de AGLs. En ellos se proponen modelos de AGs, cuyo propósito es optimizar los individuos de la población de un Algoritmo Evolutivo, esto es, aparecen AGLs que asumen el papel de los procedimientos clásicos de BL en el marco de los Algoritmos Meméticos (estos modelos se describen más profundamente en la Sección 2.3.2):

- Kazarlis y otros (2001) proponen el uso de un *Micro-AG* (μAG) (AG con una pequeña población y corta evolución) para evolucionar perturbaciones de una solución específica.

- Lozano y otros (2004) proponen un *Operador de Ascensión de Colinas basado en Cruce* que mantiene un par de padres (uno de ellos es la solución a refinar) y aplica repetidamente el operador de cruce sobre ellos.

- Mutoh y otros (2006) proponen *la aplicación repetida del operador de cruce* sobre los mejores hijos obtenidos.

- Gang y otros (2007) tratan el problema del viajante de comercio. La optimización de las soluciones de la población la lleva a cabo un AGL que refina tramos parciales, elegidos aleatoriamente, de la solución dada.

- Finalmente, Noman y Iba (2008) proponen una variación del modelo de Lozano y otros (2004) para mejorar el rendimiento de la *Evolución Diferencial* (Storn y Price, 1997; Price y otros, 2005).

2.3. Descripción de las Propuestas de AGLs

En esta sección, describimos los detalles de las diferentes propuestas de AGLs. En la Sección 2.3.1, nos centramos en las propuestas de los primeros estudios, en los que los AGLs se aplican sobre ciertas subpoblaciones de AGs distribuidos. En la Sección 2.3.2, describiremos los modelos de AGL que se han utilizado como BL dentro de un Algoritmo Memético. En la Sección 2.3.3, mencionamos otros trabajos que, aunque no se consideran AGLs, están relacionados con ellos. Por último, en la Sección 2.3.4, realizamos un resumen global de las características específicas de las propuestas de AGLs.

2.3.1. AGLs en AG Distribuidos

Los AGs distribuidos mantienen en paralelo varias *subpoblaciones* independientes que se procesan por AGs (Tanese, 1987). Periódicamente, un *mecanismo de migración* hace que uno o varios cromosomas se trasladen a otra subpoblación. Haciendo distinción entre las subpoblaciones, aplicando AGs con diferentes configuraciones, obtenemos *AGs Distribuidos Heterogéneos*. Estos algoritmos son una manera prometedora de introducir un correcto equilibrio entre intensificación y diversificación, para evitar la convergencia prematura y alcanzar soluciones precisas y robustas.

En las siguientes secciones, describimos tres modelos de AGs Distribuidos Heterogéneos que asignan diferentes papeles de intensificación o diversificación a cada subpoblación. En ellos, las subpoblaciones especializadas en intensificación son AGLs, cuya misión es refinar las soluciones provenientes de subpoblaciones diversificadoras.

Por último, comentar que estos modelos de AGLs no son autocontenidos, esto es, en muchos casos, se considera que tienen un carácter intensificador, precisamente por la presencia de AGs diversificadores.

AG Basado en Migración y Selección Artificial

Potts y otros (1994) proponen un AG Distribuido Heterogéneo que usa cuatro subpoblaciones, denotadas como *especies I–IV*, las cuales suplen diferentes grados de intensificación o diversificación mediante la aplicación de diferentes probabilidades de mutación:

- Especie II es una subpoblación usada para diversificación. Por este propósito, utiliza una probabilidad alta de mutación ($p_m = 0{,}05$).
- Especie IV es una subpoblación usada para intensificación, AGL. Para ello, utiliza una probabilidad de mutación baja ($p_m = 0{,}003$).
- Especie III es una subpoblación intermedia dedicada tanto a intensificar como a diversificar. Su probabilidad de mutación es media ($p_m = 0{,}005$).

El modelo utiliza a especie I para preservar la mejor solución que aparece en las otras especies. Para ello, selecciona el mejor individuo de entre las especies II-IV y la introduce en especie I siempre que sean mejores que los elementos que ya contiene. En generaciones predeterminadas, los elementos en especie I se reintroducen en especie IV, reemplazando todos sus elementos.

Estudios experimentales mostraron que este modelo superaba consistentemente a AGs simples y reducía el problema de la *convergencia prematura*.

AG con una Población Exploradora y otra Explotadora

Tsutsui y otros (1997) proponen un AG con dos poblaciones con diferentes misiones: una pretende explorar el espacio de búsqueda, mientras que la otra

es un AGL que busca en la vecindad de la mejor solución encontrada hasta el momento. Ambas poblaciones se procesan por AGs generacionales, sin embargo, el AGL utiliza una *mutación de grano fino* y una población con la mitad de individuos que la exploradora. Estas propiedades hacen que el comportamiento del AGL sea altamente intensificador sobre la mejor solución encontrada hasta el momento.

La propuesta exhibió un rendimiento significativamente superior al de AGs estándares en dos problemas complejos altamente multimodales.

AGs Distribuidos Graduales con Codificación Real

Herrera y Lozano (2000a) aprovecharon la disponibilidad de operadores de cruce para AGs con codificación real (AGCRs) (Herrera y otros, 1998, 2003) que generan diferentes grados de intensificación o diversificación, para diseñar AGs Distribuidos Heterogéneos basados en estos operadores. Herrera y Lozano (2000a) proponen los *AGs Distribuidos Graduales con codificación real*, que aplican diferentes operadores de cruce a cada subpoblación. Estos operadores se diferencian de acuerdo a las propiedades asociadas de intensificación y diversificación, y al grado de las mismas. El efecto obtenido es una *multiresolución paralela* con respecto a la acción de los operadores de cruce. Además, las subpoblaciones se conectan adecuadamente para explotar esta multiresolución de una forma *gradual*.

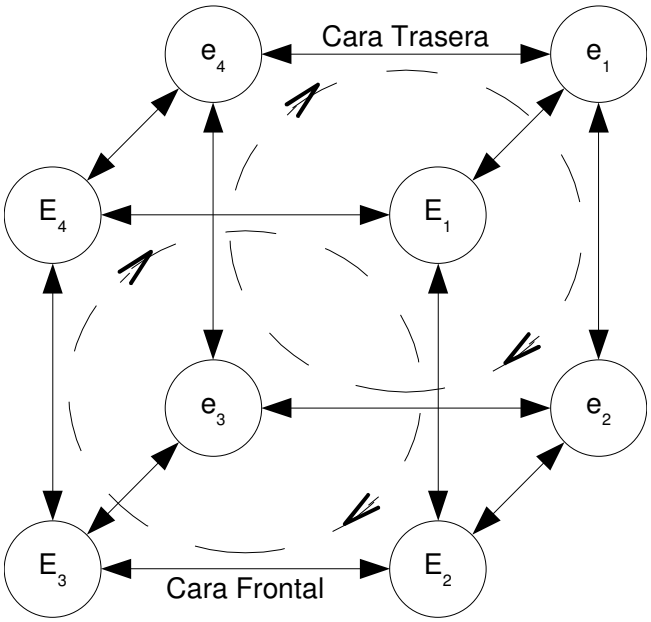


Figura 2.2: Estructura de un AG Distribuido Gradual con codificación real

Los AGs Distribuidos Graduales con codificación real se basan en una topología hipercúbica con tres dimensiones (ver Figura 2.2). En ella, hay dos caras importantes, bien diferenciadas:

- La *cara frontal* se dedica a diversificar. Se compone de cuatro subpoblaciones $(E_1, \ldots, E_4)$ en las cuales se aplican operadores diversificadores. El grado de diversificación se incrementa en el sentido de las agujas del reloj, empezando en el menor (E_1) y acabando en el mayor (E_4).
- La *cara trasera* es para intensificación. Está formada por las subpoblaciones $e_1, \ldots, e_4$, que se someten a operadores de cruce intensificadores. El grado de intensificación se incrementa en el sentido de las agujas del reloj, empezando en el menor (e_1) y acabando en el mayor (e_4). Las poblaciones $e_1, \ldots, e_4$ son AGLs que ofrecen diferentes niveles de intensificación.

La conectividad de estos modelos permite un *refinamiento gradual* cuando se producen migraciones desde una subpoblación diversificadora hacia una intensificadora, esto es, desde E_i a e_i, o entre dos subpoblaciones intensificadoras de menor grado a mayor, esto es, desde e_i a e_{i+1}.

Los resultados experimentales mostraron que los AGs Distribuidos Graduales con codificación real mejoraban consistentemente a otros AGCRs, AGCRs Distribuidos Homogéneos y otros Algoritmos Evolutivos presentados en la literatura.

2.3.2. AGLs en Algoritmos Meméticos

La principal idea de los Algoritmos Meméticos (Moscato, 1999) es la de combinar, sobre una población de agentes, procesos de cooperación y competición, presentes en los Algoritmos Evolutivos, con procesos de mejora individual. La mayoría de ellos son Algoritmos Evolutivos en los que se aplica un proceso de BL para refinar los individuos de la población (Krasnogor y Smith, 2005). Se suele afirmar que el buen rendimiento de los Algoritmos Meméticos se debe al equilibrio entre las habilidades diversificadoras del Algoritmo Evolutivo y las habilidades intensificadoras de la técnica de BL usada.

En las siguientes secciones, describimos los modelos de Algoritmos Meméticos caracterizados por reemplazar el método de BL por un AGL.

AGLs Basados en μAGs

Kazarlis y otros (2001) usan un μAG (AG con una población pequeña y corta evolución) como AGL dentro de un Algoritmo Memético. Su misión es la de refinar las soluciones de la población del algoritmo. Para ello, evoluciona una población de *perturbaciones*, cuyos valores de aptitud dependen de la solución dada por el Algoritmo Memético.

Sus principales propiedades son:

- Utiliza un tamaño de población pequeño (5 individuos). De esta forma obtiene unos niveles de presión selectiva muy altos que le permiten alcanzar soluciones precisas.
- Usa un mecanismo de selección de padres guiado por la aptitud de los individuos, la ruleta, lo cual aumenta la presión selectiva.

- El espacio de perturbaciones se define de tal forma que el μAG explora una pequeña región centrada en la solución dada. De esta forma, ofrece mejoras locales.
- Sigue el modelo *generacional* con elitismo a dos niveles, lo cual aumenta el carácter intensificador:
  - *elitismo interno*: la mejor perturbación de la generación i de la invocación actual del μAG, se añade a la población de la siguiente generación $(i+1)$. Se trata del conocido mecanismo de elitismo definido para los AGs generacionales.
  - *elitismo externo*: la mejor perturbación de la invocación k del μAG, se añade a la población inicial de la siguiente invocación $(k+1)$. Este esquema compensa la corta evolución del μAG y sigue el siguiente razonamiento: En ciertas ocasiones, el *gradiente* que lleva a mejores soluciones en el espacio de búsqueda es relativamente constante, lo que implica que la dirección óptima a mejores soluciones permanece igual entre invocaciones sucesivas del μAG.

Los autores argumentan que μAG, a diferencia de los métodos de BL comparados, es capaz de seguir *crestas* de dirección arbitraria, a pesar de su dirección, anchura, o incluso, discontinuidad. Además, un estudio empírico con cinco problemas de optimización con restricciones duras reveló que el Algoritmo Memético basado en μAG mejoraba en precisión, fiabilidad y robustez a otros doce Algoritmos Evolutivos con diferentes técnicas de BL. Por otro lado, otros investigadores también han utilizado modelos de μAGs como AGLs, obteniendo buenos resultados en otros estudios:

- Weicai y otros (2004) proponen un *AG multiagente* que hace uso de μAG.
- Meloni y otros (2003) insertan μAG en un *Algoritmo Evolutivo Multiobjetivo* para una clase de problemas de planificación en entornos de fabricación.
- Papadakis y Theocharis (2002) usan μAG dentro de un Algoritmo Memético para crear *modelos difusos TSK*.

AGL con Codificación Real: OACC

Lozano y otros (2004) proponen un Algoritmo Memético con codificación real que utiliza un *Operador de Ascensión de Colinas basado en Cruce* (OACC) como AGL. Su misión es obtener los mejores niveles de precisión que lleven a la población hacia las regiones de búsqueda más prometedoras, realizando un refinamiento efectivo sobre ellas. Además, se propone un mecanismo adaptativo para determinar la probabilidad con que las soluciones se refinan con OACC.

OACC es realmente un AGCR estacionario que mantiene un par de padres y aplica el cruce repetidamente sobre este par, hasta alcanzar un número predeterminado de descendientes, n_{off}. Entonces, el mejor descendiente se selecciona y reemplaza al peor padre, sólo si es mejor. El proceso itera durante n_{it} generaciones y devuelve, finalmente, los dos padres actuales.

El OACC propuesto se puede concebir como un *micro selecto-recombinador AGL con codificación real* que emplea el menor tamaño de población necesario para aplicar el operador de cruce, esto es, dos cromosomas. Al inicio, estos dos cromosomas son la solución que se quiere refinar y la mejor alcanzada hasta el momento. De esta forma, se incrementan las propiedades intensificadoras de OACC.

Aunque OACC puede utilizar cualquier operador de cruce, los autores usaron un *operador autoadaptativo* que genera descendientes de acuerdo a la distribución actual de los padres. Si los padres se localizan cercanos unos a otros, los descendientes se generarán densamente a su alrededor. Por otro lado, si los padres se localizan distanciados unos de otros, entonces los descendientes se encontraran distribuidos de forma distanciada a los padres y entre ellos.

Resultados experimentales mostraron que, para un amplio rango de problemas, el Algoritmo Memético con OACC mejoraba consistentemente a otros Algoritmos Meméticos presentados en la literatura.

Finalmente, en la literatura podemos encontrar algunas variantes de OACC (Dietzfelbinger y otros, 2003; Elliott y otros, 2004).

Aplicación Repetida del Operador de Cruce

Mutoh y otros (2006) proponen dos modelos de AGLs muy similares a OACC, con la diferencia de que en cada paso, todos los padres se reemplazan por los mejores hijos. Mutoh y otros (2006) aplican estos AGLs con el operador de cruce SPX (Tsutsui y otros, 1999), el cual genera un número fijo de hijos a partir de un conjunto de padres y es útil a la hora de refinar soluciones. Mientras que el primer modelo realiza siempre un número establecido de iteraciones, como OACC, el segundo utiliza unas reglas simples para adaptar el número de iteraciones según la mejora producida en la invocación anterior:

- Si la mejora es inferior a la unidad, se decrementa en un paso el número de iteraciones para la próxima invocación.
- Si la mejora es mayor a la unidad, pero inferior a un umbral dado, el número de iteraciones será el mismo que el actual.
- Si la mejora es mayor al umbral dado, el número de iteraciones se incrementará en una en la próxima invocación.

Los experimentos mostraron que el Algoritmo Memético con el segundo modelo de AGL propuesto era capaz de encontrar la solución óptima de un conjunto de problemas, entre el 30 y 50 % más rápido que su versión AG sin aplicación repetida del cruce.

AGLs que Refinan Tramos de Circuitos Hamiltonianos

Gang y otros (2007) proponen un método de BL basado en recombinación genética para refinar las soluciones de un Algoritmo Memético al tratar el *problema del viajante de comercio* (en inglés, *travelling salesman problem* o TSP),

encontrar el ciclo hamiltoniano de menor coste de un grafo completamente conexo. La idea de este método es utilizar un AG para optimizar subtramos concretos de las soluciones del Algoritmo Memético. Es por tanto un AGL. Para ello, se elige un conjunto de ciudades contiguas de la solución y se aplica el AGL para encontrar una reordenación de esas ciudades, manteniendo intactas la inicial y la final, que tenga un coste menor. Al terminar la ejecución, si se ha encontrado una reordenación mejor, se actualiza el tramo de la solución inicial.

Experimentos sobre dos problemas del viajante de comercio mostraban que los mejores resultados se obtenían cuando el AGL optimizaba tramos de longitud la mitad que las soluciones completas. Sin embargo, los autores comentan que optimizar tramos de menor longitud puede ser más beneficioso para problemas de mayor tamaño, debido al coste computacional que tiene optimizar tramos largos (se convierte en un problema tan complejo como el propio viajante de comercio).

Un AGL para Mejorar la Evolución Diferencial

Noman y Iba (2008) proponen un AGL para mejorar el rendimiento de la *Evolución Diferencial* (Price y otros, 2005; Storn y Price, 1997), la cual, como los AGs, es una MH basada en poblaciones.

El AGL propuesto por Noman y Iba (2008) es una variante del OACC de Lozano y otros (2004) que adaptativamente determina la intensificación a ofrecer de acuerdo a la retroalimentación que recibe de la Evolución Diferencial. Esta intensificación viene dada por el número de iteraciones que se ejecuta el OACC y que se conoce comúnmente como *longitud de refinamiento*. La motivación de este modelo es la dificultad, o incluso imposibilidad, de encontrar un valor de longitud de ejecución fijo, o mecanismo dinámico determinista, que sea adecuado para algunos problemas. Sus características son las siguientes:

- En cada iteración, aplica el operador de cruce sobre la solución a optimizar y otras n_p elegidas aleatoriamente de la población de la Evolución Diferencial para producir un único descendiente.
- Si el descendiente es mejor que la solución a optimizar, la reemplaza. En otro caso, da por finalizada la optimización y devuelve el control a la Evolución Diferencial. De esta forma, el AGL adapta la longitud de su ejecución, realizando ejecuciones más intensas cuando es probable mejorar la solución a optimizar, y menos intensas cuando la probabilidad es menor.
- Utiliza el operador de cruce SPX (Tsutsui y otros, 1999), el cual presenta varias ventajas:
  - no depende del sistema de coordenadas,
  - el centro de los padres y de los hijos es el mismo,
  - puede preservar la matriz de covarianza de la población con un ajuste de parámetros adecuado, y
  - es útil para realizar refinamiento.

Hemos de comentar, que en la hibridación propuesta por Noman y Iba (2008), el AGL sólo optimiza el mejor elemento de la población actual. De esta forma, se pretende reducir el consumo computacional de optimizar soluciones que probablemente no lleguen a mejorar la mejor solución alcanzada hasta el momento.

Un estudio experimental mostró que el nuevo AGL mejoraba consistentemente la velocidad de convergencia del modelo clásico de Evolución Diferencial. El rendimiento global del AGL propuesto era mejor que el de otros optimizadores locales basados en cruce. Por último, el rendimiento de la propuesta de Evolución Diferencial con el AGL era superior o al menos competitivo con otros Algoritmos Meméticos propuestos en la literatura.

2.3.3. Otros Trabajos Relacionados

Finalmente, podríamos mencionar otros tres trabajos que, aunque no los consideramos AGLs, están relacionados con la idea subyacente.

- Reijmers y otros (1999) proponen un AG para la generación de árboles filogenéticos, el cual busca en el espacio de correcciones que hay que aplicar a una solución específica para llegar a la óptima. El espacio de correcciones se define para alcanzar, desde la solución inicial, prácticamente cualquier otra solución del problema en muy pocas iteraciones. Precisamente esta propiedad hace que el modelo realice una exploración global del espacio de búsqueda y se preocupe menos por proveer sólo intensificación.
- Kwon y Kim (2003) proponen un método que iterativamente reduce la región de búsqueda de un AG por un factor dado (α) y la centra en la mejor solución encontrada hasta el momento. Entonces, se genera una nueva población de soluciones aleatorias en la nueva región y se continúa el proceso. Este modelo no se ha considerado un AGL debido a que la intensificación no viene dada por el AG, sino por el mecanismo externo que restringe su espacio de búsqueda iterativamente.
- Thierens (2004) propone un modelo de BL iterativa (Lourenço y otros, 2003) donde la técnica de BL se guía mediante una población de soluciones. La principal característica que nos ha llevado a no considerar el método interno de BL como AGL, es que no utiliza un operador de cruce como tal. Su operador de generación de soluciones realiza un recorrido sobre el vecindario de la solución actual, mediante movimientos alineados con los ejes de coordenadas (como las técnicas clásicas de BL), utilizando otra solución de su población como máscara, esto es, prohibiendo ciertos movimientos.

2.3.4. Comparación Global de las Decisiones de Diseño de los Modelos de AGL

La Tabla 2.1 reúne las propiedades, que promueven intensificación, de los modelos de AGLs presentados. En las filas aparecen cada uno de los modelos de AGL, mientras que en las columnas aparecen las siguientes decisiones de diseño:

Tabla 2.1: Propiedades de los modelos de AGL, diseñadas para promover intensificación

	Representación	**Tamaño de P**	**Op. Mutación**	**Op. Cruce**	**Sel. Aptitud**
Especie IV	-	-	Probabilidad baja	-	-
P. Explotadora	-	$\|P\|/2$	Mut. de grano fino	-	-
AGLs en AGs Dist. Graduales	-	-	-	Intensificadores	-
μAG	Perturbaciones de una solución	5	-	-	Ruleta
AGL para TSP	Tramos parciales de una solución	-	-	-	-
OACC	-	2	-	Autoadaptativo	-
OACC para Evolución Diferencial	-	n_p	-	Útil refinando	-

- *Representación*: Algunos AGLs usan un modelo de representación diferente al de la MH que lo aplica, promoviendo de esta forma intensificación. En este caso, se muestra el modelo de representación del AGL. Cuando el AGL y la MH utilizan el mismo modelo de representación de soluciones, aparecerá el carácter '-'.
- *Tamaño de la población*: El uso de un tamaño de población reducido es una técnica seguida por varios AGLs para promover intensificación. Para los AGLs que aplican esta técnica, la celda indica el tamaño de población que utilizan. Para los AGLs que no siguen esta técnica, se muestra el carácter '-'.
- *Operador de mutación*: Existen AGLs que usan operadores de mutación, o valores específicos de su probabilidad de aplicación, para intensificación. Si éste es el caso, en la celda aparecerá la propiedad correspondiente. El carácter '-' indicará que el AGL no siguen esta técnica para promover intensificación.
- *Operador de cruce*: Algunos AGLs aplican operadores de cruce especializados en promover intensificación. Para estos AGLs, la celda muestra la propiedad correspondiente. Si no se hace uso de estos operadores especializados, la celda muestra el carácter '-'.
- *Selección por aptitud*: Algunos modelos promueven intensificación resaltando la selección de individuos con altos valores de aptitud, es decir, aumentando la presión selectiva sobre los mejores individuos. Si éste es el caso, la celda muestra el mecanismo de selección. En otro caso, la celda muestra el carácter '-'.

Podemos ver que se han seguido caminos muy diversos para acentuar la intensificación en los AGs. Esto nos indica que el campo merece más estudios, y así obtener AGLs más eficaces para diferentes problemas.

2.4. Propiedades Generales de los AGLs

En esta sección, destacamos las propiedades comunes a todas las propuestas revisadas y que, a nuestra opinión, cualquier AGL debiera presentar. Éstas son:

- *Son algoritmos basados en poblaciones*: al igual que los AG clásicos, los AGLs mantienen un conjunto de soluciones que evoluciona durante la ejecución. Esta propiedad marca una diferencia importante con los métodos clásicos de BL.
- *Utilizan operadores genéticos*: realizan un proceso de búsqueda guiado por la aplicación de operadores de cruce y, en su caso, también de mutación. Esta propiedad, derivada de la anterior, también es importante para marcar diferencias con las técnicas clásicas de BL. Mientras las técnicas clásicas de BL, sólo poseen un operador de generación de nuevas soluciones, el cual comparte algunas propiedades con el operador de mutación de los AGs, los AGLs no sólo disponen de dos operadores, sino que suelen

darle más importancia a la combinación de información de más de una solución (cruce), que a la generación de nuevas soluciones a partir de la información de una sola de ellas.

- *Están diseñados para intensificar*: sus componentes y parámetros deben estar diseñados para ofrecer intensificación. Esto no implica que todos los componentes sigan esta regla, lo cual podría llevar a una convergencia prematura a soluciones pobres. En un extremo, la mayoría de los componentes pueden ser los típicos de un AG clásico (no diversificadores puros) excepto uno con una clara intención intensificadora.

  Además, debiera quedar claro que, el hecho de que los AGLs incorporen componentes, típicamente diversificadores (por ejemplo, el uso de una población de soluciones) no les impide alcanzar una intensificación fructífera, sino que les puede ayudar a mejorar la capacidad intensificadora. Como indican varios investigadores, intensificación y diversificación no son factores opuestos (Glover y Laguna, 1997). De hecho, el uso de una población de soluciones, permite a los AGLs usar más fuentes de información para diseñar estrategias de intensificación que las usadas por métodos clásicos de BL (los métodos clásicos trabajan con una sola solución). Además, la población es la forma natural de un AGL para implementar mecanismos de memoria similares a los que utilizan otros algoritmos como *Búsqueda Tabú* (Glover y Laguna, 1997), los cuales pueden usarse para aumentar aún más la intensificación.

- *Deben combinarse con una MH diversificadora*: su uso aislado no dará, en general, soluciones fiables. Esto se debe al desequilibrio que presentan entre intensificación y diversificación. Por tanto, deben, y así se ha hecho en todas las propuestas, combinarse con otras MHs que se encarguen de realizar una exploración global del espacio de búsqueda.

2.5. Los AGLs entre las MHs

En esta sección, estudiamos el papel que ocupan los AGLs en el campo de las MHs. Para ello, adoptamos una taxonomía de MHs ampliamente aceptada, y determinamos el lugar donde se sitúan los AGLs.

Hay varias maneras de clasificar y describir las MHs. Dependiendo de la característica que seleccionemos, varias clasificaciones serían posibles. Cada una de ellas es el resultado de un punto de vista específico (Blum y Roli, 2003). Una característica comúnmente usada es el número de soluciones que la MH maneja (Blum y Roli, 2003):

- *Métodos basados en trayectorias*: Este grupo se compone de MHs que trabajan sobre una sola solución. Todos ellos comparten la propiedad de describir una *trayectoria* en el espacio de búsqueda, debido a que el proceso se compone de *pasos* desde la *solución actual* hacia una nueva solución. Varios ejemplos son los procedimientos de BL (Blum y Roli, 2003; Aarts y van Laarhoven, 1992; Davis, 1991a), Búsqueda Tabú (Glover y Laguna, 1997), Enfriamiento Simulado (Kirkpatrick y otros, 1983; Laarhoven y

Aarts, 1987), y algunas MHs que internamente hacen uso de métodos de BL como: BL con Multiarranque Aleatorio (Boender y otros, 1982; Rinnoy Kan y Timmer, 1989), BL Iterativa (Lourenço y otros, 2003), y Búsqueda de Vecindario Variable (Hansen y Mladenović, 2002; Mladenovic y Hansen, 1997). En estas últimas MHs, se alternan un proceso de generación de una solución nueva y una técnica de BL que optimiza la nueva solución. Considerando la generación de una nueva solución como un *paso* desde la última solución optimizada, podemos ver que el método completo sigue el símil de trayectoria.

- *MHs basadas en poblaciones*: Éstas realizan procesos de búsqueda que describen la *evolución de un conjunto de puntos* en el espacio de búsqueda. Varios ejemplos son los AGs (Goldberg, 1989b; Holland, 1975), las Estrategias de Evolución (Schwefel, 1995), la Programación Evolutiva (Fogel, 1992, 1995), los Algoritmos de Estimación de Distribuciones (Larrañaga y Lozano, 2001), los Algoritmos de Optimización basados en Nubes de Partículas (Kennedy y Eberhart, 2001), y la Evolución Diferencial (Storn y Price, 1997; Price y otros, 2005).

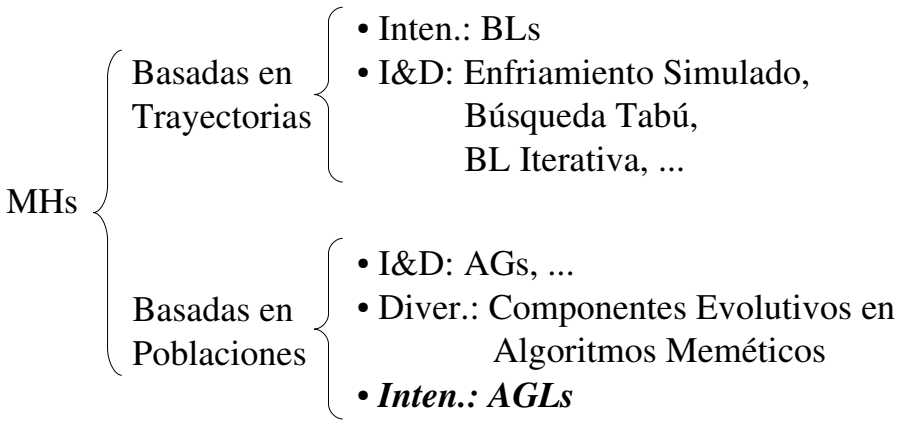


Figura 2.3: Taxonomía de las MHs y algunos ejemplos de su aplicación para promover diversificación (Diver.), Intensificación (Inten.), o ambas (I&D)

La Figura 2.3 muestra la taxonomía de MHs, según el número de soluciones que utilizan, y algunos ejemplos, normalmente aplicados para promover diversificación, intensificación o ambas. En ella se muestra que:

- Los métodos basados en trayectorias se aplican, bien para ofrecer intensificación (por ejemplo, las técnicas de BL se encargan a menudo de refinar las soluciones encontradas por una MH basada en poblaciones), o para promover ambas, intensificación y diversificación (por ejemplo, Búsqueda Tabú pretende localizar regiones prometedoras del espacio de búsqueda y obtener las mejores soluciones dentro de esas regiones a la vez).

- Por otro lado, los métodos basados en poblaciones se aplican para promover ambas, intensificación y diversificación (por ejemplo, los estudios

iniciales en AGs pretenden encontrar un equilibrio apropiado entre estas características), o para favorecer la diversificación (por ejemplo, en los Algoritmos Meméticos, la componente evolutiva toma el cargo de diversificar, y deja las necesidades de intensificar a la técnica de BL) (Moscato, 1999; Krasnogor y Smith, 2005).

Nosotros entendemos que estas formas de aplicar las diferentes MHs están altamente influenciadas por su aparición cronológica:

- Los primeros métodos basados en trayectorias eran propuestas altamente intensificadoras. Los estudios anteriores a los ochenta proponían métodos de BL (Dunham y otros, 1963; Lin y Kernighan, 1973). Después, los investigadores se dieron cuenta de la importancia de la diversificación y propusieron nuevos métodos basados en trayectorias que promoviesen tanto intensificación como diversificación. Entonces aparecieron los métodos con multiarranque (Boender y otros, 1982), Enfriamiento Simulado (Kirkpatrick y otros, 1983), y Búsqueda Tabú (Glover, 1986; Glover y Laguna, 1997).
- Por otro lado, los estudios en MHs basadas en poblaciones comenzaron con la importancia de la diversificación en mente. Desde sus inicios, los AGs incluyen un operador de mutación (Holland, 1975), y el concepto de *diversidad en la población* también apareció temprano (De Jong, 1975). Por tanto, las propuestas iniciales ya pretendían proveer ambas, intensificación y diversificación. Más tarde, aparecieron modelos híbridos donde las necesidades de diversificación se asignaban a las componentes basadas en poblaciones y las de intensificación a procedimientos de BL (Moscato, 1999; Krasnogor y Smith, 2005).

En los últimos años, varios investigadores han percibido que las MHs basadas en poblaciones también pueden aplicarse para ofrecer sólo intensificación, y de hecho, ofrecen algunas ventajas sobre las técnicas clásicas de BL. Concretamente, han aparecido varios algoritmos AGs con este propósito, y se han revisado en las secciones anteriores. En este capítulo, a estos algoritmos se les ha llamado AGLs. Los AGLs son una *novedosa categoría de MHs basadas en poblaciones para la intensificación* como se muestra resaltado en la Figura 2.3.

2.6. Conclusiones

En este capítulo, hemos introducido los AGLs, una nueva categoría de MHs especializados en intensificación. Con la intención de ofrecer una visión global y un mejor entendimiento de sus fundamentos, hemos revisado los modelos presentados en la literatura observando:

- *Evolución histórica*: Los AGLs nacen de la dificultad de proveer un equilibrio apropiado entre intensificación y diversificación en un solo AG. Las propuestas de la literatura especializan los AGs, obteniendo AGLs para intensificación. Finalmente, estos modelos adquieren una entidad propia, merecedora de un nombre y clasificación en el campo de las MHs.

- *Entorno de aplicación*: Influenciados por la evolución histórica, los AGLs aparecen primero en modelos distribuidos. Posteriormente, son componentes autocontenidos de Algoritmos Meméticos.
- *Decisiones de diseño*: Los investigadores han seguido caminos muy diversos con la intención de aumentar la intensificación en los AGLs.
- *Propiedades comunes*: Son MHs basadas en poblaciones, que, aplicando operadores de cruce, suplen intensificación a otra MH.

Finalmente, hemos clasificado los AGLs en una taxonomía, ampliamente aceptada, de las MHs. Esto nos ha permitido observar que, la forma de aplicar las MHs basadas en poblaciones se ha visto altamente influenciada por el orden cronológico de aparición de las diferentes MHs. Con la importancia de la diversificación en mente, las MHs basadas en poblaciones se habían utilizado para diversificar, o bien, diversificar e intensificar a la vez. Por tanto, los AGLs constituyen *una novedosa categoría de MHs basadas en poblaciones especializadas en intensificación.*

Capítulo 3

Algoritmo Genético Local Binario

Los métodos de BL son procedimientos de optimización capaces de, dada una solución inicial, alcanzar un óptimo local con gran precisión y bajo consumo de recursos, esto es, muestran una clara tendencia intensificadora. Las técnicas de BL son un componente fundamental de muchas MHs basadas en BL (ver Sección 1.5) que son el *estado del arte* para muchos problemas de optimización (Blum y Roli, 2003; Glover y Kochenberger, 2003; Ribeiro y Hansen, 2002; Siarry y Michalewicz, 2008; Voß y otros, 1999). Por esta razón, en la actualidad, los métodos de BL despiertan el interés de la comunidad científica, cuya intención es la de diseñar procedimientos de BL más efectivos que puedan aplicarse para la construcción de MHs basadas en BL más potentes.

Los AGLs, revisados en el Capítulo 2, son AGs especializados en intensificación. Además, estos métodos poseen algunas ventajas sobre las técnicas clásicas de BL. Según Kazarlis y otros (2001), la mayoría de los procedimientos de BL no son capaces de seguir el camino hacia el óptimo cuando éste presenta *crestas*, cuya dirección no es ortogonal al espacio de búsqueda. Sin embargo, los AGLs sí son capaces de seguirlas (Kazarlis y otros, 2001). Por ello, el estudio de AGLs y de su posible aplicación en MHs basadas en BL para obtener mejores resultados, se convierte en un campo de estudio prometedor. En concreto, los AGLs pueden asumir fácilmente el papel de los métodos de BL en las MHs basadas en BL, obteniendo, de esta forma, una nueva clase de MHs que podemos denominar MHs basadas en AGLs.

El propósito de este capítulo es doble:

1. Diseñar un AGL que, usando conceptos de los AGs, realice un proceso de refinamiento sobre soluciones iniciales, que mejore al realizado por una técnica clásica de BL. Llamaremos a este nuevo modelo de AGL, *AGL Binario* (AGLB), porque se ha diseñado para tratar problemas con esta codificación. AGLB es un AG *estacionario* que emplea un *método de reemplazo por agrupamiento* (en inglés, *crowding method*) para favorecer la formación de nichos con soluciones de alta calidad en la población.

Después, AGLB realiza un proceso de BL orientado hacia los nichos más cercanos a la solución a refinar.

2. Estudiar las ventajas que introduce el uso de AGLB en las MHs basadas en BL. Para ello, realizaremos un estudio empírico comparando los resultados de varias MHs basadas en BL que usan AGLB, con sus correspondientes versiones clásicas, esto es, usando técnicas clásicas de BL. Nos interesaremos por tres MHs basadas en BL que nunca se han considerado para la aplicación de AGLs: BL con Multiarranque Aleatorio, BL Iterativa y Búsqueda de Vecindario Variable. Además, estas MHs basadas en BL introducen una coordinación gradual y controlada con el método de refinamiento, que nos permite analizar aspectos muy concretos para el diseño de MHs basadas en AGLs, y formular conclusiones sencillas. Para contrastar los resultados, utilizaremos cuatro métodos de BL propuestos en la literatura que, a nuestro conocimiento, son un conjunto actual y representativo para el tratamiento de problemas binarios. Los resultados del estudio nos indicarán que, para un amplio rango de problemas, AGLB realiza un proceso de refinamiento eficiente y eficaz que permite obtener MHs basadas en AGLs más efectivas.

El capítulo está organizado de la siguiente forma. En la Sección 3.1, describimos AGLB. En la Sección 3.2, establecemos el marco experimental, esto es, los problemas de prueba usados y las condiciones de ejecución de los algoritmos, para nuestro estudio experimental. En la Sección 3.3, analizamos el comportamiento de AGLB al utilizarse como método de refinamiento de BL con Multiarranque Aleatorio. En la Sección 3.4, estudiamos el rendimiento de AGLB dentro de una MH basada en BL que introduce una cierta comunicación con el método de refinamiento, aportando una diversificación fija, BL Iterativa. En la Sección 3.5, estudiamos el uso de AGLB en una MH basada en BL que suple una diversificación adaptativa, Búsqueda de Vecindario Variable. Finalmente, indicamos las conclusiones y trabajos futuros en la Sección 3.6.

3.1. AGL Binario

En esta sección, proponemos un AGL que puede emplearse como componente intensificador en MHs basadas en BL, reemplazando al método clásico de BL. AGLB es un AG estacionario (véase la Sección 1.6.4) que aplica un *método de reemplazo por agrupamiento* para favorecer la formación de *nichos* en P (agrupaciones de cromosomas de alta calidad localizados en diferentes regiones del espacio de búsqueda). Después, AGLB realiza un proceso de BL que refina la solución dada, *solución actual* (X^a), de acuerdo a la información en los nichos más cercanos, los cuales representarán las regiones de búsqueda más importantes. Este proceso se lleva a cabo cruzando X^a con soluciones similares (ver la Figura 3.1).

Una propiedad destacable de AGLB es que realiza una optimización describiendo una trayectoria en el espacio de búsqueda, como lo hacen los métodos clásicos de BL. La mayoría de los métodos de BL siguen un paradigma de *ascensión de colinas*; comienzan a partir de una solución y, en cada paso, generan una

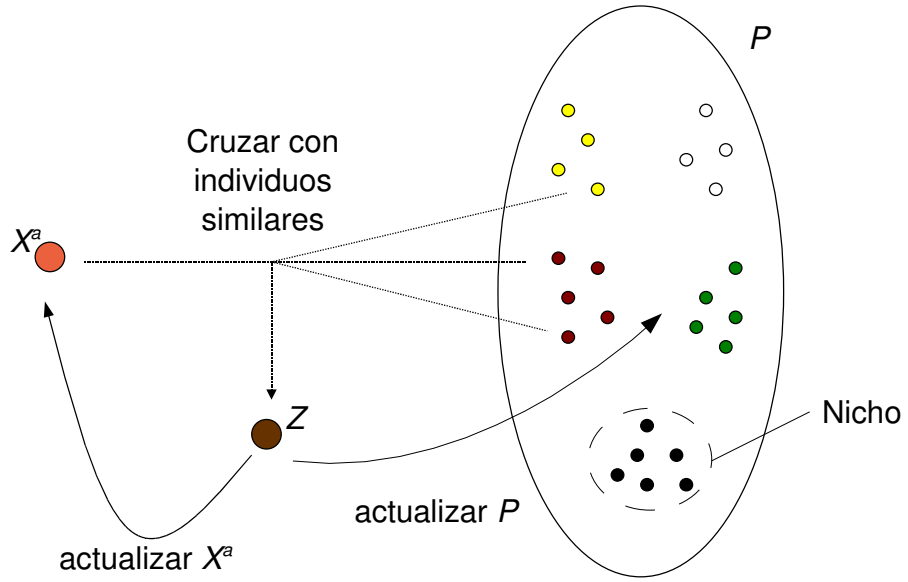


Figura 3.1: Idea de AGLB

solución candidata usando algún operador de modificación. Después, aceptan la solución candidata si presenta un valor de aptitud superior que la solución actual. La idea básica de AGLB es usar la ascensión de colinas como el criterio de aceptación y el cruce como el operador de modificación. Jones (1995) y O'Reilly y Oppacher (1995) fueron los primeros en sugerir este esquema de BL basado en cruce y, en la Sección 2.3.2 presentamos algunos modelos de AGLs que siguen el mismo esquema. La principal novedad de AGLB es su capacidad de adquirir información sobre la localización de las mejores regiones de búsqueda (mediante la formación de nichos), que después se usa para generar individuos alrededor de X^a mediante el operador de cruce. En las siguientes secciones describimos en detalle, los principales componentes de nuestra propuesta de AGL.

3.1.1. Esquema General de AGLB

Supongamos que una MH basada en BL invoca a AGLB para refinar una solución particular. Entonces, AGLB considera esta solución como X^a y lleva a cabo los siguientes pasos (Ver Figura 3.1):

1. *Selección de padres*. Se escogen m cromosomas ($Y = \{Y^1, Y^2, \ldots, Y^m\}$) similares a X^a, aplicando m veces *emparejamiento variado positivo* (Sección 3.1.2).

2. *Cruce*. Se cruzan los m padres escogidos anteriormente con X^a mediante el *operador de cruce uniforme multipadre*, generando un descendiente Z similar a X^a (Sección 3.1.3).

3. *Actualización de* X^a. X^a se actualiza si Z es mejor que X^a, en otro caso, X^a permanece igual.

4. *Actualización de P*. Finalmente, la solución descartada en el paso anterior, X^a o Z, reemplaza a un miembro de la población seleccionado según *selección por torneo restringido* (Sección 3.1.4).

Estos pasos se repiten hasta alcanzar la condición de parada que se describe en la Sección 3.1.5. Un aspecto importante a la hora de usar AGLB en una MH basada en BL es que P sólo debe inicializarse una vez, al inicio de la ejecución de la MH, y no en cada invocación como método de BL. De esta forma, AGLB puede crear nichos estables y usar la experiencia acumulada de refinamientos previos para mejorar los futuros.

3.1.2. Emparejamiento Variado Positivo

El *emparejamiento variado* es la ocurrencia natural de emparejamiento entre individuos con fenotipo similar más o menos a menudo que lo esperado por casualidad. La mayor ocurrencia de emparejamiento entre individuos con fenotipo similar se llama *emparejamiento variado positivo*, mientras que la menor ocurrencia se conoce como *emparejamiento variado negativo*. Fernandes y Rosa (2001) implementan estas ideas para diseñar dos mecanismos de selección por emparejamiento. Primero, se selecciona un padre por el método de la ruleta y después, otros n_{ass} cromosomas, por el mismo método. Entonces, se mide la similitud entre cada uno de los n_{ass} cromosomas y el primer padre. Si el emparejamiento variado es negativo, se elige el cromosoma con menor similitud como segundo padre. Si el emparejamiento es positivo, se elige el más similar.

AGLB utiliza emparejamiento variado positivo con dos pequeñas diferencias. Por un lado, el primer padre que participa en el cruce es siempre X^a. Por otro lado, los otros n_{ass} cromosomas se seleccionan aleatoriamente entre todos los de la población, esto es, sin guiarse por su valor de aptitud. Además, hemos de mencionar dos aspectos sobre el uso de este esquema de selección en AGLB: uno es que el método se aplica m veces para obtener m padres similares a X^a. El otro es que, dado que AGLB trata con cromosomas binarios, hemos de definir una función de similitud para individuos de este tipo. Nosotros hemos escogido la distancia *Hamming* entre los dos individuos, esto es, el número de genes en común.

La motivación por usar emparejamiento variado positivo viene por las siguientes dos propiedades:

- Al seleccionar padres similares a X^a, el proceso de búsqueda se centrará en la región donde ésta se encuentra, por lo que este esquema ayuda a AGLB a realizar un refinamiento local sobre X^a.

- Emparejamiento variado positivo selecciona probabilísticamente cromosomas que pertenecen a nichos cercanos a X^a, haciendo uso de la información relevante, representada en P, para optimizar X^a. La Figura 3.2 muestra el efecto de cruzar X^a con individuos de los nichos más cercanos (círculos verdes).

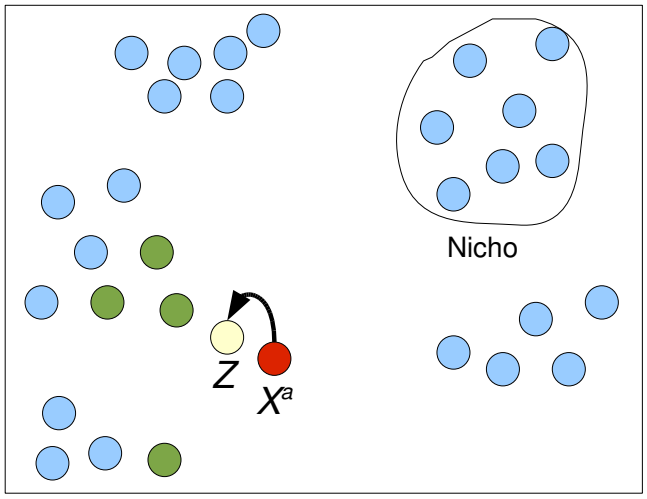


Figura 3.2: Proceso de búsqueda realizado por AGLB

3.1.3. Cruce Uniforme Multipadre con Memoria a Corto Plazo

AGLB usa una versión *multipadre* del *cruce uniforme parametrizado* (Spears y De Jong, 1991; Syswerda, 1989) específicamente diseñada para crear un descendiente cercano a X^a. El operador de cruce uniforme parametrizado crea un descendiente a partir de dos padres, eligiendo genes del primer padre con probabilidad p_f. De esta forma, utilizando un valor de p_f alto, el descendiente sera similar al primer padre. La versión *multipadre* recibe X^a y un conjunto de padres ($Y = \{Y^1, Y^2, \ldots, Y^m\}$). Entonces, crea un descendiente con genes de X^a, con probabilidad p_f, y genes de padres aleatorios del conjunto de padres.

Además, añadimos dos mecanismos para asegurarnos de que aparece material genético nuevo cada vez que se aplica el operador:

- Un mecanismo de *memoria a corto plazo* facilita el muestreo de nuevos descendientes en regiones diferentes a aquéllas donde ya se habían generado previamente, dentro de la proximidad de X^a. Para ello, el mecanismo almacena en una memoria (M) los genes donde hay diferencias entre X^a y cualquier descendiente de ésta, generado previamente ($M = \{i : z_i^k \neq x_i^a, \forall Z^k$ descendiente de $X^a\}$). Entonces, se prohíben las diferencias entre X^a y el nuevo descendiente Z^{k+1} en los genes almacenados en M. Inicialmente, y siempre que un descendiente reemplaza a X^a, M estará vacía.

- Si al final del proceso Z es igual a X^a (no ha habido modificaciones), entonces se escogerá un gen aleatorio z_i, tal que $i \notin M$, y se modificará el valor del gen, con la idea de generar material genético nuevo.

La Figura 3.3 muestra el seudocódigo del operador de cruce uniforme parametrizado multipadre con memoria a corto plazo donde $U(0,1)$ es un número

```
cruce(X^a, Y^1, ..., Y^m, M, p_f){

   for (i = 1, ..., n){

      if (i ∈ M OR U(0,1) < p_f)
         z_i ← x_i^a;

      else{
         k ← RI(1,m);
         z_i ← y_i^k;

         if (z_i ≠ x_i^a)
            M ← M ∪ i;
      }
   }

   if (Z = X^a){
      j ← RI(1,n) tal que i ∉ M;
      M ← M ∪ i;
      z_i ← 1 − z_i;
   }

   devolver Z;
}
```

Figura 3.3: Seudocódigo del operador de cruce uniforme parametrizado multi-padre con memoria a corto plazo

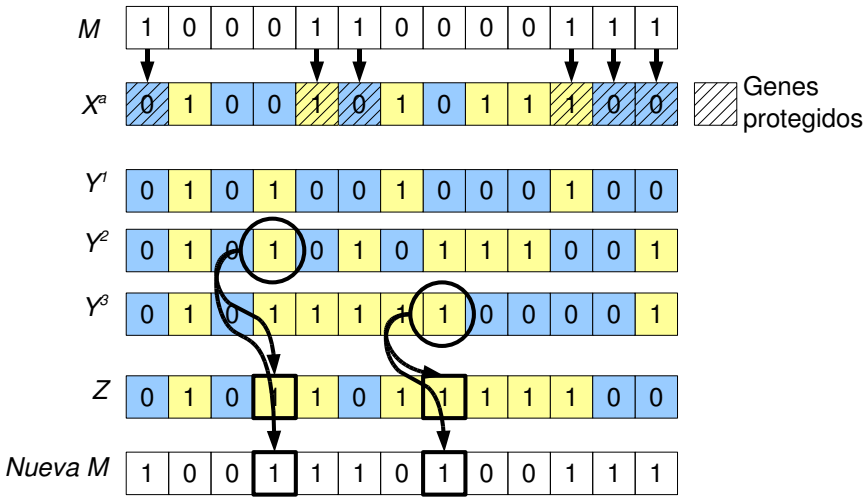


Figura 3.4: Ejemplo de aplicación del operador de cruce

aleatorio uniforme del intervalo $[0, 1]$ y $RI(1, m)$ un entero aleatorio uniforme del conjunto $\{1, 2, \ldots, m\}$. Además, la Figura 3.4 muestra un ejemplo de aplicación de este operador. En ella, M se ha implementado como un vector de valores binarios donde un 1 indica que ese gen está en M, y por tanto z_i debe ser igual a x_i^a; y Z recibe genes de padres aleatorios con una probabilidad baja, los cuales actualizan M si producen un cambio con respecto a X^a. Resumiendo, se crea un descendiente Z con las siguientes propiedades:

- z_i es igual a x_i^a para todos los genes indicados en M.
- Si el gen i no está marcado en la memoria, z_i será igual a x_i^a con probabilidad p_f. En otro caso, z_i toma el valor del gen i de un padre aleatorio Y^j. El gen i se marcará en la memoria si z_i es distinto a x_i^a.
- Finalmente, si Z y X^a son iguales, entonces, se cambiará un gen aleatorio z_k no marcado en la memoria (el cual quedará marcado).

3.1.4. Selección por Torneo Restringido

Los *métodos de agrupamiento* (Mahfoud, 1992) intentan mantener, en la población del AG, nichos estables mediante la estrategia de reemplazo como sigue: los nuevos individuos reemplazan con mayor probabilidad individuos existentes que son similares a ellos según similitud genotípica. Este tipo de métodos se usan para localizar y mantener múltiples óptimos locales en problemas multimodales.

AGLB considera la *selección por torneo restringido* (De Jong, 1975; Harik, 1995) como método de agrupamiento que reemplaza el cromosoma más similar al que se inserta en la población, de un conjunto de n_T elementos seleccionados de forma aleatoria. La aplicación de este método, junto con el uso de un tamaño de población alto, favorece la creación de nichos en la población. Posteriormente, AGLB usa los nichos como información relevante para la tarea de optimizar X^a.

El seudocódigo de la selección por torneo restringido se muestra en la Figura 3.5. Éste escoge de forma aleatoria n_T (un parámetro asociado al método) miembros de la población y busca el más parecido a la solución a insertar. Si la solución a insertar es mejor, entonces, la reemplaza.

```
seleccionTorneoRestringido(P, S){
   G_T ← Seleccionar n_T miembros aleatorios de P;
   r ← El miembro más similar a S del conjunto G_T;

   if (S es mejor que R)
      Reemplazar R con S;
}
```

Figura 3.5: Seudocódigo de la selección por torneo restringido

3.1.5. Condición de Parada

Es importante destacar que, cuando la memoria a corto plazo ha marcado todos los genes (Sección 3.1.3), entonces, AGLB no podrá mejorar X^a, porque el operador de cruce sólo produciría soluciones exactamente iguales a X^a. Por ello, esta condición determinará cuando AGLB debe parar y devolver el control y X^a a la MH basada en BL.

3.2. Marco Experimental

En esta sección, diseñamos el marco experimental para el estudio de AGLB y tres MHs basadas en BL que lo aplicarán como método de refinamiento: BL con Multiarranque Aleatorio, BL Iterativa y Búsqueda de Vecindario Variable (ver Secciones 1.5.1, 1.5.2 y 1.5.3, respectivamente). Para ello, compararemos AGLB con, a nuestro conocimiento, los métodos clásicos de BL más representativos para problemas de optimización combinatoria binaria, y sus correspondientes MHs basadas en BL.

La Sección 3.2.1 describe los métodos de BL clásicos usados en los experimentos. La Sección 3.2.2 define el conjunto de problemas de prueba y las condiciones de ejecución de los algoritmos.

3.2.1. Métodos Clásicos de BL

Hemos escogido cuatro métodos clásicos de BL para problemas combinatorios con codificación binaria, presentes en la literatura:

- *BL Primer Mejor* (BL-PM) (Blum y Roli, 2003) cambia el estado de una componente aleatoria de la solución X^a y acepta la solución resultante siempre que sea mejor que X^a. Su seudocódigo se muestra en la Figura 3.6.

- *BL Mejor* (BL-Mejor) (Blum y Roli, 2003) examina todas las soluciones que se diferencian de X^a en una sola componente, y elige la mejor de ellas, siempre que sea mejor que X^a. Su seudocódigo se muestra en la Figura 3.7

- *BL Kopt* (BL-Kopt) (Merz y Katayama, 2004) es una variante de la técnica de BL *Lin-Kernighan* (Lin y Kernighan, 1973), la cual es un componente fundamental de la mayoría de propuestas de Algoritmos Meméticos para el problema del viajante de comercio (Nguyen y otros, 2007; Ray y otros, 2007; Tsai y otros, 2004b). Utiliza lo que llamamos estructuras de vecindarios Kopt (véase la Sección 1.4.2). Su idea es buscar cambios de k bits en X^a para elegir la mejor solución encontrada. Para ello, BL-Kopt cambia iterativamente el bit, aún no cambiado, que produce la mayor mejora o el menor detrimento. Después, elige la mejor solución encontrada para repetir el proceso. Su seudocódigo se muestra en la Figura 3.8.

```
BL-PM(X^a){
   X^aux ← X^a;
   hayMejora ← verdadero;

   while(hayMejora == verdadero){
      hayMejora ← falso; i ← 1;
      Ω ← permutación de {1,2,...,n};

      while(hayMejora == falso Y i ≤ n){
         x^aux_Ωi ← 1 − x^aux_Ωi;

         if (X^aux es mejor que X^a){
            hayMejora ← verdadero;
            X^a ← X^aux;
         }
         else x^aux_Ωi ← 1 − x^aux_Ωi;

         i ← i + 1;
      }
   }
}
```

Figura 3.6: Seudocódigo de BL-PM

```
BL-Mejor(X^a){
   hayMejora ← verdadero;
   X^mejor ← X^a;

   while(hayMejora == verdadero){
      X^aux ← X^mejor;
      x^mejor_1 ← 1 − x^mejor_1;

      for (i ← 2; i ≤ n; i ← i + 1){
         x^aux_i ← 1 − x^aux_i;

         if (X^aux es mejor que X^mejor) X^mejor ← X^aux;

         x^aux_i ← 1 − x^aux_i;
      }

      if (X^mejor es mejor que X^a) X^a ← X^mejor;
      else hayMejora ← falso;
   }
}
```

Figura 3.7: Seudocódigo de BL-Mejor

```
BL-Kopt(X^a){
   hayMejora ← verdadero;
   X^gMejor ← X^a;

   while(hayMejora == verdadero){
      Γ ← {1,...,n};
      X ← X^gMejor;

      while (Γ ≠ ∅){
         indice ← encontrar j ∈ Γ tal que X_{x̄_j} es la mejor;
         x_indice ← 1 − x_indice;
         Γ ← Γ \ {indice};

         if (X es mejor que X^gMejor) X^gMejor ← X;
      }

      if (X^gMejor es mejor que X^a) X^a ← X^gMejor;
      else hayMejora ← falso;
   }
}
```

Figura 3.8: Seudocódigo de BL-Kopt. $X_{\overline{x_j}}$ simboliza la solución obtenida al aplicar $x_j \leftarrow 1 - x_j$

```
BL-PKopt(X^a){
   hayMejora ← verdadero;

   while(hayMejora == verdadero){
      Γ ← {1, ..., n};
      X^gMejor ← X^a;
      X ← X^a;

      while (Γ ≠ ∅){
         Ω ← permutación de {1, 2, ..., n};
         X^aux ← X;

         for (i ← 1; i ≤ n; i ← i + 1){
            x^aux_Ωi ← 1 − x^aux_Ωi;

            if (X^aux es mejor que X){
               X ← X^aux;
               Γ \ {Ω_i};
               X^gMejor ← X;
            }
            else x^aux_Ωi ← 1 − x^aux_Ωi;
         }

         indice ← encontrar j ∈ Γ tal que X_x̄j es la mejor;
         x_indice ← 1 − x_indice;
         Γ ← Γ \ {indice};

         if (X es mejor que X^gMejor) X^gMejor ← X;
      }

      if (X^gMejor es mejor que X^a) X^a ← X;
      else hayMejora ← falso;
   }
}
```

Figura 3.9: Seudocódigo de BL-PKopt. $X_{\overline{x_j}}$ simboliza la solución obtenida al aplicar $x_j \leftarrow 1 - x_j$

- *BL Primer Kopt* (BL-PKopt) (Katayama y Narihisa, 2001; Merz y Katayama, 2004) sigue la misma idea que Kopt, pero cambiando componentes aleatorios de la solución que produzcan una mejora, si existen, o cambiando aquél que produce el menor detrimento, en otro caso. Su seudocódigo se muestra en la Figura 3.9.

3.2.2. Configuración de los Experimentos

Hemos llevado a cabo experimentos sobre un conjunto de 22 problemas de optimización, descritos en profundidad en el Apéndice A.1. La Tabla 3.1 muestra los nombres de los problemas utilizados, dimensión, máximo número permitido de evaluaciones de la función objetivo y valor del óptimo global. El objetivo en todos ellos es maximizar la función de evaluación. Debemos hacer algunos comentarios acerca del valor presentado como óptimo global de algunos de los problemas:

- En los problemas *M-Sat* y *NKLand*, 1 es el máximo valor de aptitud posible, sin embargo, es muy probable que no exista ninguna solución óptima con tal valor, dependiendo del problema concreto que se esté resolviendo.
- Los valores óptimos de los problemas *BQP* son los mejores valores presentados por Beasley (1998).
- Los valores presentados para los problemas *Maxcut*(G10) y *Maxcut*(G19) son los mejores valores conocidos, encontrados por Helmberg y Rendl (2000).
- En los problemas *Maxcut*(G12) y *Maxcut*(G43), se muestran las cotas superiores presentadas por Festa y otros (2002).
- En el problema *Maxcut*(G18), se muestra la cota superior encontrada por Fischer y otros (2006).

En general, cada ejecución de un algoritmo (MH basada en BL con un método específico de BL) sobre un problema de prueba realizará 10^5 o 10^6 evaluaciones de la función objetivo dependiendo del problema del que se trate (ver Tabla 3.1) (se ha permitido un máximo de 10^6 evaluaciones para aquellos problemas en los que todos los algoritmos obtenían mejoras significativas, aún después de las primeras 10^5 evaluaciones). La medida de rendimiento usada es la media del mejor valor de aptitud encontrado en 50 ejecuciones independientes.

Según Garcia y otros (2008), podemos utilizar el análisis estadístico no paramétrico para comparar los resultados de diferentes MHs. Dado que para aplicar los tests no paramétricos no se requieren condiciones explícitas, es recomendable que las muestras de los resultados se obtengan siguiendo el mismo criterio, es decir, calculando la misma medida de rendimiento (media, moda, etc.), sobre un mismo número de ejecuciones de cada algoritmo y problema.

Hemos considerado el siguiente procedimiento basado en tests no paramétricos, descritos con detalle en el Apéndice C.2, para analizar los resultados de los experimentos:

Tabla 3.1: Nombre, dimensión, nº máximo de evaluaciones y valor óptimo de los problemas de prueba usados

Fnc	**Nombre**	**Dim**	**Num Evals.**	f^*
1	*Deceptive*	39	10^5	390
2	*Trap*	36	10^5	220
3	*M-Sat*(100,1200,3)	100	10^5	1
4	*M-Sat*(100,2400,3)	100	10^5	1
5	*NKLand*(48,4)	48	10^5	1
6	*NKLand*(48,12)	48	10^5	1
7	*PPeaks*(50,50)	50	10^5	1
8	*PPeaks*(50,100)	100	10^5	1
9	*PPeaks*(50,150)	150	10^5	1
10	*PPeaks*(50,200)	200	10^5	1
11	*PPeaks*(50,250)	250	10^5	1
12	*PPeaks*(100,100)	100	10^5	1
13	*BQP*(50)	50	10^5	2098
14	*BQP*(100)	100	10^5	7970
15	*BQP*(250)	250	10^5	45607
16	*BQP*(500)	500	10^6	116586
17	*Maxcut*(G10)	800	10^6	2485,08
18	*Maxcut*(G12)	800	10^6	621
19	*Maxcut*(G17)	800	10^6	No conocido
20	*Maxcut*(G18)	800	10^6	1063,4
21	*Maxcut*(G19)	800	10^6	1082,04
22	*Maxcut*(G43)	1000	10^6	7027

1. Aplicación del test de *Iman-Davenport* para detectar si existen diferencias estadísticamente significativas entre los resultados de los algoritmos de un cierto grupo.

2. Si se detectan diferencias, entonces aplicamos el test de *Holm* para comparar el mejor algoritmo (el algoritmo de control es el que obtiene el mejor ranking según el método de *Friedman*) contra el resto.

3. Utilización del test de ranking de signos de *Wilcoxon* para comparar el mejor algoritmo con aquéllos para los que el test de *Holm* no encuentra diferencias significativas.

Finalmente, AGLB usará 500 individuos en su población, $m = 10$ padres, $n_{ass} = 5$, $n_T = 15$ y $p_f = 1 - 7/Dim$.

3.3. BL con Multiarranque Aleatorio Basada en AGLB

En esta sección, estudiamos el uso de AGLB como operador de BL en BL con Multiarranque Aleatorio (BLMA) (ver Sección 1.5.1). En particular, comparamos su rendimiento con el de otros algoritmos BLMA basados en métodos clásicos de BL e investigamos la forma en que la eficacia y eficiencia de AGLB afecta al rendimiento de BLMA. Hemos implementado cinco algoritmos BLMA que llamamos BLMA-{PM, Mejor, Kopt, PKopt, AGLB} según el algoritmo de BL que utilizan. Hemos forzado que los diferentes algoritmos BLMA generen las mismas soluciones iniciales para sus métodos de BL.

Debemos indicar que la idea original de BLMA es realizar un número de invocaciones independientes del método de BL que no tienen ningún tipo de relación entre ellas. Esto no es realmente cierto en BLMA-AGLB: dado que la población de AGLB no se reinicia en cada invocación, acumula información que puede emplearse en futuras invocaciones.

La Tabla B.1, en el Apéndice B.1, muestra los resultados de los algoritmos en cada problema de prueba. En la Sección 3.3.1, compararemos los resultados finales obtenidos por los distintos algoritmos BLMA mediante un análisis estadístico no paramétrico. En la Sección 3.3.2, estudiaremos el comportamiento de los diferentes algoritmos de BL dentro de esta MH.

3.3.1. BLMA-AGLB Frente a Algoritmos BLMA con Métodos Clásicos de BL

Hemos aplicado un análisis estadístico no paramétrico para detectar posibles diferencias significativas entre los resultados de los algoritmos BLMA que usan un método clásico de BL y BLMA-AGLB. La Tabla 3.2 muestra el estadístico de *Iman-Davenport* (Apéndice C.2.2) y su valor crítico a un nivel del 5 % cuando se comparan los rankings medios (calculados según el test de *Friedman*, Apéndice

C.2.1) de cada algoritmo BLMA. Podemos observar que existen diferencias significativas entre los resultados porque el valor estadístico es mayor que el crítico (2,48).

Tabla 3.2: Test de *Iman-Davenport* sobre los algoritmos BLMA

Estadístico	**Valor crítico**
9,518	2,48

Atendiendo a este resultado, podemos proceder a realizar un test *post-hoc* para comparar el algoritmo que obtiene el mejor ranking (BLMA-AGLB) con los demás. La Tabla 3.3 muestra el ranking medio de cada algoritmo BLMA (los rankings bajos son mejores) y el procedimiento estadístico de *Holm* (columnas i, z, p-valor y α/i) (Apéndice C.2.3) al nivel 0,05 de significación (el algoritmo con mejor ranking, señalado con el carácter '*', es el algoritmo de control). La última columna indica si la hipótesis nula se rechaza (R), esto es, el algoritmo de control produce resultados significativamente mejores que los del algoritmo correspondiente, o no se rechaza (N), los resultados del algoritmo de control y del correspondiente, podrían ser equivalentes.

Tabla 3.3: Ranking y procedimiento de *Holm* sobre los algoritmos BLMA

i	**Algoritmo**	**Ranking**	**z**	**p-valor**	$\boldsymbol{\alpha/i}$	**Resultado**
4	BLMA-Mejor	4,136	5,101	3,378e−7	0,0125	**R**
3	BLMA-PKopt	3,341	3,432	5,981e−4	0,017	**R**
2	BLMA-PM	3	2,717	0,007	0,025	**R**
1	BLMA-Kopt	2,818	2,336	0,019	0,05	**R**
$*$	BLMA-AGLB	1,705				

Los resultados de la Tabla 3.3 revelan que BLMA-AGLB consigue el mejor ranking y el test de *Holm* confirma que existen diferencias significativas entre su rendimiento y el de los otros algoritmos BLMA. Por ello, podemos concluir que AGLB permite a BLMA mejorar sus resultados con respecto al uso de otros métodos clásicos de BL.

3.3.2. Comportamiento de AGLB en BLMA

A continuación, pretendemos descubrir las características de AGLB que permiten mejorar el rendimiento de BLMA. En particular, pretendemos comprobar si AGLB puede emplear correctamente la experiencia acumulada de unos refinamientos para mejorar los futuros. Para ello, investigaremos cómo se comporta AGLB a través del tiempo (desde el punto de vista de eficacia y eficiencia) cuando se ejecuta como componente de BLMA.

Hemos considerado un conjunto fijo de 50 soluciones iniciales aleatorias (S_c) y estudiado los resultados de usar AGLB para refinarlos en diferentes fases de

la ejecución de una algoritmo BLMA que aplica BL sobre 250 soluciones aleatorias. Cada 50 refinamientos de BLMA, aplicamos AGLB a las soluciones de S_c, y anotamos la media del valor objetivo alcanzado y la media del número de evaluaciones consumidas. Las Figuras 3.10 y 3.11 muestran esas medidas, respectivamente, para una ejecución sobre dos problemas de prueba diferentes, $BQP(250)$ y $Maxcut(\text{G17})$. Incluimos además los resultados obtenidos por algoritmos BLMA similares con los métodos clásicos de BL. Debemos destacar que las soluciones obtenidas tras el refinamiento de los individuos en S_c nunca se introducen en la población de AGLB, es decir, estos individuos se usan exclusivamente para comprobar el rendimiento de AGLB.

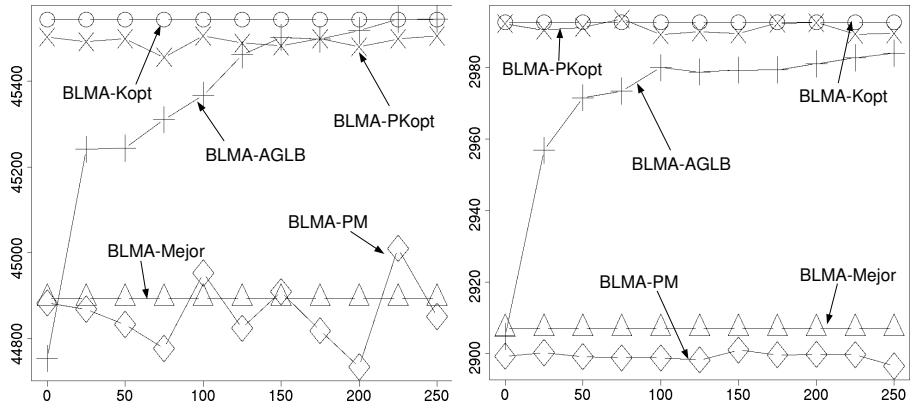


Figura 3.10: Evolución del valor objetivo para $BQP(250)$ (izquierda) y $Maxcut(\text{G17})$ (derecha)

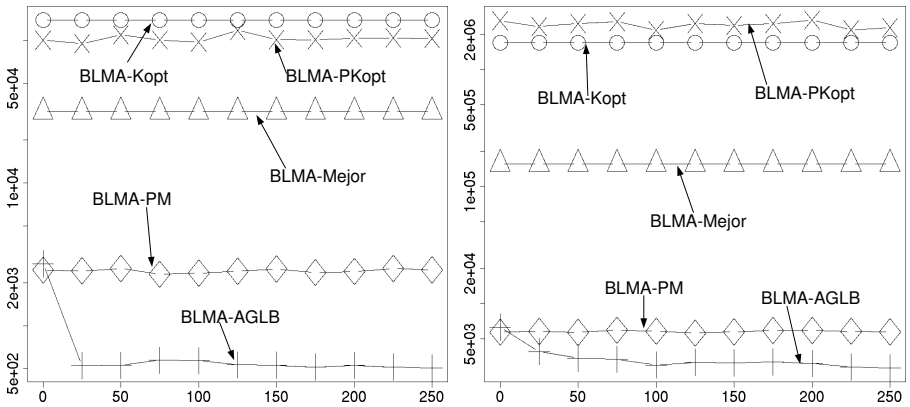


Figura 3.11: Evolución del consumo de evaluaciones para $BQP(250)$ (izquierda) y $Maxcut(\text{G17})$ (derecha)

A partir de las Figuras 3.10 y 3.11 podemos realizar los siguientes comentarios:

- El rendimiento de AGLB sobre las soluciones de S_c es mejor conforme BLMA-AGLB procesa más soluciones (ver Figura 3.10). Este hecho indica

que el conocimiento adquirido por AGLB (creando nichos con individuos de alta calidad) puede usarse de forma fructífera para guiar el refinamiento local de nuevas soluciones.

- La tendencia a mejorar exhibida por AGLB (Figura 3.10) no aparece en los otros métodos de BL. Como esperábamos, dado que ellos no incorporan ningún mecanismo de memoria, su comportamiento no se altera por decisiones previas. De hecho, en general, muestran un comportamiento similar a lo largo de toda la ejecución. Sólo podemos remarcar la conducta cambiante de BLMA-PM en el problema $BQP(250)$. En este método de BL, el orden para examinar las posiciones de la solución a cambiar es aleatorio y por ello, su comportamiento puede ser muy diferente en momentos distintos, incluso cuando utiliza las mismas soluciones iniciales (lo cual se aprecia claramente en este problema).

- AGLB empieza consumiendo un número de evaluaciones por refinamiento que es similar al de BL-PM (ver Figura 3.11). Después, AGLB reduce este consumo a lo largo de la ejecución, mientras que los métodos clásicos de BL mantienen su consumo casi constante. En las primeras fases del proceso de búsqueda, la población de AGLB no posee información útil sobre el espacio de búsqueda y guía los refinamientos de forma aleatoria, similar a como lo hace BL-PM. Después, AGLB conduce el proceso de BL hacia las zonas prometedoras que localizó previamente, aumentando la velocidad de convergencia considerablemente y haciendo más eficiente el proceso de búsqueda.

Usar un algoritmo eficiente de BL es primordial para asegurar que BLMA alcance un nivel aceptable de *fiabilidad*. Cuando se cuenta con un número limitado de evaluaciones de la función objetivo, usar un algoritmo de BL eficiente puede permitir a BLMA refinar más soluciones iniciales, favoreciendo la exploración del espacio de búsqueda. A continuación, investigamos la eficiencia de los diferentes métodos de BL en los algoritmos BLMA que analizamos en la Sección 3.3.1, los cuales consideraban un número máximo de 10^5 o 10^6 de evaluaciones por ejecución. En particular, hemos contado el número de veces que los diferentes algoritmos realizan un nuevo arranque, es decir, el número de soluciones iniciales que refinan dentro del número de evaluaciones disponibles. La Tabla 3.4 muestra el valor de esta medida (la media de 50 ejecuciones) para todos los algoritmos BLMA en cada problema de prueba. Hemos incluido, además, una columna con la dimensión del problema (la segunda), que afecta de forma importante al número de arranques que hacen los algoritmos. Además, hemos resaltado, en negrita, el mayor valor de cada línea (el mayor número de reinicios realizados para cada problema).

A partir de los resultados mostrados en la Tabla 3.4, podemos realizar los siguientes comentarios:

- BLMA-AGLB es siempre el algoritmo que refina más soluciones, dentro el número máximo de evaluaciones de cada problema. La habilidad de AGLB para converger cada vez más rápidamente, a través de la ejecución de BLMA (observado en la Figura 3.11), hace que sea finalmente el algoritmo más eficiente en todos los problemas de prueba.

Tabla 3.4: Número de reinicios realizados por cada algoritmo BLMA según su método de BL

Problema	Dim	BL-Mejor	BL-PM	BL-Kopt	BL-PKopt	AGLB
1	39	238,8	820,9	43,1	46,3	**1331,5**
2	36	213,4	870,7	53,3	46,2	**1580,2**
3	100	33,6	179,4	5,8	4,9	**411,1**
4	100	30,3	158,1	5,4	5,3	**407,1**
5	48	155,3	535,6	24,6	20,5	**878,5**
6	48	293,8	693,5	26,5	22,5	**1006,2**
7	50	106,1	555,0	34,5	29,0	**940,0**
8	100	25,2	228,5	9,0	8,0	**348,2**
9	150	11,0	137,6	4,3	4,0	**190,3**
10	200	6,1	96,7	2,8	2,0	**120,6**
11	250	4,0	73,9	1,8	2,0	**88,5**
12	100	25,9	229,5	8,7	8,0	**306,8**
13	50	103,8	442,5	23,4	20,2	**1096,9**
14	100	23,5	159,9	5,6	6,2	**459,7**
15	250	3,9	41,5	1,3	1,5	**182,0**
16	500	8,6	151,0	1,9	2,3	**905,1**
17	800	4,9	86,0	1,0	1,0	**471,7**
18	800	8,1	207,3	1,0	1,0	**326,8**
19	800	7,0	176,4	1,0	1,1	**329,9**
20	800	6,0	149,8	1,0	1,0	**457,1**
21	800	6,0	149,4	1,0	1,0	**457,0**
22	1000	3,4	82,8	1,0	1,0	**359,4**

- BLMA-Kopt y BLMA-PKopt realizan pocos reinicios. Cuando el tamaño del problema es alto (dimensiones mayores a 200), no son capaces de completar uno o dos refinamientos (es muy probable que alcancen el límite de evaluaciones antes de empezar el segundo o tercer refinamiento). Sólo para problemas de baja dimensión (50 o menos variables) pueden realizar un número significativo de reinicios. Sin embargo, para el caso de BL-Kopt, el excesivo coste computacional de cada refinamiento se premia obteniendo resultados finales muy buenos (BLMA-Kopt es el algoritmo con mejor ranking después de BLMA-AGLB en la Tabla 3.3). Debemos indicar que, dado que BL-Kopt es una variante de *Lin-Kernighan* (Lin y Kernighan, 1973), una conocida técnica de BL poderosa para el problema del viajante de comercio (Nguyen y otros, 2007; Ray y otros, 2007; Tsai y otros, 2004b), ya esperábamos esos buenos resultados.

Podemos concluir que AGLB es el método de refinamiento más apropiado para BLMA, porque combina dos características determinantes (las cuales son consecuencia de su habilidad para capturar y explotar información del espacio

de búsqueda): 1) es el método de BL más eficiente y 2) es capaz de alcanzar altos niveles de eficacia a lo largo de la ejecución (Figura 3.10). La unión de estos dos aspectos permite a BLMA-AGLB mantener un equilibrio beneficioso entre intensificación y diversificación, ofreciendo las dos ventajas al mismo tiempo: mejor fiabilidad y precisión. Esta clase de equilibrio no lo alcanza ninguno de los algoritmos BLMA que aplican técnicas clásicas de BL. Por ejemplo, probablemente, la poca eficiencia mostrada por BL-Kopt hace que BLMA-Kopt sea incapaz de alcanzar la fiabilidad necesaria para mejorar a BLMA-AGLB (Tabla 3.3).

3.4. BL Iterativa Basada en AGLB

En esta sección, estudiamos las ventajas que consigue BL Iterativa (BLI) (ver Sección 1.5.2) cuando usa como método de BL a AGLB, frente al uso de métodos clásicos de BL. Hemos implementado varios algoritmos BLI que utilizan el operador de perturbación estándar sobre la mejor solución encontrada hasta el momento (se cambia el valor de sus bits con una probabilidad correspondiente a la intensidad de perturbación indicada), denominados BLI-σ_p-{PM, Mejor, Kopt, PKopt, AGLB} según la intensidad de perturbación y el método de BL que utilizan. El marco experimental es el descrito en la Sección 3.2. A continuación, estudiamos primero cómo afecta la diversificación introducida por el valor de intensidad de perturbación, a cada combinación de algoritmo BLI y método de BL. Después, compararemos los mejores algoritmo BLI entre ellos.

3.4.1. Influencia de la Intensidad de Perturbación

En esta sección, investigamos la influencia del parámetro σ_p en el rendimiento de los algoritmos BLI. En particular, analizamos el comportamiento de estos algoritmos cuando utilizan diferentes valores para este parámetro ($\sigma_p = 0{,}1$, 0,25, 0,5 y 0,75). Las Tablas B.2 a B.5, en el Apéndice B.1, muestran los resultados de cada algoritmo BLI cuando trata cada uno de los problemas de prueba. Para cada método de BL, la Figura 3.12 muestra el ranking medio obtenido por los distintos algoritmos BLI con diferentes valores de σ_p, cuando los comparamos entre ellos.

En la Figura 3.12, hay dos hechos importantes a remarcar:

- Los algoritmos BLI con mejor ranking, de los que aplican AGLB, BL-Kopt o BL-PKopt, usan $\sigma_p = 0{,}5$. Debido a su efectividad, AGLB, BL-Kopt y BL-PKopt son capaces de afrontar con éxito intensidades de perturbación elevadas, lo que significa que es posible saltar a nuevas regiones de búsqueda no exploradas y encontrar eventualmente (por medio de los métodos de BL) una nueva mejor solución. Esto es esencial para garantizar fiabilidad en el proceso de búsqueda de BLI.

- BLI-PM y BLI-Mejor obtienen sus mejores resultados con $\sigma_p = 0{,}1$ y 0,25, respectivamente. Esto indica que BL-PM y BL-Mejor obtienen mejores soluciones cuando BLI no perturba severamente la solución actual. Sin

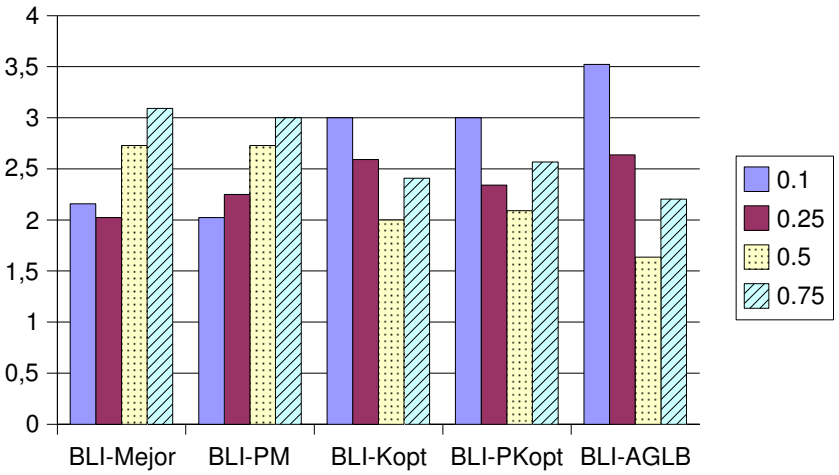


Figura 3.12: Rankings de los algoritmos BLI con mismo método de BL

embargo, esto puede dar lugar a un inconveniente importante: BLI puede ser incapaz de explorar suficientes regiones del espacio de búsqueda y obtener soluciones poco fiables en problemas complejos.

3.4.2. BLI-AGLB Frente a Algoritmos BLI con Métodos Clásicos de BL

En esta sección, comparamos los resultados de los algoritmos BLI que obtenían el mejor ranking para cada método de BL: BLI-0,25-Mejor, BLI-0,1-PM, BLI-0,5-Kopt, BLI-0,5-PKopt y BLI-0,5-AGLB. La Tabla 3.5 muestra el estadístico de *Iman-Davenport* y su valor crítico al nivel 5 % cuando comparamos los rankings de los algoritmos BLI seleccionados. Observamos que existen diferencias significativas y procedemos a realizar un estudio *post-hoc*. La Tabla 3.6 muestra el ranking medio de los algoritmos y el test de *Holm* al nivel 0,05 de significación. El algoritmo con mejor ranking (BLI-0,5-AGLB) es el de control. Por brevedad, hemos omitido los pasos intermedios del test de *Holm* (i, z, p-valor y α/i). Además, hemos aplicado el test de *Wilcoxon* por parejas (ver Apéndice C.2.4) entre el algoritmo de control y aquéllos con los que el test de *Holm* no encontró diferencias estadísticas. Las últimas cuatro columnas muestran R^-, asociado al algoritmo de control, R^+, al algoritmo correspondiente, el valor crítico, y el resultado del test. El resultado será '+' si el rendimiento del algoritmo de control es estadísticamente superior al del algoritmo correspondiente, '−' si el rendimiento del algoritmo correspondiente es superior al del algoritmo de control, y '∼' si no hay diferencias significativas.

De la Tabla 3.6, observamos claramente que BLI-0,5-AGLB obtiene resultados significativamente mejores a los de los otros algoritmos BLI. La habilidad de este algoritmo para procesar niveles de diversidad altos ($\sigma_p = 0{,}5$) puede explicar, en parte, su ventaja frente a BLI-0,25-Mejor y BLI-0,1-PM, mientras que la eficiencia superior de AGLB sobre BL-Kopt y BL-PKopt (Sección 3.3.2)

Tabla 3.5: Test de *Iman-Davenport* sobre los algoritmos BLI

Estadístico	**Valor crítico**
8,052	2,480

Tabla 3.6: Tests de *Holm* y *Wilcoxon* sobre los algoritmos BLI

			Wilcoxon			
Algoritmo	**Ranking**	***Holm***	R^+	R^-	**Cr.**	**Resultado**
BLI-0,25-Mejor	4,023	**R**				
BLI-0,5-PKopt	3,5	**R**				
BLI-0,1-PM	3	**R**				
BLI-0,5-Kopt	2,636	N	62	191	65	+
∗ BLI-0,5-AGLB	1,841					

puede justificar la ventaja sobre los algoritmos BLI basados en estos métodos de BL. Por ello, podemos concluir que AGLB es una buena técnica de BL para BLI, siendo muy competitiva frente al uso de otros métodos de BL presentados en la literatura.

3.5. Búsqueda de Vecindario Variable Basada en AGLB

En esta sección, pretendemos analizar el comportamiento de AGLB dentro de un algoritmo de Búsqueda de Vecindario Variable (BVV) (ver Sección 1.5.3), que llamaremos BVV-AGLB. En particular, llevaremos a cabo un estudio experimental para comprobar si la eficacia y eficiencia asociada a AGLB (Sección 3.3.2) y su capacidad para afrontar diferentes intensidades de perturbación (Sección 3.4.1) son apropiadas para que BVV pueda alcanzar su objetivo: producir mejoras constantemente. También compararemos la calidad de las soluciones obtenidas por BVV-AGLB con respecto a las de otros algoritmos BVV que usan métodos clásicos de BL (BVV-{PM, Mejor, Kopt, PKopt}). Todos los algoritmos BVV usan el operador de perturbación estándar para generar los vecindarios. Iterarán sobre nueve intensidades de perturbación (0,1, 0,2, 0,3, 0,4, 0,5, 0,6, 0,7, 0,8, 0,9), volviendo a la primera si no se encuentra una nueva mejor solución después de haber usado la última intensidad, hasta alcanzar el máximo número de evaluaciones. El marco experimental es el descrito en la Sección 3.2.

3.5.1. Sinergia entre BVV y AGLB

BVV hace uso de una *heurística* que ajusta el valor de σ_p con el objetivo de encontrar, en cada iteración y después de aplicar el método de BL, una solución que sea mejor a la mejor encontrada hasta el momento. En esta sección, pretendemos determinar si el uso de AGLB como método de BL de BVV permite a esta heurística actuar de forma útil, lo que significará que existe una sinergia positiva entre BVV y AGLB.

Para investigar la sinergia entre BVV y AGLB, hemos contado el número de *reinicios con éxito* (veces en que el operador de perturbación generó una nueva solución que, tras refinarse con el método de BL, mejoró la mejor solución hasta el momento) producidos a través de la ejecución de BVV-AGLB, y lo comparamos con el de BLMA-AGLB y BLI-0,5-AGLB (el mejor algoritmo BLI basado en AGLB encontrado en la Sección 3.4.1). Además, hemos calculado la misma medida para otros algoritmos BVV basados en métodos clásicos de BL, y comparado con sus correspondientes versiones de algoritmos BLMA y mejores BLI. La Tabla B.7, en el Apéndice B.1, muestra los resultados (la media de 50 ejecuciones).

La Figura 3.13 muestra el ranking medio de cada algoritmo BVV, sobre el número de reinicios con éxito (según los datos de la Tabla B.7), cuando los comparamos con los resultados de otras MHs basadas en BL que usan el mismo método de BL. Además, las Tablas 3.7 y 3.8 aplican un análisis estadístico sobre estos datos. Es importante notar que la Tabla 3.8 sólo compara las MHs basadas en BL para las cuales el test de *Iman-Davenport* encuentra diferencias significativas (MHs basadas en AGLB, BL-PM y BL-Mejor).

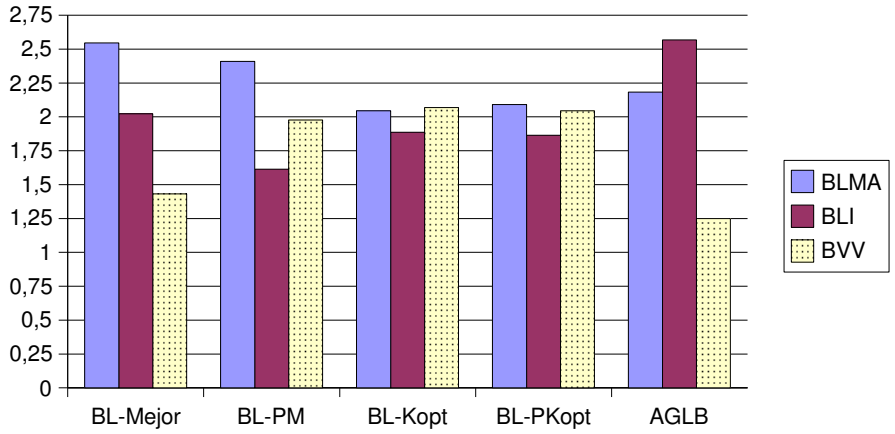


Figura 3.13: Rankings de las MHs basadas en BL según número de reinicios con éxito

Podemos destacar los siguientes hechos:

- BVV-AGLB y BVV-Mejor obtienen los mejores rankings cuando los comparamos con sus correspondientes versiones de BLMA y BLI. Además, en la Tabla 3.8 observamos que los resultados son estadísticamente significa-

Tabla 3.7: Tests de *Iman-Davenport* sobre el número de reinicios con éxito de las MHs basadas en BL

MHs basadas en...	Estadístico	Valor Crítico
AGLB	17,831	3,220
BL-PM	3,958	3,220
BL-Mejor	9,454	3,220
BL-Kopt	0,208	3,220
BL-PKopt	0,308	3,220

Tabla 3.8: Tests de *Holm* y *Wilcoxon* sobre el número de reinicios con éxito de las MHs basadas en BL

			Wilcoxon			
Algoritmos	**Ranking**	**Holm**	R^+	R^-	**Cr.**	**Resultado**
BLI-0,5-AGLB	2,568	**R**				
BLMA-AGLB	2,182	**R**				
$*$ BVV-AGLB	1,25					
BLMA-PM	2,41	**R**				
BVV-PM	1,977	N	117,5	135,5	65	$\sim$
$*$ BLI-0,1-PM	1,614					
BLMA-Mejor	2,545	**R**				
BLI-0,25-Mejor	2,023	N	36,5	216,5	65	$+$
$*$ BVV-Mejor	1,432					

tivos. Podemos concluir que la heurística de BVV para adaptar σ_p resulta beneficiosa cuando se usan AGLB y BL-Mejor, con respecto al uso de un valor constante para σ_p (caso de BLI) y la generación de soluciones iniciales aleatorias (caso de BLMA). Por tanto, existe una sinergia positiva entre BVV y estas dos técnicas de BL.

- BVV-PM no consigue alcanzar un número de reinicios con éxito superior al conseguido por BLI-0,1-PM, el cual obtiene el mejor ranking con respecto a esta medida. Esto indica que no se produce una sinergia positiva entre BVV y BL-PM. En la Sección 3.4.1, observamos que esta técnica de BL responde de forma pobre a niveles de diversificación altos. De esta forma, la acción de incrementar σ_p, cuando no hay mejoras sobre la mejor solución hasta el momento, no produce, en general, el efecto esperado.

- BVV no se ve favorecida con la integración de BL-Kopt o BL-PKopt. Hay dos posibles razones para esta circunstancia: 1) son métodos de BL que

consumen muchas evaluaciones por refinamiento (se realizan pocos reinicios) y por tanto, practicamente no permite que el proceso de adaptación de σ_p produzca efectos notables a través de la ejecución; y 2) son métodos muy eficaces, alcanzando soluciones muy precisas desde el primer refinamiento (por ejemplo, podemos comparar los resultados de BLMA-Kopt y BLMA-PKopt en el problema *Maxcut*(G10), donde sólo realizaron un refinamiento, con los de los otros algoritmos), lo que dificulta el poder encontrar mejores soluciones en iteraciones posteriores.

3.5.2. BVV-AGLB Frente a Algoritmos BVV con Métodos Clásicos de BL

Finalmente, comparamos la calidad de las soluciones obtenidas por BVV-AGLB y las obtenidas por los algoritmos BVV que usan métodos clásicos de BL. La Tabla B.6, en el Apéndice B.1, presenta los resultados de cada algoritmo BVV para cada problema de prueba. La Tabla 3.9 muestra el estadístico de Imán-Davenport y su valor crítico al 5 %, cuando comparamos los rankings medios de los algoritmos BVV. Podemos observar que existen diferencias significativas porque el estadístico es mayor que el valor crítico. La Tabla 3.10 compara los resultados del algoritmo con mejor ranking (BVV-AGLB) con los de los otros algoritmos BVV por medio de los tests de *Holm* y *Wilcoxon*.

Tabla 3.9: Test de *Iman-Davenport* sobre los algoritmos BVV

Estadístico	**Valor Crítico**
13,695	2,48

Tabla 3.10: Test de *Holm* y *Wilcoxon* sobre los algoritmos BVV

			Wilcoxon			
Algoritmos	**Ranking**	***Holm***	R^+	R^-	**Cr.**	**Resultado**
BVV-Mejor	3,841	**R**				
BVV-PKopt	4,045	**R**				
BVV-Kopt	3,068	**R**				
BVV-PM	2,364	N	62	191	65	+
∗ BVV-AGLB	1,682					

Claramente, observamos que BVV-AGLB obtiene resultados estadísticamente mejores que los de los otros algoritmos BVV. La sinergia positiva entre AGLB y BVV ha permitido a BVV-AGLB exhibir un rendimiento global superior al de los otros algoritmos. Esta sinergia es posible por: 1) la eficacia de AGLB, que permite obtener mejores soluciones incluso cuando σ_p toma valores altos

y 2) su eficiencia, que permite a BVV alcanzar rápidamente valores de σ_p que dan lugar a una mejora (es decir, el número de evaluaciones consumidas en inspecciones fallidas es menor). Estos hechos son determinantes para mejorar a los otros algoritmos BVV. Dado que los métodos clásicos de BL son menos eficientes que AGLB, sus correspondientes algoritmos BVV no dispusieron de suficientes evaluaciones para localizar soluciones con calidad superior a las alcanzadas por BVV-AGLB.

3.6. Conclusiones

En este capítulo, hemos propuesto AGLB, un AGL que aplica un *método de reemplazo por agrupamiento* para favorecer la formación de *nichos* en P. Después, AGLB realiza un proceso de BL que refina la solución dada de acuerdo a la información en los nichos más cercanos, los cuales representan las regiones de búsqueda más importantes.

Posteriormente hemos realizado un extenso estudio, con la intención de obtener nuevos resultados y conclusiones sobre los AGLs y las MHs basadas en AGLs. Hemos utilizado tres MHs basadas en BL (BLMA, BLI y BVV) que nos han permitido estudiar, de forma controlada, la relación entre AGLB y diferentes componentes diversificadores de MHs basadas en BL. Para ello, hemos estudiado los beneficios de usar AGLB como método de BL frente al uso de otros modelos clásicos de BL. Los resultados nos han permitido concluir que:

- AGLB alcanza altos niveles de eficiencia, lo que permite a la MH basada en BL realizar un mayor número de reinicios dentro de un número máximo de evaluaciones. Además, muestra un poder de intensificación beneficioso, que es capaz de afrontar los niveles de diversificación provistos por operadores con alta intensidad de perturbación (valores de σ_p elevados). AGLB presenta estas dos características gracias a su habilidad para guiar el proceso de BL usando el conocimiento de las zonas del espacio de búsqueda visitadas en invocaciones previas.

- La unión de estas dos propiedades hacen posible que BLMA, BLI y BVV afronten exitosamente el conflicto entre precisión y fiabilidad para mejorar sus resultados con respecto al uso de métodos clásicos de BL. Por tanto, AGLB es un algoritmo prometedor como método de BL para MHs basadas en BL.

Además, debemos destacar que nuestro estudio también ha revelado algunas propiedades del comportamiento de varios métodos clásicos de BL cuando los utilizamos dentro de diferentes MHs basadas en BL.

Los buenos resultados de AGLB, en diferentes MHs basadas en BL y sobre un amplio conjunto de problemas de prueba, nos animan a seguir realizando trabajos de investigación en esta línea. Como trabajos futuros, centrados en AGLB, destacamos que éste puede modificarse para incluir técnicas de diversificación como las presentes en Enfriamiento Simulado o Búsqueda Tabú. Mientras que AGLB se especializa en intensificación, Enfriamiento Simulado y Búsqueda

Tabú son MHs completas que intentan realizar una exploración suficientemente amplia del espacio de búsqueda. En un futuro próximo, pretendemos estudiar los beneficios de usar conceptos de los AGs en este tipo de métodos.

Capítulo 4

AG con Codificación Real Local

En el mundo real, muchos de los problemas de optimización presentan variables que tienen un dominio continuo, esto es, sus valores se toman del espacio real ($\Re$). A este tipo de problemas se les conoce como *problemas de optimización continua*.

En la Sección 1.6.5, ofrecimos una introducción a los AGs que tratan este tipo de problemas. Aunque en su formulación inicial, los AGs utilizaban el alfabeto binario para codificar las soluciones, otro tipo de codificaciones, como la real (Davis, 1991b; Deb y Beyer, 2001; Deb, 2005; Herrera y otros, 1998; Michalewicz, 1992), se han tenido en cuenta. El estudio de *AGs con codificación real* (AGCRs) ha recibido especial atención por muchos investigadores (Chelouah y Siarry, 2000; Deb y Tiwari, 2008; Herrera y otros, 2005; Hervás-Martínez y Ortiz-Boyer, 2005; Poojari y Varghese, 2008; Someya y Yamamura, 2005; Winter y otros, 2005; Yang y Kao, 2000), y recientemente hay un creciente interés en resolver problemas del mundo real mediante este tipo de algoritmos.

En los AGs, se ha considerado al operador de cruce como uno de los principales motores de búsqueda (De Jong y Spears, 1992; Kita, 2001) porque explota la información disponible de muestras previas para influenciar las búsquedas futuras. Por ello, la mayor parte de la investigación en AGCRs se ha centrado en el desarrollo de operadores de cruce para optimización continua efectivos, y como resultado, se han presentado muchas propuestas diferentes (Deb y Beyer, 2001; Herrera y otros, 1998, 2003).

En el campo de los AGLs para la optimización continua, los investigadores también han puesto énfasis en el operador de cruce. La mayoría de las propuestas de AGLs para optimización continua se basan en desarrollar *operadores de ascensión de colinas basados en el cruce* (Lozano y otros, 2004; Mutoh y otros, 2006; Noman y Iba, 2008), aprovechando la habilidad auto-adaptativa que muestran este tipo de operadores, la cual crea descendientes según la distribución de los padres sin usar ningún parámetro adaptativo.

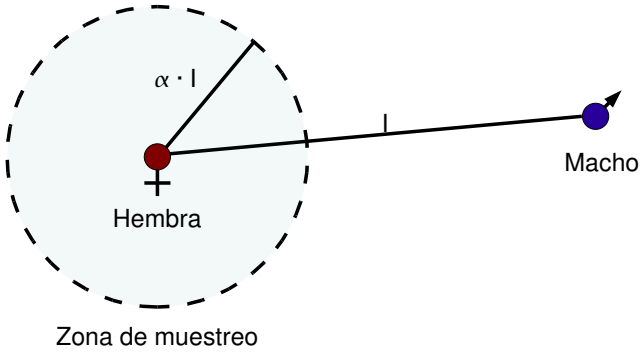


Figura 4.1: Papeles de los progenitores hembra y macho

Los *operadores de cruce centrados en un padre* (OCCPs) son una familia especial de operadores de cruce que ha recibido especial atención para el diseño de AGCRs (Ballester y Carter, 2003, 2004; Deb y Agrawal, 1995; Deb y otros, 2002a; Takahashi y Kobayashi, 2001; Voigt y otros, 1995) . En general, estos operadores usan una distribución de probabilidad para crear descendientes en una región restringida del espacio de búsqueda marcada por uno de los padres, el *progenitor hembra*. El rango de esta distribución de probabilidad depende de la distancia entre el progenitor hembra y el otro padre seleccionado para el cruce, *el progenitor macho* (ver Figura 4.1). Los experimentos llevados a cabo por Deb y otros (2002a) han mostrado que los OCCPs son una forma eficiente y significativa de resolver problemas de optimización continua.

En este capítulo, pretendemos diseñar *AGCRs especializados*, basados en el uso de OCCPs, capaces de promover intensificación útil a la hora de tratar con problemas de optimización continua. Llamaremos a este tipo de AGs, *AGCRs Locales*. Para ello, propondremos la aplicación de tres mecanismos que, determinando los progenitores que intervienen en la aplicación del OCCP, permiten ajustar de forma precisa el equilibrio entre intensificación y diversificación aportado por el AGCR:

1. *Procedimiento de diferenciación sexual* (PDS). Éste crea dos grupos de cromosomas a partir de la población: 1) G_H con los N_H cromosomas de la población que pueden ser progenitores hembra; y 2) G_M con los N_M individuos que pueden actuar como progenitores macho (N_H y N_M son dos parámetros del procedimiento). Este proceso puede considerarse como un *mecanismo de preselección*.

2. *Selección del progenitor hembra*. Presentamos un nuevo método para la selección del progenitor hembra, la *selección uniforme en fertilidad*, el cual intenta asignar un número justo de descendientes a los cromosomas que visitan la población, con la intención de proveer una búsqueda amplia del espacio. Para eso, el método selecciona, como progenitor hembra, el individuo en G_H que ha generado, con anterioridad, el menor número de descendientes.

3. *Selección del progenitor macho*. Hemos considerado el *emparejamiento variado negativo* (Fernandes y Rosa, 2001; Matsui, 1999) para la selección

del progenitor macho. Dado el conjunto de candidatos machos G_M, este procedimiento escoge el cromosoma con mayor distancia Euclídea al progenitor hembra. Con esta estrategia, forzamos usar distribuciones de probabilidad amplias, que favorecen la creación de descendientes con material genético diverso.

Como resultado, el ajuste preciso de los parámetros de PDS (N_H y N_M) permite especializar el nuevo modelo de AGCR basado en OCCPs y obtener:

- *AGCRs Locales* que obtienen soluciones precisas a la hora de alcanzar un óptimo local, y también
- *AGCRs Globales* que obtienen soluciones fiables, realizando una exploración global del espacio de búsqueda.

Posteriormente, con la intención de obtener un método robusto que obtenga soluciones de alta calidad en problemas de optimización continua con diferentes características, diseñaremos una *MH híbrida* que combine los dos tipos de AGCRs especializados, un AGCR Local y un AGCR Global, y aproveche las ventajas de ambos simultaneamente. Finalmente, compararemos la MH resultante con otras propuestas de la literatura sobre un amplio conjunto de este tipo de problemas.

El capítulo se organiza de la siguiente forma: en la Sección 4.1 introducimos aspectos relevantes de los OCCPs y describimos PBX-α (Lozano y otros, 2004), un OCCP considerado en este estudio. En la Sección 4.2, proponemos el nuevo método para la selección de progenitores hembra, selección uniforme en fertilidad. Además, estudiamos su efectividad comparándola con la de otros mecanismos de selección de padres presentados en la literatura. En la Sección 4.3, examinamos los efectos producidos por el mecanismo de selección del progenitor macho sugerido, emparejamiento variado negativo, sobre la operación de PBX-α. Debemos remarcar que la selección uniforme en fertilidad y el emparejamiento variado negativo se presentan y estudian antes que PDS para facilitar sus análisis (los estudios de las Secciones 4.2 y 4.3 no aplican PDS). En la Sección 4.4, presentamos PDS y llevamos a cabo experimentos para investigar su comportamiento cuando se incorpora a un AGCR con selección uniforme en fertilidad y emparejamiento variado negativo. En la Sección 4.5, explicamos cómo PDS nos permite diseñar AGCRs que, mediante el ajuste de sus parámetros (N_H y N_M), se especializan en AGCRs Locales o AGCRs Globales. Además, diseñamos una MH híbrida que combina un AGCR Local y un AGCR Global, y comparamos su rendimiento con el de otras MHs propuestas en la literatura para el problema de la optimización continua. Finalmente, presentamos las conclusiones y trabajos futuros en la Sección 4.6.

4.1. Operadores de Cruce Centrados en un Padre

Los OCCPs son operadores de cruce que producen descendientes en regiones cercanas al progenitor hembra, con mayor probabilidad que en cualquier otra

región del espacio de búsqueda. En particular, éstos determinan los genes de los descendientes extrayendo valores de intervalos definidos en la proximidad de los genes de la hembra, a través de distribuciones de probabilidad. Los rangos de la distribución de probabilidad utilizada dependerán de la distancia, en el espacio de decisión, entre los genes de la hembra y los del macho. Podemos encontrar bastantes ejemplos de OCCPs propuestos en la literatura (Ballester y Carter, 2004,?; Ballester y Richards, 2006; Deb y Agrawal, 1995; Deb y otros, 2002a; Lozano y otros, 2004; Takahashi y Kobayashi, 2001; Voigt y otros, 1995).

En la Sección 4.1.1, describimos un ejemplo de OCCP usado en este estudio. En la Sección 4.1.2, discutimos las ventajas de los OCCPs en el diseño de AGLs y explicamos por qué operan como operadores de mutación auto-adaptativos para el problema de la optimización continua.

4.1.1. El Operador de Cruce PBX-α

PBX-α (Lozano y otros, 2004) es una variante *centrada en un padre* del operador BLX-α (Eshelman y Schaffer, 1993). Sean $X = (x_1, \ldots, x_n)$ e $Y = (y_1, \ldots, y_n)$ $(x_i, y_i \in [a_i, b_i] \subseteq \Re,\ i = 1, \ldots, n)$ dos cromosomas con codificación real seleccionados para la aplicación del operador de cruce. PBX-α genera el descendiente $Z = (z_1, \ldots, z_n)$, donde z_i se escoge de forma aleatoria (uniformemente) del intervalo $[l_i, u_i]$ con $l_i = max\{a_i, x_i - I \cdot \alpha\}$, $u_i = min\{b_i, x_i + I \cdot \alpha\}$, $I = |x_i - y_i|$ y α es un parámetro asociado al operador.

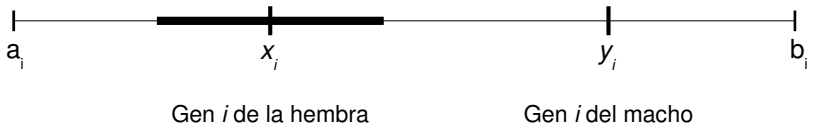


Figura 4.2: Efecto del operador de cruce PBX-α

Los efectos de este operador de cruce pueden observarse en la Figura 4.2:

- X es el progenitor hembra. Su papel es el de apuntar a las zonas del espacio de búsqueda donde se crearán descendientes ($[l_i, u_i]$).
- Y es el progenitor macho. Se utiliza para determinar la extensión de estas zonas, ya que el rango de la distribución de probabilidad depende de la distancia entre hembra y macho.

El grado de diversidad introducido por este operador puede ajustarse fácilmente variando el parámetro α. Cuanto más alto sea el valor de α, se genera mayor diversidad. Los valores típicos de α suele pertenecer al intervalo [0,5, 1].

4.1.2. Ventajas de los OCCPs

Los experimentos llevados a cabo por Deb y otros (2002a) han mostrado que los OCCPs son un mecanismo prometedor y eficiente a la hora de resol-

ver problemas de optimización continua. Nosotros pensamos que esos buenos resultados se deben a que los OCCPs combinan dos propiedades ventajosas:

- *Los OCCPs actúan como un operador de mutación.* Los OCCPs generan soluciones que son cercanas al progenitor hembra. De esta forma, pueden considerarse como un tipo especial de mutación. De hecho, es interesante resaltar que la mayoría de los modelos de AGCRs basados en OCCPs no usan un operador adicional de mutación (Ballester y Carter, 2003, 2004; Deb y otros, 2002a). Por otro lado, dos de las líneas más importantes de investigación en Algoritmos Evolutivos que usan codificación real ponen su atención en la mutación como operador principal para generar nuevas soluciones. Éstas son las *Estrategias de Evolución* (Beyer y Schwefel, 2002; Schwefel, 1995) y la *Programación Evolutiva* (Fogel, 1995). Éstas simulan la evolución como un proceso fenotípico, esto es, un proceso que enfatiza la relación entre el comportamiento de los padres y el de los descendientes, más que en su relación genética. De esta forma, el énfasis se sitúa en el uso del operador de mutación que genera un rango continuo de diversidad de compotamiento y mantiene una alta correlación entre el comportamiento de los padres y el de sus descendientes (Fogel, 1994).

  Deb (2005) adopta una idea similar para justificar el trabajo de los OCCPs: dado que cada padre lo selecciona cuidadosamente el mecanismo de selección, para la mayoría de los problemas de optimización continua se puede asumir que soluciones cercanas a esos padres serán, seguramente, tan buenas como sus padres. Al contrario, no se puede asumir que soluciones cercanas al centro de los padres sean tan buenas como sus padres. A partir de esta idea, podemos añadir un importante comentario adicional: la operación de los OCCPs puede ser particularmente prometedora cuando se aplica sobre individuos con altos valores de adecuación. Esto argumenta el hecho de que la mayoría de los AGCRs basados en OCCPs presentes en la literatura siguen el modelo *estacionario* (Sección 1.6.4), porque éstos mantienen unos niveles de presión selectiva mayores que los del modelo *generacional* (De Jong y Sarma, 1993).

- *Los OCCPs son operadores de cruce auto-adaptativos.* Los OCCPs definen la distribución de probabilidad de las soluciones descendientes de acuerdo a una medida de distancia entre las soluciones progenitoras. Si los padres se sitúan unos cerca de otros, los descendientes generados por el cruce se distribuyen densamente alrededor de la hembra. En otro caso, si los padres se sitúan unos lejos de otros, entonces los descendientes se distribuirán de forma esparcida. Por tanto, los OCCPs pueden ajustar su rango de acción dependiendo de la diversidad de la población, usando información específica contenida en los padres. De esta forma, dependiendo del nivel actual de diversidad en la población, los OCCPs pueden favorecer la producción de diversidad adicional (divergir) o el refinamiento de soluciones (converger). Este comportamiento se alcanza sin ningún mecanismo adaptativo externo.

  De hecho, se ha demostrado que los AGCRs con operadores de cruce con esta característica exhiben un comportamiento *auto-adaptativo* similar al observado en las Estrategias de Evolución y la Programación Evolutiva

(Deb y Beyer, 2001; Kita, 2001). Además, Beyer y Deb (2001) argumentan que un operador de variación que aprovecha la diferencia de los padres en el espacio de búsqueda es esencial para que el Algoritmo Evolutivo resultante exhiba un comportamiento auto-adaptativo a nivel de la población.

Para resumir, podemos conluir que los OCCPs pueden considerarse *operadores de mutación para codificación real auto-adaptativos*. También se han propuesto varias técnicas de mutación auto-adaptativas para las Estrategias de Evolución y la Programación Evolutiva (Bäck, 1996), sin embargo, existe una clara diferencia:

- Las Estrategias de Evolución y la Programación Evolutiva evolucionan los parámetros de estos operadores, como *desviaciones estándares*, simultáneamente con las variables de decisión.
- Los OCCPs calculan implícitamente la desviación estándar usando información de la distribución de los individuos de la población.

Finalmente, debemos destacar que, debido a que los OCCPs trabajan como operadores de mutación auto-adaptativos, éstos resultan adecuados para el diseño de AGLs. De hecho, dos propuestas de AGLs para el problema de la optimización continua (Lozano y otros, 2004; Noman y Iba, 2008) usan este tipo de operadores.

4.2. Mecanismo de Selección de la Hembra: Selección Uniforme en Fertilidad

Muchas de las técnicas de reemplazo presentadas para los AGEs (véase la Sección 1.6.4) pueden introducir el siguiente inconveniente: *algunos individuos pueden permanecer en la población durante mucho tiempo*. Esta situación puede causar que algunas zonas del espacio de búsqueda se exploten excesivamente, a expensas de ignorar otras áreas que pueden ser prometedoras. Una técnica para evitar el riesgo derivado de este problema, consiste en limitar el número de hijos que reciben los cromosomas durante su estancia en la población. En la literatura, encontramos diferentes esquemas de selección que implementan esta idea (Branke y otros, 1999; De Jong y Sarma, 1993; Ghosh y otros, 1998).

En esta sección, presentamos un nuevo mecanismo de selección de padres, llamado *selección uniforme en fertilidad* (SUF), que regula, siguiendo esta idea, el número de hijos que cada cromosoma produce. SUF almacena el número de veces que cada cromosoma de la población (C_i) se selecciona como hembra ($n_H(C_i)$) y escoge como tal el cromosoma con el valor $n_H(\cdot)$ más bajo. El seudocódigo de SUF se muestra en la Figura 4.3.

SUF se ha diseñado para favorecer la diversidad únicamente, ya que no muestra ninguna preferencia sobre los individuos mejor adaptados. Esto implica que debe combinarse con una estrategia de reemplazo con presión selectiva hacia estos individuos. En particular, proponemos aplicar SUF junto a la estrategia

```
C_min ← buscar cromosoma tal que n_H(·) sea mínimo;
n_H(C_min) ← n_H(C_min) + 1;
devolver C_min como hembra seleccionada;
```

Figura 4.3: Seudocódigo de SUF

reemplazar el peor (RP) (Sección 1.6.4), que introduce una presión selectiva alta, ya que mantiene en la población los mejores individuos encontrados hasta el momento. De esta forma, todos los individuos de la población representan zonas prometedoras del espacio de búsqueda, que merecen que PBX-α intensifique. Esto justifica la forma en la que opera SUF: induce una búsqueda extensa, proporcionando a todos los cromosomas de la población las mismas oportunidades de seleccionarse como hembra.

Mediante la combinación de SUF y RP (SUF&RP), juntamos un esquema de selección diversificador, que favorece la producción de descendientes en zonas muy diferentes del espacio de búsqueda, con un método de reemplazo que introduce una alta presión selectiva. También otros autores han sugerido esquemas de AGs que conectan técnicas muy diversificadoras y técnicas con alta explotación con la intención de obtener una búsqueda efectiva. Por ejemplo, Shimodaira (1996) propone un algoritmo que aplica mutaciones con una alta probabilidad y un mecanismo de selección elitista sobre la población. Eshelman (1991) presenta un AG que combina un operador de cruce muy perturbador con un mecanismo de selección conservador. Finalmente, van Kemenade y otros (1995) sugieren que presiones selectivas altas permiten aplicar operadores de cruce más perturbadores.

En la Sección 4.2.1, realizamos una comparación empírica del rendimiento de la combinación SUF&RP, comparándola con otras posibles combinaciones de mecanismos de selección y estrategias de reemplazo. En la Sección 4.2.2, investigamos el comportamiento de SUF y su influencia sobre el número de descendientes que reciben los cromosomas que visitan la población.

4.2.1. Estudio de la Combinación SUF&RP

La efectividad final de un AGE viene determinada por el equilibrio entre intensificación y diversificación derivado de la combinación del mecanismo de selección de padres y la estrategia de reemplazo que aplica. El objetivo de esta sección, es comprobar si el equilibrio mantenido por la combinación SUF&RP produce efectos beneficiosos sobre la operación del AGE. Para ello, vamos a comparar empíricamente esta combinación frente a otras combinaciones de mecanismos de selección y estrategias de reemplazo propuestas en la literatura. Concretamente, implementaremos todas las combinaciones de mecanismos y estrategias para AGEs descritas en las Secciones 1.6.4 y 1.6.4, respectivamente. La Tabla 4.1 contiene las posibles combinaciones de estos métodos. Hemos usado los siguientes acrónimos:

Tabla 4.1: Combinaciones de mecanismos de selección y reemplazo estudiadas

		Estrategia de Reemplazo		
		Pres. Sel.	**Div.**	**Div. & Pres. Sel.**
Selección	**Pres. Sel. (ST)**	ST&RP	ST&FIFO	ST&STR
de	**Div. (SA)**	SA&RP	-	SA&STR
padres	**Div. &**	SE&RP	SE&FIFO	SE&STR
	Pres. Sel.	SUA&RP	SUA&FIFO	SUA&STR
	(SE, SUA, SOD)	SOD&RP	SOD&FIFO	SOD&STR

- ST: selección por torneo.
- SA: selección aleatoria.
- SE: selección de extremos.
- SUA: selección uniforme en aptitud.
- SOD: selección orientada a la diversidad.
- RP: reemplazar el peor.
- FIFO: primero en entrar, primero en salir (en inglés *first-in-first-out*).
- STR: selección por torneo restringido.

El parámetro ω de STR se asignó a 5 y n_T, en ST, a 2.

Los ocho tipos de combinaciones de la Tabla 4.1 nos permiten analizar los efectos derivados de la unión de diferentes alternativas para manejar diversidad y presión selectiva. No hemos considerado la combinación de mecanismos de selección y estrategias de reemplazo que propician únicamente diversidad, ya que convierten el AG en un procedimiento de búsqueda aleatoria (la presión selectiva es crítica para asegurar el progreso efectivo en el proceso de optimización).

Hemos llevado a cabo experimentos de minimización sobre un conjunto de seis problemas representativos de los descritos en el Apéndice A.2. Consideramos este conjunto reducido de problemas de prueba para facilitar el análisis de SUF:

- Tres funciones unimodales: f_{esf}, f_{Ros} y f_{Sch}.
- Dos funciones multimodales: f_{Ras}, f_{Gri}.
- Un problema real complejo: P_{sle}.

Hemos implementado diferentes AGEs que se distinguen únicamente en el mecanismo de selección y la estrategia de reemplazo. Utilizan codificación real y aplican el operador PBX-α con un valor fijo para α ($\alpha = 1$). No hacen uso de operador de mutación. Los mecanismos de selección considerados se utilizan

para determinar el progenitor hembra, mientras que el progenitor macho se escoge de forma aleatoria. El tamaño de la población es de 60 cromosomas. Todos los algoritmos se han ejecutado 50 veces con 100.000 evaluaciones para cada ejecución.

Tabla 4.2: Resultados de los AGEs comparados

Algoritmos	f_{esf}	f_{Ros}	f_{Sch}	f_{Ras}	f_{Gri}	P_{sle}	+	∼	−
ST&RP	4,11e−61−	1,84e+01+	3,18e−04+	9,65e+01+	1,02e−02+	2,48e+02+	5	0	1
ST&FIFO	6,54e+00+	1,30e+03+	4,34e+03+	1,64e+02+	4,83e+00+	4,31e+02+	6	0	0
ST&STR	1,77e−26+	1,87e+01+	1,91e+00+	4,76e+01+	2,32e+02+	8,74e+01−	5	0	1
SA&RP	1,04e−39+	1,96e+01+	7,22e−02+	5,30e+01+	1,11e−02+	1,40e+02∼	5	1	0
SA&STR	2,48e−15+	1,94e+01+	5,71e+01+	3,48e+01+	1,53e−02+	7,72e+01−	5	0	1
SE&RP	4,23e−55+	1,90e+01+	8,62e−04+	9,10e+01+	1,74e−02+	1,95e+02+	6	0	0
SE&FIFO	1,18e+01+	2,45e+03+	4,91e+03+	1,82e+02+	9,47e+00+	5,56e+02+	6	0	0
SE&STR	2,19e−22+	2,21e+01+	1,42e+01+	3,32e+01+	1,42e−02+	9,48e+01∼	5	1	0
SUA&RP	1,42e−75−	1,58e+01∼	7,78e−06∼	2,05e+02+	2,98e−02+	3,67e+02+	3	2	1
SUA&FIFO	8,35e+00+	3,53e+03+	9,93e+03+	1,26e+02+	9,57e+00+	6,51e+02+	6	0	0
SUA&STR	2,41e−08+	2,06e+01+	6,67e+02+	4,76e+01+	1,64e−02+	9,65e+02∼	5	1	0
SOD&RP	4,92e−43+	1,79e+01+	7,45e−02+	7,62e+01+	3,02e−02+	3,25e+02+	6	0	0
SOD&FIFO	2,75e+01+	1,27e+04+	1,07e+04+	2,09e+02+	3,16e+01+	1,09e+03+	6	0	0
SOD&STR	6,43e−16+	2,72e+01+	9,88e+01+	5,50e+01+	3,40e−02+	1,71e+02+	6	0	0
SUF&RP	2,86e−56	1,56e+01	8,90e−08	2,64e+01	3,20e−03	1,20e+02			

La Tabla 4.2 muestra los resultados obtenidos. La medida de eficacia utilizada es la media del mejor valor de la función objetivo encontrado al final de cada ejecución. Además, se ha aplicado un test de *Student* (a nivel 0,05 de significación) para comprobar si existen diferencias significativas entre la efectividad de SUF&RP y la de los otros algoritmos. Notamos la dirección de las diferencias significativas de la siguiente forma:

- El signo más (+): la efectividad de SUF&RP es mejor que la del algoritmo correspondiente.
- El signo menos (−): el algoritmo correspondiente mejora a SUF&RP.
- El signo aproximado (∼): no se detectaron diferencias.

Los lugares donde no aparecen estos signos corresponden con los resultados de SUF&RP. En la Tabla 4.2 hemos incluido las tres columnas finales para facilitar el análisis de los resultados. Éstas contienen el número de veces que SUF&RP produce resultados mejores, peores o sin diferencias significativas frente a cada uno de los algoritmos alternativos (con respecto al test de *Student*).

Podemos observar que la combinación SUF&RP mejora consistentemente a todos los demás algoritmos. SUF&RP proporciona los mejores resultados en cuatro de los seis problemas de prueba. También, obtiene muchas mejoras y pocas pérdidas en efectividad. Además, el número de equivalencias (diferencias no detectadas) es también bajo. Estos resultados revelan que SUF y RP interactúan adecuadamente para permitir que se alcancen las mejores soluciones. Con

la aplicación de RP, la población obtiene información de la localización de las áreas más prometedoras, y con SUF, se lleva a cabo una exploración exhaustiva de ellas.

A continuación, examinamos las características de las combinaciones que obtienen las mejores soluciones. La Tabla 4.3 muestra los cinco algoritmos que obtuvieron los mejores resultados para cada problema.

Tabla 4.3: Algoritmos que obtienen los mejores resultados

f_{esf}	f_{Ros}	f_{Sch}	f_{Ras}	f_{Gri}	P_{sle}
SUA&RP	SUF&RP	SUF&RP	SUF&RP	SUF&RP	SA&STR
ST&RP	SUA&RP	SUA&RP	SE&STR	ST&RP	ST&STR
SUF&RP	SOD&RP	ST&RP	SA&STR	SA&RP	SE&STR
SE&RP	ST&RP	SE&RP	ST&STR	SE&STR	SUA&STR
SOD&RP	ST&STR	SA&RP	SUA&STR	SA&STR	SUF&RP

Una inspección de la Tabla 4.3 nos permite realizar los siguientes comentarios:

- La combinación SUF&RP es la más robusta. De hecho, obtiene los mejores resultados para cuatro de los seis problemas de prueba.
- En general, la estrategia de reemplazo RP es muy exitosa. La mayoría de los mejores algoritmos utilizan esta estrategia.
- La estrategia de reemplazo STR es competitiva en los problemas complejos, f_{Ras}, f_{Gri} y P_{sle}. Esto se debe a su habilidad para mantener diversidad en la población (es un método de agrupamiento). En particular, la unión de STR y SA (mecanismo de selección que favorece sólo la diversificación) produce un comportamiento robusto para esos problemas. Esta combinación ya la sugirió Harik (1995).
- SUA es parte de dos combinaciones remarcables: SUA&RP y SUA&STR. La primera obtiene buenos resultados en los problemas unimodales. La segunda aparece entre los mejores algoritmos para dos de los problemas complejos: f_{Ras} y P_{sle}. SUA da preferencia a individuos con valores de aptitud poco presentes en la población. De esta forma, SUA comparte con SUF, el ánimo de ofrecer un muestreo uniforme de las áreas de búsqueda representadas en la población. Los buenos resultados de SUA y SUF nos permiten concluir que esta idea representa un camino importante para mejorar el rendimiento de los AGEs.
- La combinación ST&RP es ventajosa en los problemas unimodales: f_{esf}, f_{Ros} y f_{Sch}. Sin embargo, ésta no aparece entre las mejores combinaciones para problemas complejos. Esto es razonable, porque ambos, ST y RP, sólo producen presión selectiva, lo cual es beneficioso en problemas unimodales y fatal en complejos.

4.2.2. Efectos de SUF

En esta sección, pretendemos descubrir las características del comportamiento de SUF que afectan beneficiosamente al uso de PBX-α. Para ello, introducimos la Figura 4.4, que compara SUF y los mecanismos de selección usados en la Sección 4.2.1. Ésta muestra el porcentaje de cromosomas que, durante su vida, no se seleccionaron como progenitor hembra (no produjeron descendientes) o se seleccionaron una vez (1 descendiente), dos veces y así en adelante, al resolver el problema f_{esf}. Se utilizó el reemplazo RP en todos los algoritmos.

Podemos destacar los siguientes hechos:

- Con SUF, la mayoría de los cromosomas que pasan por la población han producido dos o tres hijos. Por ello, se exploran equitativamente todas las zonas representadas en la población. Sin embargo, con las otras técnicas de selección, un considerable porcentaje de cromosomas no genera descendientes durante su estancia en la población. Esto indica que ciertas regiones de búsqueda que merecen la atención no se muestrean con el suficiente interés, o caen en el olvido. Esto puede ser fatal para los problemas complejos.

- ST, SUA, y SOD hacen que pocos cromosomas produzcan muchos descendientes y que la mayoría produzcan pocos descendientes. Esto se debe, primero, a que el método de selección muestra una alta tendencia a seleccionar los mejores individuos de la población como hembras, y segundo, al uso de la estrategia de reemplazo RP, la cual mantiene en la población los mejores individuos, dándoles la oportunidad de producir descendientes durante largos periodos de tiempo.

- SA y SE reducen el desequilibrio de los métodos de selección anteriores. Por un lado, SA elimina la predisposición a seleccionar los individuos más adaptados. Por otro lado, SE favorece la selección de ambos, los mejores y los peores.

4.3. Emparejamiento Variado Negativo

En esta sección, realizamos el estudio de la selección del progenitor macho. Esta tarea se puede lleva a cabo por mecanismos de *emparejamiento*, los cuales determinan la forma en que los cromosomas se emparejan para aplicarles el operador de cruce. En los AGs convencionales, no se aplican estrategias de emparejamiento; esto es, los padres se combinan sin realizar ningún examen después de que se hubiesen elegido, aleatoriamente o sólo por su valor de aptitud. El emparejamiento en la naturaleza es más complicado. Inspirados en esta observación, en la literatura se han propuesto varias estrategias de emparejamiento para AGs que tratan con los temas de diversidad en la población o presión selectiva de una forma natural:

- *Prohibición de cruces según genealogía* (Craighurst y Martin, 1995). En esta técnica, no se permite el cruce entre individuos con un cierto grado

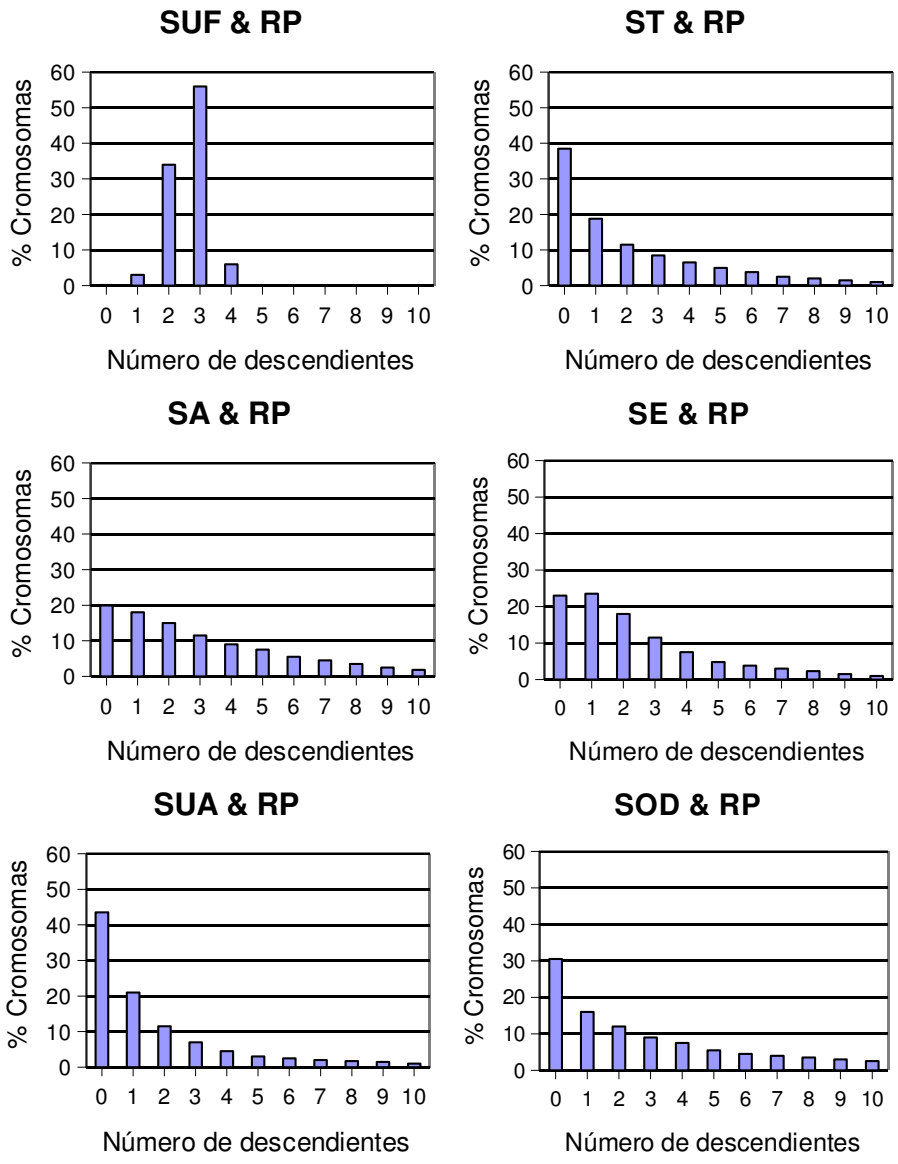


Figura 4.4: Distribuciones del número de descendientes por hembra

de parentesco. Precisamente, la naturaleza fomenta la diversidad en la población de esta manera. Se definen varios niveles de prevención de incesto, los cuales definen hasta donde, en el árbol genealógico, se deben evitar cruces entre individuos del AG relacionados. Los autores denominan estos niveles como leyes de incesto. Un ejemplo es la *segunda ley de incesto*: un cromosoma no puede emparejarse consigo mismo, sus padres, sus hijos (porque es el padre de ellos), o sus hermanos (porque sus padres son los mismos).

- *Prohibición de incesto* (Eshelman y Schaffer, 1991). Ésta sugiere que los individuos con una distancia *Hamming* pequeña podrían ser parientes. Por tanto, el emparejamiento sólo se permite si la distancia *Hamming* se encuentra por encima de un determinado umbral, el cual decrece a medida que el proceso evoluciona.
- *Diferenciación de cromosomas* (Bandyopadhyay y otros, 1998). Los cromosomas de la población se distinguen en dos categorías según el valor contenido en uno de sus *bits*. Inicialmente, se asigna el valor al *bit* diferenciador según la distancia *Hamming* máxima entre los cromosomas. El cruce se realiza sólo entre individuos pertenecientes a diferente categoría.
- *Emparejamiento variado* (Fernandes y Rosa, 2001). Se trata del procedimiento descrito en la Sección 3.1.2, el cual empareja individuos con genotipo similar con mayor o menor probabilidad que la esperada por un método aleatorio. Al método con mayor probabilidad de emparejar individuos con genotipo similar se le llama *emparejamiento variado positivo* y al método con menor probabilidad, *emparejamiento variado negativo* (EVN).

  Matsui (1999) propone un método similar que se llama *selección por torneo correlativo*, el cual escoge la pareja con mayor valor de aptitud y distancia *Hamming* de un conjunto de candidatos.

Hemos elegido EVN como técnica para escoger el progenitor macho por dos razones:

- EVN fuerza la creación de diversidad de manera directa. La mayoría de los mecanismos de emparejamiento presentados en la literatura (Bandyopadhyay y otros, 1998; Craighurst y Martin, 1995; Eshelman y Schaffer, 1991) son realmente *estrategias de decisión*, porque determinan si dos padres, ya elegidos, pueden cruzar o no. La idea es esperar que una pareja de padres adecuada, que cumple una determinada condición, llegue en cualquier momento. Sin embargo, EVN actúa directamente: encuentra la pareja de padres que realmente promoverá la generación de descendientes diferentes.
- En un AGE con el reemplazo RP, la sucesiva aplicación de EVN y PBX-α puede producir una *diversificación auto-adaptativa* que favorezca la producción de *diversidad útil* (esto es, diversidad en la población que, de alguna forma, ayude a producir buenas soluciones) (Mahfoud, 1995).

En la Sección 4.3.1, explicamos la segunda razón con más profundidad y en la Sección 4.3.2, comparamos el rendimiento de EVN con el del emparejamiento aleatorio, el cual se ha aplicado en los experimentos previos.

4.3.1. Diversificación Autoadaptativa por EVN y PBX-α

Un AGE con la estrategia de reemplazo RP mantiene una población con los mejores elementos que han aparecido hasta el momento (*población élite*). Hay dos tipos importantes de información en una población élite que se pueden explotar para generar nuevos elementos: 1) dónde se encuentra los mejores elementos y 2) cómo se distribuyen estos elementos (juntos, dispersos, etc.). Precisamente, algunos operadores de cruce para codificación real obtienen su habilidad auto-adaptativa por el uso del segundo tipo de información.

EVN maneja el segundo tipo de información también, con la intención de influenciar la extensión sobre la que PBX-α genera los descendientes. En particular, EVN extrae información de la población élite: cómo de lejos se distribuyen los individuos prometedores unos de otros. Entonces, PBX-α puede explotar este tipo de información con dos intenciones:

- Usar las áreas más amplias para la generación de descendientes. La idea es crear descendientes lo más distantes posible de sus padres, para contribuir a la diversidad de la población.
- Producir descendientes con alto valor de aptitud. Debido a que usamos información de la distribución de los mejores individuos para generar nuevos cromosomas, esperamos que éstos muestren también altos valores de adecuación.

De esta forma, la sucesiva aplicación de EVN y PBX-α permite alcanzar los dos objetivos de un AG simultáneamente: *obtener soluciones de gran calidad y elevar la diversidad en la población, esto es, promover diversidad útil.* Además, debemos destacar que dado que los niveles de diversidad provistos por EVN y PBX-α dependen directamente de la distribución de los individuos de la población, podemos decir que éstos llevan a cabo una diversificación auto-adaptativa.

4.3.2. Análisis Empírico de EVN

En esta sección, analizamos cómo afecta EVN al rendimiento de un AGE que aplica el reemplazo RP. Además, estamos particularmente interesados en medir la influencia del parámetro α, asociado a PBX-α, sobre la acción de EVN, porque este parámetro determina la extensión asociada a la distribución de probabilidad usada por el operador para crear el descendiente.

La Tabla 4.4 compara el rendimiento de algoritmos que aplican el emparejamiento aleatorio para la selección del progenitor macho con otros que utilizan EVN. Se han probado varios valores de α (0,7, 0,8, 0,9 y 1,0). Hemos experimentado con otros valores para este parámetro; sin embargo, ofrecían resultados pobres y se descartaron. Todos los algoritmos utilizan SUF para seleccionar el progenitor hembra y la estrategia de reemplazo RP. A los primeros algoritmos los hemos llamado EA-α (Emparejamiento Aleatorio) y a los segundos, EVN-α. Las funciones de prueba y condiciones de ejecución son las mismas utilizadas en los experimentos previos. Indicamos, en negrita, el resultado que, de acuerdo al test de *Student*, es significativamente mejor. Si no se detectan diferencias entre los algoritmos correspondientes, ninguno de los dos aparecerá remarcado.

Tabla 4.4: Resultados comparando EA-α y EVN-α

Algoritmos	f_{esf}	f_{Ros}	f_{Sch}	f_{Ras}	f_{Gri}	P_{sle}
EA-0,7	1,97e−66	2,24e+1	5,41e−03	1,65e+2	2,51e−2	6,60e+2
EVN-0,7	**9,88e−95**	1,91e+1	**7,97e−11**	**7,49e+1**	**5,17e−3**	**1,89e+2**
EA-0,8	9,89e−80	1,98e+1	5,29e−08	8,09e+1	4,58e−3	3,39e+2
EVN-0,8	1,44e−81	**1,47e+1**	6,05e−08	**3,08e+1**	6,84e−3	**7,14e+1**
EA-0,9	**4,35e−71**	1,56e+1	**5,87e−09**	4,08e+1	**4,04e−3**	2,07e+2
EVN-0,9	3,67e−60	**1,42e+1**	2,68e−03	**1,45e+1**	8,81e−3	**3,50e+1**
EA-1.0	**2,86e−56**	1,56e+1	**8,90e−08**	2,64e+1	**3,20e−3**	1,20e+2
EVN-1,0	2,10e−41	**1,48e+1**	3,28e+00	**1,04e+1**	1,00e−2	**2,25e+1**

Claramente, nos damos cuenta de que, en general, con $\alpha = 0{,}7$ y $\alpha = 0{,}8$, EVN obtienen mejoras significativas en la mayoría de los problemas, con respecto a su homólogo con EA. Con $\alpha = 0{,}9$ y $\alpha = 1{,}0$, también obtiene los mejores resultados para f_{Ros}, el problema multimodal f_{Ras}, y el problema real complejo P_{sle}. Estos hechos sugieren que EVN es una herramienta prometedora para aumentar la diversificación, capaz de mejorar el rendimiento de los AGEs. Esta afirmación puede verse reforzada en la Tabla 4.5, la cual destaca los cinco algoritmos que devolvieron los mejores resultados para cada problema. De esta tabla podemos destacar que:

- Para la mayoría de los problemas, el algoritmo que mejor funciona utiliza EVN.
- En general, la mejor solución para cada problema se alcanza con EVN, usando diferentes valores de α. Para los problemas multimodales, f_{Ras} y f_{Gri}, y el complejo P_{sle}, el valor más efectivo es $\alpha = 1{,}0$, mientras que para los problemas unimodales, los valores menores son más beneficiosos. Esto indica que α tiene una importante influencia sobre la efectividad de EVN.
- A pesar del hecho anterior, podemos considerar que el algoritmo que usa EVN y $\alpha = 0{,}8$ (EVN-0,8) (en negrita en la Tabla 4.5), consigue una robustez aceptable con respecto a los otros algoritmos, porque aparece entre los mejores algoritmos para todos los problemas considerados en nuestra experimentación. Ninguno del resto permite obtener un rendimiento mejor.

Otra observación interesante de la Tabla 4.4, es que los resultados de EVN con $\alpha = 0{,}9$ y $\alpha = 1{,}0$ son peores que los de los algoritmos correspondientes con EA, para los problemas unimodales f_{esf} y f_{Sch}, y para el multimodal menos complejo, esto es, f_{Gri}. La aplicación de EVN con altos valores de α hace que PBX-α use distribuciones de probabilidades demasiado extensas (ver Figura 4.2). Esto puede ser adecuado en problemas complejos donde la diversidad puede ayudar a alcanzar las zonas de búsqueda prometedoras (como hemos observado, $\alpha = 1{,}0$ es la mejor elección para estos problemas); sin embargo, esta

Tabla 4.5: Algoritmos que obtienen los mejores resultados

f_{esf}	f_{Ros}	f_{Sch}	f_{Ras}	f_{Gri}	P_{sle}
EVN-0,7	EVN-0,9	EVN-0,7	EVN-1,0	EA-1,0	EVN-1,0
EVN-0,8	**EVN-0,8**	EA-0,9	EVN-0,9	EA-0,9	EVN-0,9
EA-0,8	EVN-1,0	EA-0,8	EA-1,0	EA-0,8	**EVN-0,8**
EA-0,9	EA-0,9	**EVN-0,8**	**EVN-0,8**	EVN-0,7	EA-1,0
EA-0,7	EA-1,0	EA-1,0	EA-0,9	**EVN-0,8**	EVN-0,7

excesiva cantidad de exploración se vuelve perjudicial para problemas con las características de las funciones de prueba antes mencionadas.

4.4. Procedimiento de Diferenciación Sexual

Hasta este momento, cualquier individuo de la población podía utilizarse como progenitor hembra o progenitor macho. Sin embargo, como se ha visto, cada uno tiene una misión diferente: el progenitor hembra centra la zona de muestreo del OCCP, mientras que el progenitor macho determina la extensión de esta zona. Por tanto, pensamos que la diferenciación sexual de los cromosomas en dos grupos (G_H y G_M) puede producir efectos importantes en el uso de los OCCPs.

En esta sección, presentamos un *Procedimiento de Diferenciación Sexual* que determina qué individuos de la población actual pueden convertirse en: sólo progenitores hembra, sólo progenitores macho, o tanto progenitores hembra como macho. La inclusión de este proceso es directa. Se debe ejecutar antes de la aplicación de los mecanismos de selección de padres (mecanismo de selección de hembra y mecanismo de selección de macho), los cuales se aplicarán en sus correspondientes grupos. El resto del algoritmo se ejecutará de la manera usual. De esta forma, PDS podría considerarse como un mecanismo de preselección.

PDS utiliza dos parámetros, N_H y N_M, con $N_H \leq N$ y $N_M \leq N$ (N es el tamaño de la población) y produce los dos grupos, G_H y G_M, como sigue (véase la Figura 4.5 donde se asume que los individuos de la población están ordenados de acuerdo a sus valores de aptitud):

- G_H se compone de los N_H mejores individuos de la población, y
- G_M, de los N_M mejores individuos de la población.

Además, se cumple que $N_H = N$ o bien $N_M = N$. A continuación, damos dos observaciones derivadas de esta definición:

1. En el caso de $N_H = N_M$, no hay PDS, de forma que cualquier miembro de la población se puede seleccionar como progenitor hembra o macho, esto es, el OCCP se utiliza de la forma estándar.

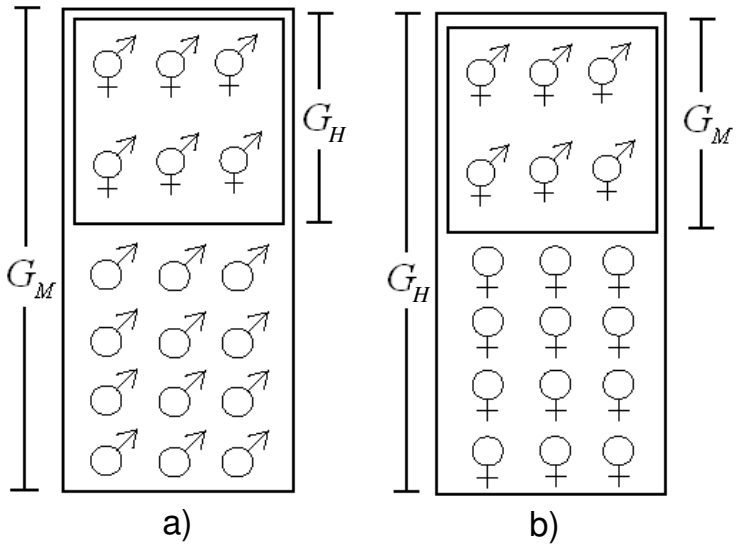


Figura 4.5: Procedimiento de Diferenciación Sexual: a) $N_H < N_M$ y b) $N_H > N_M$

2. $G_H \cap G_M \neq \emptyset$, lo que significa que algunos individuos pueden asumir el papel de progenitores tanto hembra como macho (ver Figura 4.5). En particular, se ha considerado que los N_{min}, con $N_{min} = min\{N_H, N_M\}$, mejores individuos de la población tendrán buenas características para utilizarse con ambos papeles.

Otra característica importante de PDS es que introduce presión selectiva (intensificación) en los procedimientos de selección de los progenitores hembra y macho que se aplican a continuación. Además, debemos remarcar que:

- El impacto del parámetro N_H sobre la presión selectiva producida es simple y predecible.
- El rango de presión selectiva que se puede conseguir variando N_H es muy amplio.

Éstas son dos características deseables de un proceso de selección (Bäck, 1994). Por un lado, cuando N_H es bajo, se produce una alta presión selectiva que fuerza al proceso de búsqueda a enfocarse en las mejores regiones. Por otro lado, si N_H es bajo, la presión selectiva es más suave, proveyendo más regiones de muestreo, representadas por la población actual.

La Figura 4.6 muestra nuestro modelo de AGE basado en OCCPs, que llamaremos PDS-S&E. Inicialmente, se ejecuta PDS para obtener los grupos G_H y G_M a partir de la población. Después, SUF selecciona el progenitor hembra de entre los individuos en G_H, y EVN escoge el progenitor macho de entre los miembros de G_M. El resto de pasos se ejecutan de la forma usual.

```
1. Construir G_H y G_M aplicando PDS;
2. Seleccionar una hembra de G_H mediante SUF;
3. Seleccionar un macho de G_M mediante EVN;
4. Crear un descendiente aplicando PBX-α a los padres;
5. Evaluar el descendiente con la función de prueba;
6. Introducir el descendiente en la población usando RP;
```

Figura 4.6: Seudocódigo de una iteración de PDS-S&E

En este punto, reconocemos que la idea de incorporar diferenciación en los cromosomas de un AG no es nueva. Otros autores (Bandyopadhyay y otros, 1998; Goh y otros, 2003) añaden un procedimiento de diferenciación sexual para investigar nuevos modelos o producir una clara y equilibrada separación entre intensificación y diversificación. La principal diferencia con el modelo propuesto en este capítulo, reside en tener en cuenta los diferentes papeles que juegan los progenitores en la aplicación de OCCPs.

A continuación, investigamos la influencia de los parámetros asociados a PDS (N_H y N_M) en el rendimiento del algoritmo PDS-S&E. Hemos llevado a cabo experimentos con este algoritmo en los problemas utilizados en las Secciones 4.2.1 y 4.3.2, considerando diferentes valores para N_H y N_M (N_H=1, 5, 25, 50, 100, 200, 300 y 400 individuos, y $N_M = 25, 50, 100, 200, 300$, y 400 individuos). Se probaron todas las posibles combinaciones de estos valores. Los algoritmos se ejecutaron 50 veces, cada una con un máximo de 100.000 evaluaciones. La medida de rendimiento usada es la media del mejor valor de aptitud encontrado al final de cada ejecución. La Tabla B.8 (Apéndice B.2.1) muestra los resultados obtenidos.

Ahora, examinamos las características de las combinaciones de valores para N_H y N_M que alcanzan las mejores soluciones. La Tabla 4.6 muestra las quince combinaciones de valores que devolvieron los mejores resultados para cada problema de prueba de forma ordenada. Se han resaltado, en negrita, las mejores combinaciones donde $N_H = N_M$, esto es, las combinaciones que representan el mejor AGE que no usa PDS.

Para cada problema de prueba, la combinación que consigue los mejores resultados cumple que $N_H \neq N_M$, esto es, utiliza PDS. Hemos aplicado un test de *Student* para saber si las diferencias entre los rendimientos de la mejor combinación con $N_H \neq N_M$ y la mejor combinación con $N_H = N_M$ son significativas o no. La Tabla 4.7 muestra los resultados (hemos introducido el signo '*' cuando el test rechaza la hipótesis nula, esto es, hay diferencias significativas). Para la mayoría de las funciones, podemos ver que realmente existen diferencias significativas, esto es, PDS mejora el rendimiento del AGE que utiliza PBX-α. Sólo para f_{Gri}, no se rechazó la hipótesis nula.

Otra observación importante, de la Tabla 4.6, es la gran diferencia entre los mejores valores de N_H y N_M para cada problema. En particular, podemos remarcar la gran diferencia entre los mejores valores de N_H para los problemas

Tabla 4.6: Mejores combinaciones de valores de N_H y N_M

f_{esf}	f_{Ros}	f_{Sch}	f_{Ras}	f_{Gri}	P_{sle}
5, 100	5, 100	25, 50	400,100	400, 300	200, 400
5, 50	5, 200	**50**, **50**	300,100	200, 400	300, 400
1, 200	5, 300	5, 100	**200**,**200**	**300**, **300**	**400**, **400**
25, 50	25, 100	100, 25	400,50	300, 400	50, 400
5, 200	25, 200	25, 100	100,400	400, 200	400, 300
1, 300	25, 300	5, 200	200,100	100, 300	300, 300
25, 100	5, 400	200, 25	100,300	300, 200	400, 200
5, 300	25, 400	100, 50	100,200	200, 300	100, 400
1, 400	25, 50	5, 300	300,200	100, 400	200, 300
50, **50**	50, 100	50, 100	200,300	300, 100	300, 200
100, 25	50, 200	300, 25	300,50	50, 400	100, 300
5, 400	50, 300	25, 200	50,400	200, 200	50, 300
200, 25	**50**, **50**	5, 400	200,50	100, 200	25, 400
50, 25	50, 400	400, 25	400,25	400, 400	200, 200
25, 200	100, 50	200, 50	50,300	25, 400	100, 200

Tabla 4.7: $N_H \neq N_M$ frente $N_H = N_M$

Problema de Prueba	N_H, N_M	**Media de la Mejor Aptitud**
f_{esf}	5, 100 *	9,98e−187
	50, 50	6,19e−095
f_{Ros}	5, 100 *	1,56e+000
	50, 50	1,46e+001
f_{Sch}	25, 50 *	1,01e−012
	50, 50	2,54e−010
f_{Ras}	400, 100 *	2,60e+000
	200, 200	3,58e+000
f_{Gri}	400, 300	3,48e−004
	300, 300	5,42e−004
P_{sle}	200, 400 *	5,45e+000
	400, 400	7,84e+000

complejos y multimodales (f_{Ras}, f_{Gri}) y para los problemas unimodales (f_{esf}, f_{Ros} y f_{Sch}):

- Valores bajos de N_H (por ejemplo, 5 o 25 individuos) producen una alta intensificación, la cual es muy interesante para obtener *precisión* en problemas unimodales. Esto es, los algoritmos PDS-S&E con valores bajos de N_H son AGLs para optimización continua. Sin embargo, este no es el único aspecto determinante para alcanzar tal éxito en este tipo de problemas (y en particular, en nuestro caso, ya que seguimos una *inicialización desplazada*, esto es, lejana del óptimo global, descrita en la Sección A.2.1). Usar valores altos del parámetro N_M (por ejemplo, 50 o 100 individuos) amplía las zonas de muestreo. Esto produce una sinergia con la presión selectiva, asegurando que el algoritmo puede progresar a mejores zonas.
- El uso de valores altos para N_H, junto con la aplicación de SUF, induce una búsqueda dispersa, debido a que diferentes progenitores hembra se convierten en el centro de atención del OCCP. Esta alta exploración del espacio de búsqueda es esencial para obtener resultados fiables en problemas multimodales y complejos. Además, la aplicación de valores muy altos para el parámetro N_M (por ejemplo, 400 individuos) refuerza esta habilidad exploratoria, induciendo un rendimiento prometedor en los problemas más complejos (por ejemplo P_{sle}).

Por ello, podemos concluir que PDS nos permite especializar el AGE en intensificación, usando valores bajos para el parámetro N_H, y así obtener buenos resultados en problemas unimodales, o especializarlo en diversificación para obtener buenos resultados en problemas multimodales y complejos, usando valores más altos para el mismo parámetro.

4.5. AGCRs Especializados: AGCRs Locales y AGCRs Globales

A partir de los resultados del estudio anterior, observamos que PDS hace que los AGCRs puedan usar los OCCPs de forma más efectiva, ya que permite:

- ajustar, variando N_H, el nivel de presión selectiva adecuado, de forma precisa y dentro de un amplio rango posible,
- y generar unos niveles de exploración adecuados, controlando N_M.

Esta característica hace que podamos diseñar dos clases diferentes de *AGCRs especializados* (ver Figura 4.7):

- *AGCRs Locales*: son AGCRs especializados en aportar intensificación, siendo capaces de alcanzar soluciones muy *precisas* dentro de una región del espacio de búsqueda. Esta propiedad es muy interesante cuando se tratan problemas unimodales. De los experimentos realizados, observamos que los AGCRs Locales suelen alcanzar el óptimo global con un error

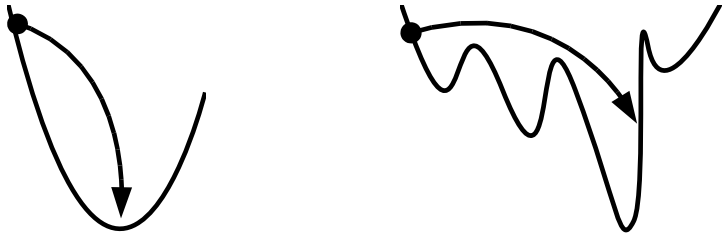


Figura 4.7: AGCRs especializados: a) AGCRs Locales, y b) AGCRs Globales

muy reducido. Un ejemplo puede ser PDS-S&E con $N_H = 5$ y $N_M = 100$ (ver Tabla 4.6).

- *AGCRs Globales*: son AGCRs especializados en realizar una exploración amplia del espacio de búsqueda, ofreciendo soluciones muy *fiables*. Esto es interesante cuando nos enfrentamos a problemas con muchos óptimos locales o con alto nivel de complejidad. De los resultados obtenidos, podemos concluir que estos algoritmos son capaces de salvar óptimos locales para acercarse a las regiones más prometedoras. Este aspecto es más notable si consideramos que la población del AGCR se había inicializado con individuos generados en una región alejada del óptimo global. Un ejemplo de AGCR Global puede ser PDS-S&E con $N_H = 200$ y $N_M = 400$ (ver Tabla 4.6).

Para obtener un comportamiento robusto frente a problemas con características diferentes, nos proponemos diseñar un modelo híbrido con los AGCRs Globales y los AGCRs Locales. Nuestra intención es aprovechar las ventajas de ambos simultaneamente, para alcanzar y refinar las regiones más prometedoras del espacio de búsqueda. Por ello, el objetivo de esta sección es diseñar un modelo AGCR híbrido robusto frente a la mayoría de problemas prácticos.

4.5.1. El Conflicto entre Precisión y Fiabilidad

A la hora de buscar el óptimo global, existe un conflicto fundamental entre *precisión* y *fiabilidad* en la mayoría de los problemas prácticos (Renders y Flasse, 1996). Tradicionalmente, este conflicto se ha tratado por medio de operadores genéticos avanzados (por ejemplo, el operador de mutación no uniforme propuesto por Michalewicz (1992), la adaptación de parámetros de control del AG, por Eiben y otros (1999), poblaciones distribuidas heterogéneas, por Herrera y Lozano (2000a), etc.). En la actualidad, una alternativa que recibe especial atención es la *hibridación* de AGs con otras técnicas de búsqueda. Tres ejemplos importantes son:

- *Algoritmos Meméticos* (Moscato, 1999). Su principal idea es la de combinar, sobre una población de agentes, procesos de cooperación y competición, presentes en los Algoritmos Evolutivos, con procesos de mejora

individual. Una amplia representación de los Algoritmos Meméticos son Algoritmos Evolutivos en los que se aplica un proceso de BL para refinar los individuos de la población (Krasnogor y Smith, 2005).

- *Primero AG y después BL* (Chelouah y Siarry, 2003; Harada y otros, 2006). Se trata de una MH basada en BL reciente que lleva a cabo, principalmente, dos fases (ver Figura 4.8): En la primera, se requiere la aplicación de un AG durante un porción de la ejecución total; posteriormente, en la segunda fase, se realiza un proceso de BL sobre la mejor solución encontrada, o mejores soluciones encontradas, por el AG.

- *Hibridación de AGs con diferentes propósitos.* En esta categoría entran, entre otras, todas las propuestas de AGLs, revisadas en la Sección 2.3, que se combinan con otro AG. En la mayoría de estas propuestas, el AGL se encargaba de la intensificación y el otro AG de la diversificación.

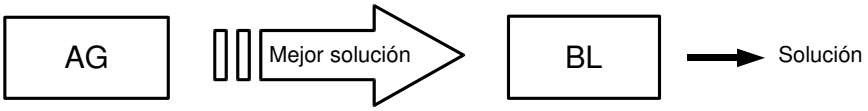


Figura 4.8: MH *primero AG y después BL*

4.5.2. Combinando AGCRs Locales y AGCRs Globales

En esta sección, proponemos un modelo de AGCR híbrido que combina un AGCR Local (con N_H^L y N_M^L como sus parámetros N_H y N_M, respectivamente) y un AGCR Global (con N_H^G y N_M^G como sus parámetros N_H y N_M, respectivamente). La hibridación se lleva a cabo siguiendo la idea de primero AG y después BL (Chelouah y Siarry, 2003; Harada y otros, 2006) porque es simple (ver Figura 4.9):

1. Se ejecuta el AGCR Global durante un porcentaje P_G % de las evaluaciones disponibles.

2. Aplicamos el AGCR Local hasta consumir el resto de las evaluaciones disponibles. La población inicial del AGCR Local se compone de los N_{max}, con N_{max} igual al máximo entre N_H^L y N_M^L, mejores individuos de la población final del AGCR Global.

Debemos indicar, además, que este modelo no se ha tenido en cuenta anteriormente para hacer uso de AGLs.

Esta hibridación sigue la heurística clásica: '*primero exploración y después explotación*'. Esta heurística la siguen otras técnicas de búsqueda como *Enfriamiento Simulado* (Kirkpatrick y otros, 1983). Con la diversificación inicial, se incrementa la probabilidad de encontrar zonas cercanas a las soluciones óptimas. Entonces, suponiendo que la población tiene información de esas zonas, se produce la convergencia hacia el óptimo por medio de la intensificación.

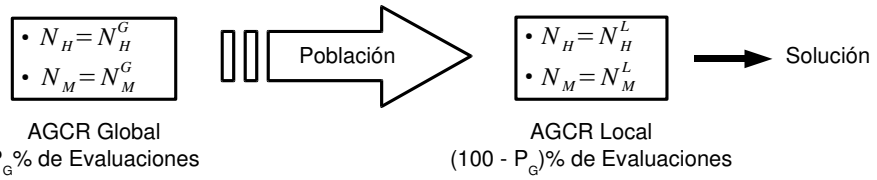


Figura 4.9: Modelo de AGCR híbrido

Hemos implementado tres algoritmos de este modelo. El AGCR Global es el algoritmo PDS-S&E con $N_H^G = 200$ y $N_M^G = 400$ y el AGCR Local es el mismo algoritmo pero con $N_H^L = 5$ y $N_M^L = 100$. Los algoritmos se distinguen por los valores considerados para P_G ($P_G = 25\,\%$, $50\,\%$ y $75\,\%$). Los llamaremos GL-25, GL-50 y GL-75, respectivamente. Sus resultados se muestran en la Tabla 4.8. Hemos incluido los resultado del AGCR Global y del AGCR Local al ejecutarse independientemente.

Tabla 4.8: Resultados de los modelos híbridos

Algoritmo	f_{esf}	f_{Ros}	f_{Sch}	f_{Ras}	f_{Gri}	P_{sle}
AGCR Local	9,98e−187	1,56e+0	1,74e−9	2,90e+2	1,27e−02	1,64e+2
GL-25	3,17e−147	7,61e−1	1,61e−7	1,33e+1	2,22e−17	4,69e+0
GL-50	1,29e−104	6,03e+0	1,22e−5	8,26e+0	1,33e−17	3,25e+0
GL-75	3,94e−061	1,22e+1	3,26e−2	3,74e+0	0,00e+00	2,72e+0
AGCR Global	2,95e−018	1,91e+1	3,12e+1	1,92e+1	4,93e−04	5,45e+0

Debemos destacar que, para cuatro problemas, existen algoritmos de nuestro modelo híbrido que obtienen mejores resultados que el solo uso del AGCR Global o del AGCR Local correspondiente: f_{Ros} (GL-25), f_{Ras} (GL-50 y GL-75), f_{Gri} (GL-25, GL-50 y GL-75) y P_{sle} (GL-25, GL-50 y GL-75). Esto significa que la técnica de hibridación propuesta es una forma adecuada de producir *sinergia* entre el AGCR Global y el AGCR Local.

Otra observación importante es que GL-25 y GL-75 han obtenido soluciones para f_{Ros} y f_{Gri}, respectivamente, que son las mejores comparadas con las obtenidas por todos los algoritmos previamente estudiados (ver Tabla 4.7).

4.5.3. Comparación con Otros Algoritmos

La principal intención de esta sección, es comparar nuestro modelo AGCR híbrido con otras MHs presentadas en la literatura para la optimización continua:

- *Dos AGCRs basados en OCCPs*: SPC-PNX (Ballester y Carter, 2004) y G3-PCX (Deb y otros, 2002a).

- SPC-PNX es un AGCR estacionario que usa el operador de cruce PNX. Hemos considerado cuatro algoritmos que usan diferentes tamaños de población ($N = 40$, 60, 100 y 200). Los llamaremos SPC-PNX-N.
- G3-PCX es otro AGCR estacionario que aplica el operador de cruce PCX (ver Sección 1.6.5). Hemos implementado varios algoritmos que usan diferentes valores para el parámetro λ del operador PCX (λ= 1, 2, 3 y 4), $\mu = 3$, y una población con 150 individuos. Los otros parámetros del operador PCX son: $\sigma_\varsigma^2 = 0{,}1$ y $\sigma_\eta^2 = 0{,}1$. Nos referiremos a estos algoritmos como G3-PCX-λ.

- *Dos AGCRs híbridos*: un Algoritmo Memético con codificación real y el operador de ascensión de colinas basado en cruce (AMCR-OACC) (Lozano y otros, 2004) y un algoritmo híbrido que combina CHC (Eshelman, 1991) con el algoritmo de Solis y Wets (1981), el cual es una técnica de BL (CHC-SW).
  - AMCR-OACC es la propuesta inicial donde se utilizó OACC, AGL comentado en la Sección 2.3.2. Sus autores afirman que este algoritmo mejora el rendimiento de otros Algoritmos Meméticos con codificación real presentes en la literatura (Lozano y otros, 2004).
  - CHC-SW sigue la idea primero AG y después BL usada en nuestro modelo híbrido. Primero, se ejecuta CHC durante el $P_G\,\%$ de las evaluaciones disponibles, y después, el método de BL refina la mejor solución encontrada por CHC. Hemos considerado los mismos tres valores para P_G, usados anteriormente en nuestro modelo híbrido ($P_G\,\%$= 25 %, 50 % y 75 %). Estos algoritmos se llamarán CHC-SW-P_G.

- *Enfriamiento Simulado Mejorado* (ESM) (Siarry y otros, 1997). Es una variante continua del conocido Enfriamiento Simulado (Kirkpatrick y otros, 1983). Se usaron los valores de los parámetros propuestos por Siarry y otros (1997). Además, también se llevó a cabo el proceso de normalización de variables, propuesto en ese trabajo, al intervalo $[-1, 1]$.

- *Evolución Diferencial* (ED[1]) (Storn y Price, 1997). Es un Algoritmo Evolutivo que sigue la idea de *Simplex de Nelder & Mead* (Nelder y Mead, 1965). ED evoluciona una población de soluciones combinando sus individuos. Hemos considerado tres algoritmos:
  - ED clásico, que usa la estrategia */rand/1/bin*, con $10 \cdot D$ miembros en su población (D es el número de variables del problema), $F = 0{,}8$ y $CR = 0{,}5$, como sugiere el código; y
  - dos algoritmos con 60 y 100 individuos, que usan $F = 0{,}5$, $CR = 0{,}8$ y la estrategia */rand/1/exp*. Éstos se nombrarán ED-60 y ED-100, respectivamente.

- *Optimizador de Partículas con Aprendizaje Exhaustivo* (OPAE[2]) (Liang y otros, 2004b,a). Es una variante de los Algoritmos Basados en Nubes

[1] Código MATLAB disponible en http://www.icsi.berkeley.edu/ storn/code.html#matl
[2] El código se ofrece en http://www.ntu.edu.sg/home/EPNSugan/

de Partículas (Eberhart y Kennedy, 1995; Kennedy y Eberhart, 2001), los cuales simulan el comportamiento de las aves volando. Hemos usado dos algoritmos: OPAE con 10 partículas, como Liang y otros (2004a) sugieren, y OPAE-30, que usa 30 partículas.

- *Estrategia de Evolución con Adaptación de la Matriz de Covarianza* (EE-AMC[3]) (Hansen y Ostermeier, 2001; Hansen y otros, 2003). Representa el estado del arte de las Estrategias de Evolución y es un referente en el campo de la optimización continua. El *tamaño del paso* inicial (en inglés *step-size*), parámetro $\sigma^{(0)}$ del algoritmo, se ha puesto a la mitad del intervalo de inicialización, como sugiere el código.

- *Programación Evolutiva con Mutaciones basadas en la Distribución de Probabilidad Levy* (PEML) (Lee y Yao, 2004) y la variante llamada PEML adaptativo (PEMLA) (Lee y Yao, 2004). PEML tiene un parámetro α, que los autores recomiendan asignar a 1,4. PEMLA es una variante que utiliza cuatro valores para el parámetro α simultáneamente. Hemos implementado ambos algoritmos. Además, hemos probado dos valores para el *tamaño de paso* inicial σ_0 (1 con $\sigma_0 = 0{,}015 \cdot D_{width}$, y 2 con $\sigma_0 = 0{,}01$). Por tanto, se han añadido cuatro algoritmos, PEML-1, PEML-2, PEMLA-1 y PEMLA-2.

Para esta comparación, usaremos las 18 funciones de prueba descritas en el Apéndice A.2. Hemos elegido GL-25 (ver la Tabla 4.8) para la comparación, porque muestra un grado aceptable de robustez frente a problemas con diferentes características. Se han realizado 50 ejecuciones por cada algoritmo, consumiendo, cada una, 100.000 evaluaciones de la función objetivo. Las Tablas B.9 a B.11 muestran los resultados de cada algoritmo en cada función (Apéndice B.2.2).

Hemos aplicado un análisis estadístico no paramétrico para detectar el algoritmo que mejor se comporta en el conjunto de todas las funciones de prueba, y detectar si hay diferencias significativas entre sus resultados y el de los otros algoritmos. La Tabla 4.9 muestra el estadístico de *Iman-Davenport* (ver Apéndice C) y su valor crítico al 5 % cuando comparamos los rankings medios (calculados según el test de *Friedman* (Apéndice C)) de todos los algoritmos sobre todos los problemas. Podemos observar que existen diferencias significativas entre los resultados, porque el valor estadístico es mayor que su valor crítico (1,557).

Tabla 4.9: Test de *Iman-Davenport* con todas las funciones de prueba

Iman-Davenport	**Valor crítico**
15,423	1,557

Atendiendo a este resultado, procedemos a comparar el algoritmo con el mejor ranking (GL-25) con el resto mediante un análisis estadístico post-hoc. La Tabla 4.10 muestra el ranking medio de cada algoritmo (los rankings bajos son mejores) y los resultados de los tests de *Holm* y *Wilcoxon* al nivel 0,05 de

[3]Hemos usado el código MATLAB, versión 2,34a, disponible en http://www.bionik.tu-berlin.de/user/niko/index.html

significación, comparando el mejor algoritmo (en la última fila) con el resto. En la columna *Holm*, el carácter 'R' indica que la hipótesis nula se rechaza, esto es, hay diferencias significativas entre el ranking de GL-25 y el del algoritmo correspondiente; mientras que el carácter 'N' indica que la hipótesis nula no se pudo rechazar. En las columnas correspondientes al test de *Wilcoxon*, se muestran los valores de R^- (asociados al algoritmo de mejor ranking, esto es, GL-25) y los de R^+ (asociados al algoritmo correspondiente) junto con el valor crítico. En la última columna, el signo más (+) indica que se rechaza la hipótesis nula de este test, esto es, hay diferencias significativas entre los resultados obtenidos por el algoritmo de mejor ranking (GL-25) y los del algoritmo correspondiente, y que, además, los resultados de GL-25 son mejores. El signo menos (−) también indicará que la hipótesis nula se rechaza, pero que los resultados de GL-25 son peores. Por último, el signo de similitud (∼) indicará que la hipótesis nula no se pudo rechazar.

Podemos ver que GL-25 obtiene el mejor ranking. Además, los tests estadísticos encuentran diferencias significativas entre su rendimiento y el de la mayoría de los algoritmos restantes, siempre a favor de GL-25. Por tanto, podemos concluir que GL-25 es un algoritmo de optimización muy robusto frente a problemas con diferentes características, que ofrece soluciones de alta calidad. Además, para conocer el comportamiento de GL-25 en cada problema, frente al de los demás algoritmos, hemos aplicado el test de *Student*, cuando se cumplen las condiciones de normalidad e igualdad de varianzas, o el de *Mann-Whitney* en otro caso (ver Apéndice C). Los resultados de estos tests se han añadido a las Tablas B.9 a B.11. Aquí, hemos introducido la Figura 4.10, que muestra el porcentaje de funciones en que GL-25 obtiene resultados significativamente superiores (+), resultados sin diferencia estadísticamente significativa (∼), o resultados significativamente inferiores (−) frente a cada uno de los algoritmos.

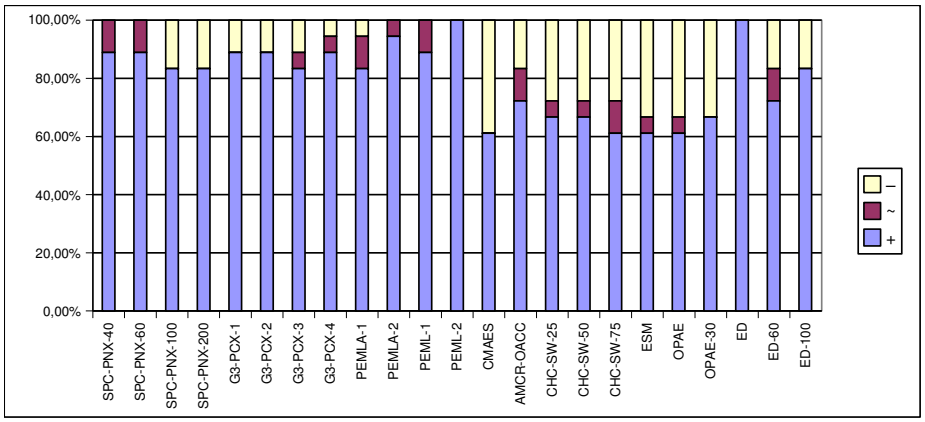


Figura 4.10: Resultados de GL-25 con respecto a cada algoritmo de comparación

A partir de la Figura 4.10, podemos realizar los siguientes comentarios:

- GL-25 obtiene mejores resultados que todos los algoritmos para más del 60 % de los problemas de prueba. Por ello, GL-25 obtenía el mejor ranking en el estudio anterior y se convierte en un algoritmo prometedor para trabajar con problemas de optimización continua.

Tabla 4.10: Tests de *Holm* y *Wilcoxon* con todas las funciones de prueba

			Wilcoxon			
Algoritmo	**Ranking**	***Holm***	R^+	R^-	**Valor crítico**	**Resultado**
ED	20,778	R				
PEML-2	20,111	R				
G3-PCX-1	19,222	R				
PEMLA-2	19,167	R				
G3-PCX-2	17,722	R				
G3-PCX-3	17,056	R				
PEML-1	16,611	R				
G3-PCX-4	16,444	R				
PEMLA-1	15,778	R				
SPC-PNX-40	12,722	R				
ED-100	12,333	R				
SPC-PNX-60	10,833	N	13	158	40	+
SPC-PNX-200	10,778	N	33	138	40	+
SPC-PNX-100	10,278	N	36	135	40	+
EE-AMC	9,889	N	40	131	40	+
OPAE-30	9,889	N	67	104	40	∼
ED-60	9,806	N	28	143	40	+
ESM	9,583	N	70	101	40	∼
CHC-SW-25	8,111	N	38	133	40	+
AMCR-OACC	7,722	N	49	122	40	∼
OPAE	7,194	N	71	100	40	∼
CHC-SW-50	6,889	N	42	129	40	∼
CHC-SW-75	6,250	N	53	118	40	∼
GL-25	4,833					

- EE-AMC, ESM, OPAE y CHC-SW son los más competitivos para GL-25 con respecto a los resultados de este último análisis estadístico.

Para comparar el comportamiento de GL-25 con el de EE-AMC, ESM, OPAE y CHC-SW, examinamos sus resultados en dos grupos diferentes de problemas: unimodales (f_{esf} – f_{QNoise}), y multimodales (f_{Ras} – $f_{Schaffer}$, y P_{sle} y P_{fms}). Además, hemos añadido el algoritmo AMCR-OACC, para el cual los tests estadísticos no paramétricos no podían rechazar la hipótesis nula. Las Tablas 4.11 y 4.12 realizan un análisis estadístico sobre los resultados en funciones unimodales. Las Tablas 4.13 y 4.14 realizan un análisis estadístico sobre los resultados en las funciones multimodales. En ambos casos, el test de *Iman-Davenport* detectó diferencias significativas y procedimos con un análisis *post-hoc*. Además, la Figura 4.11 muestra el porcentaje, en grupos separados de funciones, para las que GL-25 obtiene mejores resultados, peores o sin diferencias estadísticas, que el resto de algoritmos. Podemos observar que:

Tabla 4.11: Test de *Iman-Davenport* sólo con las funciones unimodales

Iman-Davenport	**Valor crítico**
4,316	2,138

Tabla 4.12: Tests de *Holm* y *Wilcoxon* sólo con las funciones unimodales

			Wilcoxon			
Algoritmo	**Ranking**	***Holm***	R^+	R^-	**Valor crítico**	**Resultado**
OPAE-30	8,143	R				
ESM	6,429	R				
CHC-SW-25	5,571	N	7	21	2	∼
OPAE	5,571	N	6	22	2	∼
CHC-SW-75	5,143	N	6	22	2	∼
CHC-SW-50	5	N	7	21	2	∼
AMCR-OACC	3,857	N	6	22	2	∼
GL-25	2,857	N	6	22	2	∼
EE-AMC	2,429					

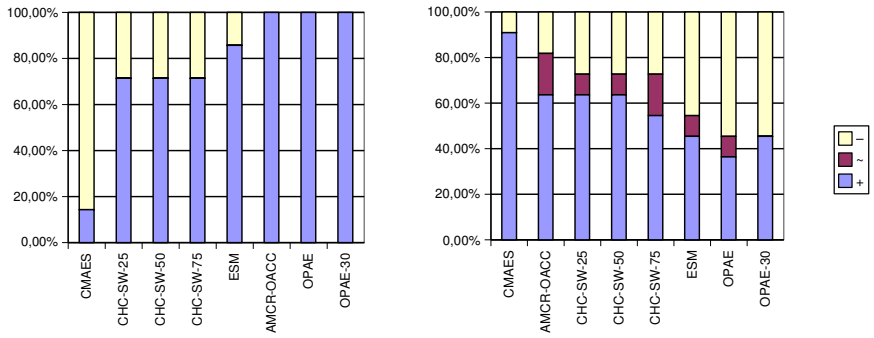
Tabla 4.13: Test de *Iman-Davenport* sólo con las funciones multimodales

Iman-Davenport	**Valor crítico**
2,654	2,056

- GL-25 obtiene, en los dos casos, el segundo mejor ranking y resultados comparables a los del mejor ranking. Esto muestra claramente que GL-25 es un algoritmo muy robusto para el problema de la optimización continua.

Tabla 4.14: Test de *Holm* y *Wilcoxon* sólo con las funciones multimodales

			Wilcoxon			
Algoritmo	**Ranking**	***Holm***	R^+	R^-	**Valor crítico**	**Resultado**
EE-AMC	7,636	R				
AMCR-OACC	6	N	7	59	10	+
CHC-SW-25	5,773	N	20	46	10	∼
CHC-SW-50	4,955	N	24	42	10	∼
ESM	4,455	N	23,5	42,5	10	∼
CHC-SW-75	4,273	N	28,5	37,5	10	∼
OPAE-30	4,091	N	47	19	10	∼
GL-25	3,909	N	28	38	10	∼
OPAE	3,909					



(a) Funciones unimodales (b) Funciones multimodales

Figura 4.11: GL-25 frente a EE-AMC, CHC-SW, ESM, OPAE y AMCR-OACC: (a) funciones unimodales, y (b) funciones multimodales

- EE-AMC es el mejor algoritmo para problemas unimodales, aunque no consigue muchas diferencias significativas con el resto de algoritmos. Por otro lado, OPAE es el mejor algoritmo para problemas multimodales, y tampoco consigue muchas diferencias significativas.

- Por un lado, EE-AMC obtiene mejores resultados que GL-25 en casi todas las funciones unimodales (Figura 4.11.a). Sin embargo, GL-25 mejora a EE-AMC en la mayoría de las funciones multimodales (Figura 4.11.b). Teniendo en cuenta que algunos autores consideran EE-AMC como una estrategia de BL (Hansen y Ostermeier, 1996), entendemos que obtenga tan buenos resultados en funciones unimodales y no tan buenos en funciones multimodales.

- Por otro lado, ESM, OPAE y OPAE-30 mejoran a GL-25 en la mayoría de las funciones multimodales (Figura 4.11.b). Sin embargo, GL-25 obtiene mejores resultados en las unimodales (Figura 4.11.a). Podemos ver

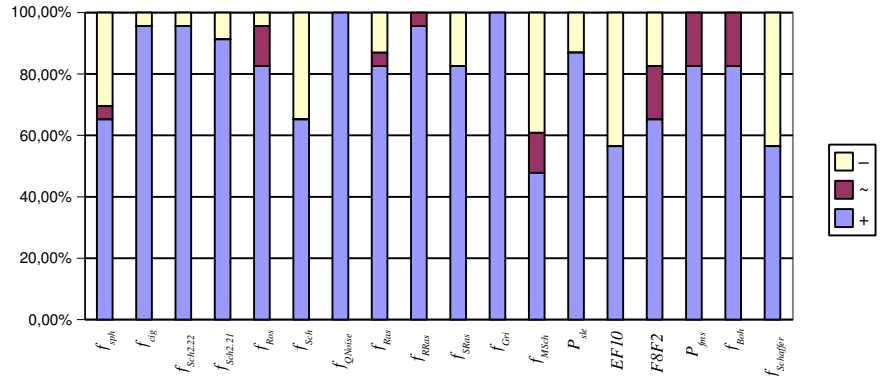


Figura 4.12: Resultados de GL-25 en cada problema

en las Tablas B.9 a B.11, que ESM, OPAE y OPAE-30 obtienen resultados destacables en funciones multimodales (por ejemplo, los resultados obtenidos por ESM, OPAE y OPAE-30 para las funciones f_{SRas} y f_{Boh}), mientras que esto no ocurre en las funciones unimodales. Esto revela que los algoritmos ESM, OPAE y OPAE-30 producen una búsqueda global.

- Aunque GL-25 mejora a los algoritmos CHC-SW, éstos muestran un comportamiento similar, con respecto a GL-25, para ambos tipos de problemas de prueba. Esto puede deberse a que estos algoritmos se han diseñado para obtener un comportamiento equilibrado entre búsqueda global y local, combinando el algoritmo CHC con la técnica de BL de *Solis* y *Wets*.

Finalmente, introducimos la Figura 4.12, con la intención de ver el rendimiento general de GL-25 en cada uno de los problemas. Ésta muestra el porcentaje de algoritmos para los que GL-25 obtiene mejores, peores o resultados sin diferencias estadísticas para cada función, con respecto a los otros algoritmos de comparación.

A partir de la Figura 4.12, podemos comentar las siguientes observaciones:

- GL-25 consigue mejores resultados que el 60 % de todos los algoritmos en todas las funciones, excepto en f_{MSch}, $EF10$ y $f_{Schaffer}$, donde obtiene mejores o iguales resultados que más del 50 %. Por ello, concluimos que el equilibrio entre búsqueda global y local obtenido por GL-25 produce beneficios destacables.

- GL-25 obtiene los mejores resultados para f_{QNoise}, que se destaca por presentar alto ruido.

- GL-25 es uno de los mejores algoritmos para las funciones multimodales f_{Gri} y f_{Boh}, los dos problemas reales, P_{sle} y P_{fms}, y la función de Rastrigin Rotada, f_{RRas}.

- Aunque el rendimiento de GL-25 es bueno en todas las funciones, las mayores dificultades aparecen en f_{esf}, f_{Sch}, f_{MSch}, $EF10$ y $f_{Schaffer}$.

Por un lado, f_{esf} y f_{Sch} son funciones unimodales poco complejas, que requieren un comportamiento fuertemente local para poder resolverse adecuadamente, mientras que f_{MSch}, $EF10$ y $f_{Schaffer}$ son problemas muy complejos que necesitan un aporte extra de búsqueda global para obtener resultados fiables. Dado que GL-25 pretende mantener un equilibrio entre búsqueda local y global, no alcanza soluciones altamente precisas en estos problemas.

Podemos concluir que nuestra propuesta de combinar un AGCR Global con un AGCR Local es muy competitiva con el estado del arte en *MHs para la optimización continua*. Esto se debe a que realiza un proceso robusto para problemas de prueba con diferentes características.

4.6. Conclusiones

Este capítulo trata el estudio de AGCRs Locales basados en OCCPs, y de su integración con AGCRs Globales, para el problema de la optimización continua. Para ello, hemos presentado un procedimiento de diferenciación sexual, un mecanismo de selección del progenitor hembra (selección uniforme en fertilidad) y hemos seleccionado un mecanismo de selección del progenitor macho (emparejamiento variado negativo) que, al determinar los progenitores que intervienen en la aplicación del OCCP, nos permiten ajustar de forma precisa el equilibrio entre intensificación y diversificación que aportará el AGCR:

- El procedimiento de diferenciación sexual asigna el papel de progenitor hembra y/o macho a los cromosomas de la población.
- Selección uniforme en fertilidad extiende la búsqueda, porque obliga al OCCP a considerar diferentes zonas para la generación de descendientes. Para ello, fomenta una cobertura intensa de las regiones representadas por los progenitores hembra.
- Emparejamiento variado negativo diversifica, porque fuerza al OCCP a crear descendientes poco similares a sus padres. De esta forma, este mecanismo de emparejamiento ayuda al OCCP a producir diversidad útil.

Un estudio experimental llevado a cabo con el operador de cruce PBX-α ha mostrado que el ajuste preciso de los parámetros del procedimiento de diferenciación sexual (N_H y N_M) permite especializar el nuevo modelo de AGCR y obtener:

- *AGCRs Locales* que obtienen soluciones precisas a la hora de alcanzar un óptimo local, y también
- *AGCRs Globales* que obtienen soluciones fiables, realizando una exploración global del espacio de búsqueda.

Con la intención de obtener un algoritmo *robusto*, hemos seguido una técnica simple de hibridación para combinar estos algoritmos de búsqueda especializados. Hemos confirmado empíricamente que: 1) esta técnica permite que aparezca

sinergia entre el AGCR Global y el AGCR Local, esto es, su combinación obtiene resultados mejores que el solo uso de cualquiera de los dos, y 2) la MH resultante es muy competitiva frente al *estado del arte en MHs para la optimización continua*.

Comentarios Finales

Dedicamos esta sección a resumir brevemente los resultados alcanzados y a destacar las principales conclusiones obtenidas en esta memoria. Presentaremos las publicaciones asociadas a esta memoria y comentaremos algunos aspectos sobre trabajos futuros que siguen la línea aquí expuesta, y otras líneas de investigación que se puedan derivar.

A Resumen y Conclusiones

En esta memoria, hemos estudiado profundamente los *Algoritmos Genéticos Locales*. Para ello, hemos realizado una revisión de las propuestas presentadas en la literatura. Posteriormente, hemos presentado dos nuevas propuestas, la primera para problemas con codificación binaria, y la segunda para problemas de optimización continua. En un primer estudio, hemos concluido que los AGLs pueden promover una intensificación más fructífera que los métodos clásicos de BL. En el segundo, además, hemos acometido el estudio de una MH que, aplicando el segundo AGL propuesto, se muestre competitiva frente al estado del arte en los problemas de optimización continua.

Los siguientes apartados resumen brevemente los resultados obtenidos y presentan algunas conclusiones sobre los mismos.

A Algoritmos Genéticos Locales

Hemos introducido el concepto de *Algoritmo Genético Local*, una categoría novedosa de MHs basadas en poblaciones especializadas en promover intensificación. Éstos aparecen por la dificultad de proveer un equilibrio apropiado entre intensificación y diversificación en un solo AG. La opción propuesta en la literatura es especializar el AG, obteniendo AGLs para intensificación, que se combinan con otra MH diversificadora.

Para ofrecer una visión global de los AGLs, hemos analizado cuidadosamente las decisiones de diseño de las diferentes propuestas de la literatura. Nos hemos percatado de que los autores han seguido caminos muy diversos para aumentar la intensificación en el AG, lo que nos hace intuir el gran número de posibilidades disponibles, y nos sugiere que el diseño de AGLs más potentes es susceptible de más estudios.

Por su naturaleza intensificadora, los AGLs deben combinarse con otras MHs diversificadoras. Tras estudiar éstas últimas, nos damos cuenta de que los AGLs se aplican inicialmente en conjunción con otros AGs dentro de modelos distribuidos, y posteriormente, como procedimiento de BL en Algoritmos Meméticos. Este resultado nos muestra la facilidad de aplicación, y significación de este nuevo tipo de MHs, y además, nos deja abierto el estudio de su aplicación en otras MHs.

Finalmente, con la intención de clasificarlos entre las MHs existentes, los hemos comparado con éstas en base a dos criterios: 1) el uso de sólo una, o una población de soluciones, y 2) su interés por ofrecer intensificación, diversificación o ambas. Como resultado, hemos observado que las *MHs basadas en trayectorias* han evolucionado, de forma natural, desde intensificación hacia un equilibrio entre intensificación y diversificación, mientras que las *MHs basadas en poblaciones* lo han hecho desde un equilibrio entre ambas hacia diversificación. Los AGLs se convierten, por tanto, en una novedosa y atractiva forma de utilizar a las MHs basadas en poblaciones para promover intensificación.

B Algoritmo Genético Local Binario

En el Capítulo 3, hemos propuesto AGLB, un AGL que aplica un *método de reemplazo por agrupamiento* para favorecer la formación de *nichos* en su población, los cuales representan las regiones de búsqueda más importantes. Después, AGLB realiza un proceso de BL que refina la solución dada de acuerdo a la información en los nichos más cercanos.

Con la intención de obtener nuevos resultados y conclusiones en el campo de los AGLs, y MHs que los utilicen, hemos realizado un extenso estudio, analizando el comportamiento de AGLB en tres MHs basadas en BL, que no se habían utilizando anteriormente, y que introducen, de forma progresiva, una coordinación controlada con el método de refinamiento. Su comparación con las correspondientes MHs basadas en BL que utilizan métodos clásicos de BL nos ha permitido concluir que:

- AGLB provee una intensificación útil porque realiza un refinamiento eficaz y eficiente. Por un lado, AGLB es capaz de encontrar soluciones muy precisas en los diferentes entornos en los que se ha utilizado (con diversificación aleatoria, fija o adaptativa). Por otro lado, AGLB consume menos evaluaciones por refinamiento que cualquier otra BL comparada.

- La capacidad de aprendizaje y buen uso de la información adquirida, observados en AGLB, marca una importante diferencia con respecto a los métodos clásicos de BL. AGLB utiliza esta información para mejorar el proceso de refinamiento de nuevas soluciones iniciales, tanto en eficiencia, como en eficacia.

- En general, AGLB puede mejorar los resultados de MHs basadas en BL con diferentes características, permitiendo alcanzar un equilibrio adecuado y estratégico entre diversificación e intensificación.

C Algoritmo Genético Local Continuo

Hemos diseñado un modelo de *AGCRs especializados*, basados en operadores de cruce centrados en un padre que incorpora tres mecanismos para determinar los cromosomas que participan en la operación del operador de cruce centrado en un padre:

- Un *procedimiento de diferenciación sexual* cataloga los cromosomas de la población como posibles progenitores hembra y/o macho.
- La *selección uniforme en fertilidad* fomenta que las regiones representadas por las hembras sean equivalentemente exploradas por el operador de cruce.
- El *emparejamiento variado positivo* pretende que el operador de cruce cree descendientes poco similares a sus padres, ayudando a producir diversidad útil.

Un estudio experimental, llevado a cabo con el operador de cruce PBX-α, ha mostrado que estos tres procesos permiten, mediante el ajuste preciso de sus parámetros de control, originar dos tipos de AGCRs especializados para diferentes problemas de optimización continua:

- *AGCRs Locales* especializados en intensificación y que obtienen soluciones precisas a la hora de alcanzar un óptimo local.
- *AGCRs Globales* especializados en diversificación y que obtienen soluciones fiables realizando una exploración global del espacio de búsqueda.

Finalmente, hemos diseñado una *MH híbrida* que combina un AGCR Local y un AGCR Global con la intención de obtener sus ventajas simultaneamente. Hemos confirmado empíricamente que la MH resultante permite que aparezca sinergia entre el AGCR Local y el AGCR Global, y además, sea muy competitiva con el *estado del arte en MHs para la optimización continua*.

B Publicaciones Asociadas a la Tesis

A continuación, presentamos un listado de las publicaciones asociadas a esta memoria.

Publicaciones en Revistas Internacionales

1. García-Martínez C, Lozano M, Herrera F, Molina D, Sánchez AM (2008) Global and Local Real-Coded Genetic Algorithms Based on Parent-Centric Crossover Operators. Eur. J. Oper. Res. 185:1088–1113.
2. Lozano M, García-Martínez C (2008) Hybrid Metaheuristics with Evolutionary Algorithms that Specialize in Intensification and Diversification: Overview and Progress Report. Comput. Oper. Res. Sometido.

3. García-Martínez C, Lozano M (2008) Performance Evaluation of a Local Evolutionary Algorithm. J. Heur. Sometido

Publicaciones en Capítulos de Libro

1. García-Martínez C, Lozano M (2008) Local Search based on Genetic Algorithms. En: Siarry P, Michalewicz Z (eds) Advances in Metaheuristics for Hard Optimization, Springer's Natural Computing Series, pp. 199–221.

Publicaciones en Congresos Internacionales

1. García-Martínez C, Lozano M (2005) Hybrid Real-Coded Genetic Algorithms with Female and Male Differentiation. Proc. del Congreso IEEE en Evolutionary Computation, pp. 896–903.
2. García-Martínez C, Lozano M, Molina D (2006) A Local Genetic Algorithm for Binary-Coded Problems. Proc. del Parallel Problem Solving from Nature - PPSN IX, Lecture Notes in Computer Science 4193, pp. 192–201.

Publicaciones en Congresos Nacionales

1. García C, Lozano M (2005) Diferenciación sexual de cromosomas para la aplicación de operadores de cruce con codificación real centrados en un padre. Cuarto Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB05), Thomson Editores, pp. 389–396.
2. García-Martínez C, Lozano M (2007) Algoritmos Genéticos Locales. Actas del Segundo Congreso Español de Informática (CEDI 2007), I Jornadas sobre Algoritmos Evolutivos y Metaheurísticas (JAEM'07), Zaragoza (España), pp. 245–252.
3. Lozano M, Herrera F, García C (2005) Técnicas de diversificación para la mejora de los operadores de cruce centrados en un padre. Cuarto Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB05), Thomson Editores, pp. 15–22.

C Trabajos Futuros

En nuestra opinión, existe mucho campo merecedor de esfuerzos en investigación para descubrir aplicaciones exitosas de los AGLs, el cual se escapa del ámbito de esta memoria. Tras los estudios realizados, creemos que hay fuertes motivaciones para garantizar futuras investigaciones en las siguientes áreas:

- Estudiar el rendimiento de otras MHs basadas en BL, como *Algoritmos basados en Nubes de Partículas* o *Procedimientos de Búsqueda Ávidos Aleatorios y Adaptativos*, entre otros, cuando aplican AGLs. La relación y comunicación entre la MH basada en BL y el AGL sería de particular interés.

- Aplicar AGLs en MHs basadas en BL para resolver otros tipos de problemas, como la *optimización con restricciones*, *problemas multimodales* o *problemas multiobjetivo*.

- Estudiar la posibilidad de aplicar AGLs distribuidos con subpoblaciones especializadas en diferentes regiones del espacio de búsqueda. Aquí, los mecanismos de comunicación, encargados de manejar las subpoblaciones (crear, mezclar y migrar soluciones entre subpoblaciones), serían objeto de estudio.

- Diseñar *Algoritmos Evolutivos para intensificación* a partir de otras MHs basadas en poblaciones como las *Estrategias de Evolución*, los *Algoritmos de Estimación de Distribuciones* o los *Algoritmos basados en Nubes de Partículas*, por ejemplo.

En las siguientes secciones, detallamos las ideas comentadas.

I AGLs en Otras MHs Basadas en BL

Al aplicar AGLs en otras MHs basadas en BL, hay que prestar especial atención al flujo de información que aparece entre ellos. Podemos distinguir tres formas de aplicar AGLs:

- *Con comunicación mínima*: Aquí, la MH basada en BL sólo aporta soluciones iniciales al AGL, el cual devuelve las mismas refinadas a la MH basada en BL. AGLB, por ejemplo, se aplicó con comunicación mínima en los estudios del Capítulo 3.

- *Con comunicación limitada*: En este caso, la MH basada en BL ofrece cierta información al AGL para guiar el refinamiento y/o el AGL provee información a la MH basada en BL para guiar la búsqueda global. OACC (Sección 2.3.2) y los AGCRs especializados (Sección 4.5) son dos ejemplos de AGLs que se usaron con comunicación limitada. En el primero, la MH basada en BL ofrecía a OACC la solución a refinar y la mejor encontrada hasta el momento. En el segundo, el AGCR Local recibía toda la población final del AGCR Global ejecutado previamente.

- *Con comunicación absoluta*: En este caso, ambos, MH basada en BL y AGL, tienen a su disposición, y en cualquier momento, todo el conocimiento del problema en cuestión, que posee el otro. A nuestro conocimiento, no existe aún una propuesta que use este tipo de comunicación.

Mientras que el primer tipo de combinación es directamente aplicable a cualquier MH basada en BL, las otras dos permiten: 1) a la MH basada en BL aprovechar más información que sólo la solución refinada y/o 2) al AGL definir diferentes estrategias de optimización, que le puedan interesar a la MH basada en BL en algún momento de la búsqueda.

II Extensión a Otros Problemas

Existen algunos problemas que se diferencian de la optimización clásica de una función objetivo como son: la optimización con restricciones, la multimodal y la multiobjetivo. Adoptando mecanismos propuestos en el campo de los AGs, podemos diseñar AGLs que traten estos problemas. A continuación, describiremos estas tres clases de problemas.

- Muchos problemas de optimización en ciencia e ingeniería involucran un número de restricciones que la solución óptima debe satisfacer. Un problema de optimización con restricciones normalmente se define con una función a minimizar, un conjunto de restricciones de desigualdad y un conjunto de restricciones de igualdad sobre las variables de decisión.

  Existen varios métodos para para tratar los problemas con restricciones en el campo de los AGs (Deb, 2000; Michalewicz y Schoenauer, 1996). Adaptando los AGLs para que apliquen estos métodos, les daríamos la capacidad de tratar con este tipo de problemas.

- En la *optimización multimodal* no basta con encontrar una solución óptima del problema, sino que se pretende alcanzar un conjunto de soluciones óptimas (idealmente, todas las existentes) para ofrecerlas al experto y que pueda evaluarlas según otros criterios (costo de implementación, recuperación frente a errores,...).

  Para ello, los AGs utilizan técnicas que crean y mantienen *nichos*, grupos de soluciones similares entre ellas, localizadas en diferentes regiones del espacio de búsqueda (Sareni y Krahenbuhl, 1998). Las principales técnicas presentadas en la literatura son la *compartición de recursos* (Goldberg y Richardson, 1987; Goldberg, 1989b), el *método de limpiado* (Pétrowski, 1996), la *agrupamiento por reemplazo* (Harik, 1995; Mahfoud, 1992) y la combinación de *modelos con separación espacial* y *métodos de aprendizaje no supervisado* (Streichert y otros, 2004; Yang y otros, 2004).

  Por otro lado, varios estudios proponen el uso de procedimientos de BL sobre las soluciones de los diferentes nichos para obtener los óptimos locales con alta precisión (Chen y Kang, 2005; Feng y otros, 2006). Por tanto, la aplicación de AGLs en este tipo de problemas se convierte en un tópico de interés.

- Los *problemas multiobjetivo* (Chankong y Haimes, 1983) se caracterizan por el hecho de que hay que optimizar varios objetivos simultáneamente. Por tanto, no suele haber una sola mejor solución que resuelva el problema, es decir, que sea mejor que el resto con respecto a todos los objetivos, como en la optimización con una sola función objetivo. En vez de eso, hay un conjunto de soluciones que se consideran mejores que el resto cuando ofrecen mejores resultados para todos los objetivos, a este conjunto se le denomina conjunto *Pareto*. Las soluciones del conjunto pareto se conocen como soluciones *no dominadas* (Chankong y Haimes, 1983), mientras que las demás se conocen como soluciones dominadas. Dado que ninguna solución del conjunto Pareto es absolutamente mejor que las otras soluciones no dominadas, todas ellas son igual de aceptables con respecto a la satisfacción de todos los objetivos.

La aplicación de Algoritmos Evolutivos para el problema de la optimización multiobjetivo ha tenido un creciente interés en los últimos años (cec, 2007; Zitzler y otros, 2001; Obayashi y otros, 2007). Coello y otros (2007) realiza una clasificación de los modelos según el momento en que el experto suministra información sobre la preferencia entre los objetivos.

Por otro lado, los métodos de BL también han tenido un papel importante en propuestas muy competitivas para este tipo de problemas (Ishibuchi y otros, 2003; Kelner y otros, 2008; Knowles y Corne, 2000; Paquete y otros, 2007). Por ello, pensamos que la aplicación de AGLs en este campo pueden abrir novedosas líneas de investigación.

III Algoritmos Evolutivos para Intensificación

Como se ha mostrado en esta memoria, las MHs basadas en poblaciones pueden diseñarse para que ofrezcan intensificación a otra MH. En particular, nos hemos centrado en los AGs. Sin embargo, las ideas presentadas pueden aplicarse a otras MHs basadas en poblaciones, originando conceptos como *Algoritmos Evolutivos Locales*. A continuación, ofrecemos algunas referencias de trabajos en los que ya se han dado unos primeros pasos:

- *Estrategias de Evolución Locales*: Guanqi y Shouyi (2003) destaca la relación que tienen este tipo de MHs basadas en poblaciones y las MHs basadas en trayectorias. Auger y Hansen (2005) adapta EE-AMC (Hansen y Ostermeier, 2001) para que opere como método de BL dentro de una BL con Multiarranque Aleatorio.

- *Algoritmos de Estimación de Distribuciones Locales*: Sastry y Goldberg (2004) proponen un Algoritmo de Estimación de Distribuciones Local que usa una *estructura de vecindarios competente* para buscar en el *espacio de bloques constructores* definido por un modelo probabilístico estimado. Lima y otros (2006) presentan un modelo similar para problemas jerárquicos donde los bloques constructores se solapan unos a otros.

- *Algoritmos Basados en Colonias de Hormigas Locales*: Blum (2002) propone un Algoritmo basado en Colonias de Hormigas que alterna tres fases, dos de las cuales están diseñadas para promover una mayor intensificación: *convergencia local* y *convergencia al mejor*. En estas fases, sólo una solución de alta calidad se usa para actualizar las cantidades de feromona. Randall (2006) propone otro modelo con tres fases. En este caso, en la fase denominada *intensificadora*, algunos componentes de las soluciones se fijan a sus valores más prometedores (según la mejor solución obtenida), y las hormigas se encargan de proponer soluciones parciales con el resto de componentes. Por último, Kong y otros (2008) presentan uno con varias fases que suplen diferentes grados de intensificación, según si usan la mejor solución encontrada, o la mejor de la iteración anterior, para actualizar las cantidades de feromona.

Apéndice A

Problemas de Prueba

En este apéndice, describimos los problemas de prueba usados en la presente memoria. La Sección A.1 se centra en los problemas combinatorios con codificación binaria usados en los estudios comparativos del Capítulo 3. La Sección A.2 describe los problemas de optimización continua utilizados en los estudios empíricos del Capítulo 4.

A.1. Problemas con Codificación Binaria

En esta sección, describimos los problemas combinatorios con codificación binaria usados en la experimentación del Capítulo 3.

A.1.1. Problemas Engañosos o *Deceptive*

En los *problemas engañosos* (más conocidos por su término en inglés, *deceptive*) (Goldberg y otros, 1989), hay ciertos esquemas que guían el proceso de búsqueda hacia algunas soluciones que no son globalmente competitivas. Esto se debe a que, el esquema que presenta el óptimo global tiene poca significación y por ello, no prolifera durante el proceso de búsqueda. El problema *deceptive* usado se compone de la concatenación de trece subproblemas de tres variables binarias cada uno. Cada subproblema aporta una cantidad al valor de aptitud de la solución, indicada en la Tabla A.1. El valor de aptitud de la solución es la suma de todas las cantidades aportadas por cada subproblema.

Tabla A.1: Cantidad aportada en los subproblemas engañosos de orden 3

Bits	000	001	010	100	110	011	101	111
Aportación	28	26	22	14	0	0	0	30

Como puede apreciarse, la solución óptima consiste en la concatenación de unos. Sin embargo, salvo esta excepción, los subproblemas aportan un valor de aptitud más alto cuantos menos unos aparezcan.

A.1.2. Problemas Trampa o *Trap*

Un *problema trampa* (más conocido por su término en inglés, *trap*) (Thierens, 2004) se compone de subproblemas *deceptive* de diferentes longitudes. Específicamente, la función de aptitud de estos problemas $f(X)$ se construye añadiendo subfunciones de longitud uno (F_1), dos (F_2) y tres (F_3). Cada subfunción tiene dos óptimos: el valor óptimo se obtiene con una cadena sólo de unos, mientras que la cadena de ceros representa un óptimo local. El valor de aptitud de cualquier otra cadena se determina por el número de ceros: más ceros, mayor valor. Esto produce una base de atracción amplia hacia el óptimo local. El valor de aptitud para las subfunciones se especifica en la Tabla A.2, donde las columnas indican el número de unos presentes en las subfunciones F_1, F_2 y F_3. La función $f(X)$ se compone de cuatro subfunciones F_3, seis subfunciones F_2, y doce subfunciones F_1. El número de variables binarias es 36. La función $f(X)$ presenta 2^{10} soluciones óptimas de las cuales sólo una es la óptima global: la cadena sólo con unos, que tiene un valor de aptitud de 220.

Tabla A.2: Valor de aptitud de las subfunciones F_i de longitud i

Número de unos	**ninguno**	**uno**	**dos**	**tres**
F_3	4	2	0	10
F_2	5	0	10	
F_1	0	10		

$$f(X) = \sum_{i=0}^{3} F_3(x_{3i}x_{3i+1}x_{3i+2}) + \sum_{i=0}^{5} F_2(x_{2i+12}x_{2i+13}) + \sum_{i=0}^{11} F_1(x_{24+i}) \quad \text{(A.1)}$$

A.1.3. Problemas de Máxima Satisfacción o *Max-Sat*

El problema de *satisfacción* en lógica de proposiciones (Smith y otros, 2003) es la tarea de decidir si una fórmula proposicional dada, tiene modelo. Más formalmente, dado un conjunto de m cláusulas $(C_1, \ldots, C_m)$ que relacionan n variables booleanas $(x_1, \ldots, x_n)$, el problema de satisfacción consiste en decidir si existe una asignación de valores a las variables tal que todas las cláusulas sean satisfechas simultáneamente.

El problema de *máxima satisfacción* (*Max-Sat*) es la variante para optimización del problema de satisfacción y puede verse como una generalización de éste: dada una fórmula proposicional en forma normal clausulada, el problema *Max-Sat* consiste entonces en encontrar una asignación de valores a las variables

que maximiza el número de cláusulas satisfechas. La función asociada devuelve el ratio de cláusulas satisfechas.

En este tipo de problemas, una solución (cadena de bits) se traduce en una interpretación de la fórmula proposicional, asignando a cada variable x_i el valor verdadero, si x_i es 1, o falso, si x_i es 0.

Hemos usado dos conjuntos de problemas *Max-Sat* con 100 variables, tres variables por cláusula, y 1200 y 2400 cláusulas, respectivamente. Estos problemas se han obtenido a partir del generador comentado en De Jong y otros (1997). Los denotamos como *M-Sat*(n, m, l), donde l indica el número de variables relacionadas en cada cláusula. Cada ejecución i de cada algoritmo usará la misma semilla para generar el problema *M-Sat*(n, m, l), esto es, la ejecución i-ésima de cada algoritmo usa la semilla $semilla_i$, mientras que la ejecución j-ésima usa la semilla $semilla_j$.

A.1.4. Campos de Aptitud NK o *NK-Landscapes*

En los *campos de aptitud NK* (en inglés, *NK-Landscapes*) (NKLand) (Kauffman, 1989), N representa el número de genes en un cromosoma haploide y K representa el número de ligaduras que cada gen tiene con otros genes del mismo cromosoma. Para obtener el valor de aptitud del cromosoma al completo, se realiza la media de las contribuciones de cada *locus*:

$$f(X) = \frac{1}{N}\sum_{i=1}^{N} f(locus_i) \tag{A.2}$$

donde la aportación en el valor de aptitud de cada *locus* ($f(locus_i)$) se calcula de la siguiente forma: primero, se obtiene la cadena binaria representada por el gen i y los K genes a los que está ligado; después, se traduce la cadena binaria a su valor entero; finalmente, el valor entero se utiliza como índice en una tabla de tamaño 2^{K+1} con números generados aleatoriamente de forma uniforme en el intervalo $[0, 1]$; el valor de la celda indicada representará la aportación del $locus_i$. Para un gen dado, se puede establecer el conjunto de K genes con los que se relaciona, inicialmente de forma aleatoria o considerando los genes adyacentes.

Hemos usado dos conjuntos de problemas *NKLand* con diferentes valores para N y K: el primero con $N = 48$ y $K = 4$ y el segundo con $N = 48$ y $K = 12$. Los hemos denominado *NKLand*(N, K). Éstos se han obtenido según el generador comentado en De Jong y otros (1997). Cada ejecución i de cada algoritmo usará la misma semilla para generar el problema *NKLand*(N, K), esto es, la ejecución i-ésima de cada algoritmo usa la semilla $semilla_i$, mientras que la ejecución j-ésima usa la semilla $semilla_j$.

A.1.5. Problemas de P Picos o *PPeaks*

En los problema de P *Picos* (en inglés, *PPeaks*) (Spears, 2000) se tienen un cierto número de picos (el grado de multimodalidad del problema). Para un problema con P picos, se generan aleatoriamente P cadenas de bits de longitud

L. Cada una de esas cadenas representa un pico (un óptimo local) en el campo valores de aptitud. Se pueden asignar varias alturas a los picos basándose en diferentes esquemas (misma altura, lineal, logarítmica, etc.). Para evaluar una solución S, primero, se localiza el pico más cercano según la distancia *Hamming* de sus variables de decisión, lo llamaremos $Pico_n(S)$. Entonces, el valor de aptitud de S será el número de bits que la cadena tienen en común con $Pico_n(S)$, dividido por L, y escalado por la altura de $Pico_n(S)$. En caso de empate, al buscar el pico más cercano, se elegirá el de mayor altura.

Hemos usado diferentes problemas *PPeaks*, denotados como *PPeaks*(P, L). Cada ejecución i, de cada algoritmo, ha usado la misma semilla ($semilla_i$) para generar el problema *PPeaks*(P, L). Además, hemos seguido el esquema lineal para asignar las alturas a los picos en el rango [0,6, 1].

A.1.6. Problema del Corte Máximo o *Max-cut*

El problema del *corte máximo* (en inglés, *Max-cut*) (Karp, 1972) se define como sigue: sea un grafo conexo no dirigido $G = (V, E)$, donde se dan $V = \{1, 2, \ldots, n\}$ nodos, y $E \subset \{(i, j) : 1 \leq i < j \leq n\}$ arcos. Sean los pesos $w_{ij} = w_{ji}$ tales que $w_{ij} = 0\ \forall (i, j) \notin E$, y en particular, sean $w_{ii} = 0$. El problema del corte máximo consiste en encontrar una bipartición (V_1, V_2) de V tal que la suma de los pesos de los arcos entre nodos de V_1 y V_2 sea máxima.

Al tratar el problema del corte máximo, una cadena de bits (solución) representa una bipartición donde cada nodo i pertenece al conjunto V_1, si la posición i-ésima de la cadena de bits es un 0, o al conjunto V_2, si es un 1.

Hemos usado seis problemas (G10, G12, G17, G18, G19 y G43), obtenidos de Helmberg y Rendl (2000). Los hemos denominado como *Maxcut*(G10), *Maxcut*(G12) y así en adelante.

A.1.7. Problema de la Programación Cuadrática Binaria sin Restricciones

El objetivo de la *programación cuadrática binaria sin restricciones* (en inglés *unconstrained binary quadratic programming*, o *BQP*) (Gulati y otros, 1984) es el de encontrar, dada una matriz $n \times n$ racional simétrica $Q = (Q_{ij})$, un vector de longitud n que maximiza la siguiente cantidad:

$$f(x) = x^t Q x = \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} x_i x_j \tag{A.3}$$

Hemos usado cuatro problemas con diferentes valores de n, obtenidos de Beasley (1990). Éstos son los primeros de los ficheros 'bqp50', 'bqp100', 'bqp250' y 'bqp500'. Los denominamos $BQP(50)$, $BQP(100)$, $BQP(250)$ y $BQP(500)$, respectivamente.

A.2. Problemas de Optimización Continua

El conjunto de problemas de optimización continua usado se compone de dieciséis funciones de prueba y dos problemas reales. Describimos las funciones en la Sección A.2.1, y los problemas reales, en la Sección A.2.2.

A.2.1. Funciones de Prueba

A continuación, describimos las dieciséis funciones de prueba utilizadas. Indicaremos el número de variables de decisión de cada problema (Dimensión), los dominios de las variables, los rangos donde las poblaciones de los algoritmos se han inicializado, y el valor de aptitud de la solución óptima. Además, indicamos las funciones que *no son separables*, esto es, no se pueden optimizar teniendo en cuenta las variables individualmente, sino que es necesario tratarlas todas simultáneamente:

- *Esfera*:

$$f_{esf}(X) = \sum_{i=1}^{n} x_i^2 \tag{A.4}$$

  - Dimensión: 25
  - Dominio: $[-5{,}12, 5{,}12]$
  - Inicialización: $[4, 5]$
  - Valor de aptitud del óptimo global: 0

- *Cigar-tablet*:

$$f_{cig}(X) = x_1^2 + 10^8 x_n^2 + 10^4 \sum_{i=2}^{n-1} x_i^2 \tag{A.5}$$

  - Dimensión: 25
  - Dominio: $[-7, 7]$
  - Inicialización: $[5, 7]$
  - Valor de aptitud del óptimo global: 0

- *Problema* 2,22 *de Schwefel*:

$$f_{Sch2,22}(X) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i| \tag{A.6}$$

  - Dimensión: 25
  - Dominio: $[-10, 10]$
  - Inicialización: $[8, 10]$
  - Valor de aptitud del óptimo global: 0

- *Problema* 2,21 *de Schwefel*:

$$f_{Sch2,21}(X) = max\{|x_i|,\ 1 \leq i \leq n\} \tag{A.7}$$

- Dimensión: 100
- Dominio: $[-100, 100]$
- Inicialización: $[80, 100]$
- Valor de aptitud del óptimo global: 0

- *Función generalizada de Rosenbrock* (no separable):

$$f_{Ros}(X) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right] \tag{A.8}$$

  - Dimensión: 25
  - Dominio: $[-5{,}12, 5{,}12]$
  - Inicialización: $[-5, -4]$
  - Valor de aptitud del óptimo global: 0

- *Problema* 1,2 *de Schwefel* (no separable):

$$f_{Sch}(X) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2 \tag{A.9}$$

  - Dimensión: 25
  - Dominio: $[-65{,}536, 65, 536]$
  - Inicialización: $[60, 65]$
  - Valor de aptitud del óptimo global: 0

- *Función quartic con ruido*:

$$f_{QNoise}(X) = random[0, 1) + \sum_{i=1}^{n} i x_i^4 \tag{A.10}$$

  - Dimensión: 25
  - Dominio: $[-1{,}28, 1{,}28]$
  - Inicialización: $[-1{,}28, 1{,}28]$
  - Valor de aptitud del óptimo global: 0

- *Función generalizada de Rastrigin*:

$$f_{Ras}(X) = 10n + \sum_{i=1}^{n} x_i^2 - 10cos(2\pi x_i) \tag{A.11}$$

  - Dimensión: 25
  - Dominio: $[-5{,}12, 5{,}12]$
  - Inicialización: $[4, 5]$
  - Valor de aptitud del óptimo global: 0

- *Función generalizada de Rastrigin rotada* (no separable):

$$f_{RRas}(X) = 10n + \sum_{i=1}^{n}(z_i^2 - 10cos(2\pi z_i)) \tag{A.12}$$

$$\text{con } z = Ax \text{ y } A_{ij} = \begin{cases} 4/5 & \text{si } i = j \\ (-1)^{i+1}3/5 & \text{si } |i-j| = 1 \\ 0 & \text{en otro caso} \end{cases}$$

  - Dimensión: 25
  - Dominio: $[-5{,}12, 5{,}12]$
  - Inicialización: $[4, 5]$
  - Valor de aptitud del óptimo global: 0

- *Función generalizada de Rastrigin con costras*:

$$f_{SRas}(X) = 10n + \sum_{i=1}^{n}\left(\left(10^{(i-1)/(n-1)}x_i\right)^2 - 10cos\left(2\pi 10^{(i-1)/(n-1)}x_i\right)\right) \tag{A.13}$$

  - Dimensión: 25
  - Dominio: $[-5{,}12, 5{,}12]$
  - Inicialización: $[4, 5]$
  - Valor de aptitud del óptimo global: 0

- *Función generalizada de Griewank*:

$$f_{Gri}(X) = \frac{1}{4000}\sum_{i=1}^{n}x_i^2 - \prod_{i=1}^{n}cos\left(\frac{x}{\sqrt{i}}\right) + 1 \tag{A.14}$$

  - Dimensión: 15
  - Dominio: $[-600, 600]$
  - Inicialización: $[580, 600]$
  - Valor de aptitud del óptimo global: 0

- *Problema* 2,26 *generalizado de Schwefel*:

$$f_{MSch}(X) = -\sum_{i=1}^{n}x_i sin\left(\sqrt{|x_i|}\right) \tag{A.15}$$

  - Dimensión: 30
  - Dominio: $[-500, 500]$
  - Inicialización: $[-500, 300]$
  - Valor de aptitud del óptimo global: -12,5695

- *Función F10 expandida* (no separable):

$$EF10(X) = F10(x_n, x_1) + \sum_{i=1}^{n-1} F10(x_i, x_{i+1}) \qquad \text{(A.16)}$$

$$\text{con } F10(x, y) = (x^2 + y^2)^{0,25}(sin^2(50(x^2 + y^2)^{0,1}) + 1)$$

  - Dimensión: 15
  - Dominio: $[-100, 100]$
  - Inicialización: $[-100, 100]$
  - Valor de aptitud del óptimo global: 0

- *Función compuesta* $f_{Gri}(f_{Ros}(x))$ (no separable):

$$F8F2(X) = f_{Gri}(f_{Ros}(x_n, x_1)) + \sum_{i=1}^{n-1} f_{Gri}(f_{Ros}(x_i, x_{i+1})) \qquad \text{(A.17)}$$

  - Dimensión: 10
  - Dominio: $[-5, 5]$
  - Inicialización: $[-5, 5]$
  - Valor de aptitud del óptimo global: 0

- *Función de Bohachevsky* (no separable):

$$f_{Boh}(X) = \sum_{i=1}^{n-1}(x_i^2 + 2x_{i+1}^2 - 0{,}3cos(3\pi x_i) - 0{,}4cos(4\pi x_{i+1}) + 0{,}7) \qquad \text{(A.18)}$$

  - Dimensión: 25
  - Dominio: $[-15, 15]$
  - Inicialización: $[10, 15]$
  - Valor de aptitud del óptimo global: 0

- *Función de Schaffer* (no separable):

$$f_{Schaffer}(X) = \sum_{i=1}^{n-1}(x_i^2 + x_{i+1}^2)^{0,25}\left[sin^2(50(x_i^2 + x_{i+1}^2)^{0,1}) + 1\right] \qquad \text{(A.19)}$$

  - Dimensión: 25
  - Dominio: $[-100, 100]$
  - Inicialización: $[50, 100]$
  - Valor de aptitud del óptimo global: 0

Hemos inicializado las poblaciones de los algoritmos con soluciones alejadas de la base de atracción del óptimo global por dos razones (Deb y otros, 2002b):

- Evitar la ventaja que tienen los algoritmos con tendencia inherente a crear soluciones cercanas al centro de los padres.
- Asegurarnos de que los algoritmos deben superar varios mínimos locales antes de alcanzar la base de atracción del óptimo global, cuando tratan con funciones multimodales.

A.2.2. Problemas Reales

En esta sección, describimos los dos problemas reales utilizados en los experimentos del Capítulo 4.

Sistema de Ecuaciones Lineales

En los *sistemas de ecuaciones lineales* (Eshelman y otros, 1997), se pretende obtener un vector x, dada la matriz A y el vector b de la expresión: $Ax = b$. La función usada para este experimento es:

$$P_{sle}(x_1, \ldots, x_n) = \sum_{i=1}^{n} \sum_{j=1}^{n} |(a_{ij} x_i) - b_j| \tag{A.20}$$

Hemos estudiado un problema con 10 variables de decisión. Hemos considerado que el dominio de las variables es el intervalo $[-127, 127]$, y que se utilizan los siguientes datos:

$$A = \begin{pmatrix} 5 & 4 & 5 & 2 & 9 & 5 & 4 & 2 & 3 & 1 \\ 9 & 7 & 1 & 1 & 7 & 2 & 2 & 6 & 6 & 9 \\ 3 & 1 & 8 & 6 & 9 & 7 & 4 & 2 & 1 & 6 \\ 8 & 3 & 7 & 3 & 7 & 5 & 3 & 9 & 9 & 5 \\ 9 & 5 & 1 & 6 & 3 & 4 & 2 & 3 & 3 & 9 \\ 1 & 2 & 3 & 1 & 7 & 6 & 6 & 3 & 3 & 3 \\ 1 & 5 & 7 & 8 & 1 & 4 & 7 & 8 & 4 & 8 \\ 9 & 3 & 8 & 6 & 3 & 4 & 7 & 1 & 8 & 1 \\ 8 & 2 & 8 & 5 & 3 & 8 & 7 & 2 & 7 & 5 \\ 2 & 1 & 2 & 2 & 9 & 8 & 7 & 4 & 4 & 1 \end{pmatrix} \text{ y } b = \begin{pmatrix} 40 \\ 50 \\ 47 \\ 59 \\ 45 \\ 35 \\ 53 \\ 50 \\ 55 \\ 40 \end{pmatrix} \tag{A.21}$$

Claramente, el mejor valor para este problema es $P_{sle}(x^*) = 0$. En los sistemas de ecuaciones lineales es fácil controlar la ligadura de parámetros, esto es, su no linealidad. Además, esta no linealidad no se deteriora al incrementar el número de parámetros utilizados.

Las poblaciones de los algoritmos se inicializaron en el intervalo $[-120, -100]$.

Identificación de Parámetros para la Modulación por Frecuencia del Sonido

En el *problema de la identificación de parámetros para la modulación por frecuencia* (Tsutsui y Fujimoto, 1993), se pretende encontrar el valor de seis parámetros (a_1, w_1, a_2, w_2, a_3 y w_3) del modelo representado por:

$$y(t) = a_1 \cdot sin(w_1 \cdot t \cdot \theta + a_2 \cdot sin(w_2 \cdot t \cdot \theta + a_3 \cdot sin(w_3 \cdot t \cdot \theta))), \tag{A.22}$$

con $\theta = (2 \cdot \pi/100)$. El valor de aptitud de una solución se define como la suma de los errores al cuadrado entre los datos obtenidos y el siguiente modelo de

datos:

$$P_{fms} = (a_1, w_1, a_2, w_2, a_3, w_3) = \sum_{t=0}^{100} (y(t) - y_0(t))^2, \tag{A.23}$$

donde el modelo de datos viene dado por la siguiente fórmula:

$$y_0(t) = 1{,}0 \cdot sin(5{,}0 \cdot t \cdot \theta + 1{,}5 \cdot sin(4{,}8 \cdot t \cdot \theta + 2{,}0 \cdot sin(4{,}9 \cdot t \cdot \theta))). \tag{A.24}$$

El dominio de todos los parámetros es el rango $[-6{,}4, -6{,}35]$ (la población inicial se generó uniformemente en el mismo rango). Se trata de un problema multimodal y altamente complejo, teniendo una fuerte epítasis, con mínimo igual a $P_{fms}(x^*) = 0$.

Apéndice B

Resultados

En este apéndice, recopilamos los resultados de los algoritmos utilizados en los Capítulos 3 y 4. La Sección B.1 muestra los resultados de los algoritmos estudiados en el Capítulo 3, los cuales optimizan los problemas combinatorios con codificación binaria descritos en la Sección A.1. Por otro lado, la Sección B.2 presenta los resultados de los algoritmos para optimización continua estudiados en el Capítulo 4. Los problemas tratados por estos algoritmos se describen en la Sección A.2.

B.1. Resultados en Problemas Binarios

Las siguientes tablas (Tablas B.1 a B.6) muestran los resultados de los diferentes algoritmos BLMA (Sección 3.3), BLI-0,1, BLI-0,25, BLI-0,5, BLI-0,75 (Sección 3.4) y BVV (Sección 3.5), sobre los problemas de prueba utilizados. Hemos añadido el resultado de un test estadístico (el test de *Student*, cuando se cumple las condiciones de normalidad e igualdad de varianzas, y el de *Mann-Whitney* en otro caso, ambos descritos en el Apéndice C) para comprobar si existen diferencias significativas en el rendimiento de las MH basadas en métodos clásicos de BL y las basadas en AGLB, en cada uno de los problemas de prueba. El resultado indica:

- Un signo más (+): el rendimiento de la MH basada en AGLB es superior al de la MH basada en BL correspondiente.
- Un signo menos (−): el rendimiento de la MH basada en AGLB es inferior al de la MH basada en BL correspondiente.
- Un signo de aproximación (∼): no se detectaron diferencias significativas.

Además, hemos añadido las tres últimas filas que cuentan el número de funciones en las que la MH basada en AGLB produce resultados mejores, peores o sin diferencia estadística que los obtenidos por la MH basada en BL correspondiente.

La Tabla B.7 muestra el número de reinicios con éxito de los algoritmos BLMA, mejor BLI y BVV para cada método de refinamiento (media de 50

Tabla B.1: Resultados de BLMA

Problema	**BL-Mejor**	**BL-PM**	**BL-Kopt**	**BL-PKopt**	**AGLB**
Deceptive	386−	381∼	390−	375+	380
Trap	219+	213+	220∼	202+	220
M-Sat(100,1200,3)	0,955+	0,958+	0,959 ∼	0,959 ∼	0,960
M-Sat(100,2400,3)	0,935+	0,936 ∼	0,937 ∼	0,937 ∼	0,937
NKLand(48,4)	0,760+	0,762+	0,771 ∼	0,767+	0,774
NKLand(48,12)	0,742−	0,745−	0,763−	0,749−	0,739
PPeaks(50,50)	1∼	1∼	1,000 ∼	0,999+	1
PPeaks(50,100)	0,997+	1,000 ∼	0,993+	0,978+	1
PPeaks(50,150)	0,990+	1,000 ∼	0,971+	0,948+	1
PPeaks(50,200)	0,973+	1,000 ∼	0,964+	0,915+	1,000
PPeaks(50,250)	0,937+	0,999 ∼	0,919+	0,877+	0,999
PPeaks(100,100)	0,997+	1∼	0,990+	0,980+	1
BQP(50)	2094∼	2098∼	2098∼	2098∼	2098
BQP(100)	7849+	7899+	7938∼	7886+	7946
BQP(250)	45168+	45545+	45516+	45511+	45563
BQP(500)	114792+	115939∼	115717+	116251−	115923
Maxcut(G10)	1762+	1819+	1871+	1887+	1897
Maxcut(G12)	421+	437+	534−	529−	508
Maxcut(G17)	2920+	2931+	2991+	2987+	3008
Maxcut(G18)	843+	866+	916+	929+	953
Maxcut(G19)	759+	782+	831+	842+	869
Maxcut(G43)	6410+	6460+	6506+	6548+	6572
+	18	11	12	16	
∼	2	10	7	3	
−	2	1	3	3	

Tabla B.2: Resultados de los algoritmos BLI-0,1 (los resultados se han comparado con los de BLI-0,5-AGLB)

Problema	**BL-Mejor**	**BL-PM**	**BL-Kopt**	**BL-PKopt**	**AGLB**
Deceptive	390−	387−	390−	373+	388−
Trap	220∼	220∼	220∼	196+	220∼
M-Sat(100,1200,3)	0,957+	0,958+	0,958+	0,958 ∼	0,957+
M-Sat(100,2400,3)	0,935+	0,937 ∼	0,936+	0,936+	0,936+
NKLand(48,4)	0,773 ∼	0,774 ∼	0,769+	0,769+	0,773 ∼
NKLand(48,12)	0,761−	0,768−	0,768−	0,753−	0,763−
PPeaks(50,50)	0,892+	0,877+	0,959+	0,847+	0,908+
PPeaks(50,100)	0,871+	0,843+	0,930+	0,830+	0,873+
PPeaks(50,150)	0,861+	0,857+	0,931+	0,855+	0,883+
PPeaks(50,200)	0,867+	0,831+	0,922+	0,838+	0,869+
PPeaks(50,250)	0,855+	0,821+	0,884+	0,827+	0,858+
PPeaks(100,100)	0,881+	0,859+	0,927+	0,861+	0,914+
BQP(50)	2080+	2098∼	2098∼	2095∼	2093∼
BQP(100)	7871+	7915+	7913+	7867+	7887+
BQP(250)	45277+	45590∼	45534∼	45490+	45520∼
BQP(500)	115422+	116454−	115651+	116211−	115808∼
Maxcut(G10)	1829+	1921−	1869+	1886+	1872+
Maxcut(G12)	482+	517−	535−	529−	522−
Maxcut(G17)	2969+	2992+	2991+	2986+	3002+
Maxcut(G18)	900+	950+	920+	929+	935+
Maxcut(G19)	810+	861+	830+	839+	846+
Maxcut(G43)	6472+	6582−	6504+	6545+	6548+
+	18	11	16	17	14
∼	2	5	3	2	5
−	2	6	3	3	3

Tabla B.3: Resultados de los algoritmos BLI-0,25 (los resultados se han comparado con los de BLI-0,5-AGLB)

Problema	**BL-Mejor**	**BL-PM**	**BL-Kopt**	**BL-PKopt**	**AGLB**
Deceptive	390−	383−	390−	375+	385−
Trap	220∼	220∼	220∼	197+	220∼
M-Sat(100,1200,3)	0,957+	0,959 ∼	0,959 ∼	0,959 ∼	0,958+
M-Sat(100,2400,3)	0,936 ∼	0,937 ∼	0,937 ∼	0,937 ∼	0,936 ∼
NKLand(48,4)	0,773 ∼	0,773 ∼	0,773 ∼	0,769 ∼	0,774 ∼
NKLand(48,12)	0,752−	0,753−	0,764−	0,751−	0,754−
PPeaks(50,50)	0,996+	0,997+	0,980+	0,958+	0,991+
PPeaks(50,100)	0,883+	0,908+	0,926+	0,840+	0,924+
PPeaks(50,150)	0,861+	0,861+	0,923+	0,855+	0,891+
PPeaks(50,200)	0,867+	0,831+	0,921+	0,838+	0,869+
PPeaks(50,250)	0,855+	0,821+	0,886+	0,827+	0,858+
PPeaks(100,100)	0,905+	0,929+	0,929+	0,871+	0,958+
BQP(50)	2098∼	2098∼	2098∼	2096∼	2098∼
BQP(100)	7925+	7934∼	7940∼	7868+	7903+
BQP(250)	45379+	45584∼	45534∼	45490+	45524∼
BQP(500)	115802∼	116537−	115680∼	116214−	115824∼
Maxcut(G10)	1823+	1916∼	1869+	1886+	1877+
Maxcut(G12)	449+	471+	535−	529−	526−
Maxcut(G17)	2940+	2954+	2991+	2986+	3007∼
Maxcut(G18)	885+	922+	920+	929+	947+
Maxcut(G19)	795+	844+	830+	839+	855+
Maxcut(G43)	6452+	6546+	6504+	6545+	6560+
+	15	11	11	15	12
∼	5	8	8	4	7
−	2	3	3	3	3

Tabla B.4: Resultados de los algoritmos BLI-0,5

Problema	**BL-Mejor**	**BL-PM**	**BL-Kopt**	**BL-PKopt**	**AGLB**
Deceptive	386−	381−	390−	376+	380
Trap	219∼	212+	220∼	201+	220
M-Sat(100,1200,3)	0,955+	0,957+	0,959 ∼	0,959 ∼	0,960
M-Sat(100,2400,3)	0,935+	0,936 ∼	0,937 ∼	0,937 ∼	0,937
NKLand(48,4)	0,761+	0,762+	0,770 ∼	0,767+	0,774
NKLand(48,12)	0,743−	0,748−	0,765−	0,749−	0,738
PPeaks(50,50)	1∼	1∼	1∼	0,997+	1
PPeaks(50,100)	0,997+	1∼	0,994+	0,975+	1
PPeaks(50,150)	0,989+	1,000 ∼	0,981+	0,960+	1
PPeaks(50,200)	0,980+	1,000 ∼	0,962+	0,913+	1,000
PPeaks(50,250)	0,946+	0,999 ∼	0,918+	0,896+	1,000
PPeaks(100,100)	0,998+	1,000 ∼	0,993+	0,977+	1,000
BQP(50)	2095∼	2098∼	2098∼	2095∼	2098
BQP(100)	7836+	7897+	7944∼	7884+	7946
BQP(250)	45104+	45524+	45534∼	45488+	45551
BQP(500)	114815+	115890∼	115686∼	116261−	115862
Maxcut(G10)	1764+	1822+	1869+	1886+	1908
Maxcut(G12)	422+	437+	535−	529−	510
Maxcut(G17)	2920+	2931+	2991+	2986+	3009
Maxcut(G18)	845+	868+	920+	929+	956
Maxcut(G19)	764+	781+	830+	839+	868
Maxcut(G43)	6410+	6456+	6504+	6545+	6571
+	17	11	10	16	
∼	3	9	9	3	
−	2	2	3	3	

Tabla B.5: Resultados de los algoritmos BLI-0,75 (los resultados se han comparado con los de BLI-0,5-AGLB)

Problema	**BL-Mejor**	**BL-PM**	**BL-Kopt**	**BL-PKopt**	**AGLB**
Deceptive	382−	380∼	390−	376+	380∼
Trap	211+	205+	220∼	206+	220∼
M-Sat(100,1200,3)	0,953+	0,956+	0,959 ∼	0,959 ∼	0,958+
M-Sat(100,2400,3)	0,933+	0,935+	0,937 ∼	0,936 ∼	0,936+
NKLand(48,4)	0,753+	0,755+	0,770 ∼	0,766+	0,767+
NKLand(48,12)	0,742−	0,749−	0,762−	0,747−	0,736 ∼
PPeaks(50,50)	1,000 ∼	1,000 ∼	0,999+	0,995+	1,000 ∼
PPeaks(50,100)	0,997+	1,000 ∼	0,989+	0,981+	0,999 ∼
PPeaks(50,150)	0,988+	0,999+	0,975+	0,948+	0,997+
PPeaks(50,200)	0,973+	0,997+	0,959+	0,916+	0,997+
PPeaks(50,250)	0,946+	0,996+	0,922+	0,894+	0,997+
PPeaks(100,100)	0,995+	0,999+	0,991+	0,977+	1,000 ∼
BQP(50)	2006+	2059+	2098∼	2094∼	2095∼
BQP(100)	7699+	7827+	7946∼	7880+	7920+
BQP(250)	44880+	45452+	45534∼	45484+	45583∼
BQP(500)	113823+	115207+	115638+	116095−	116024∼
Maxcut(G10)	1821+	1916∼	1869+	1886+	1893+
Maxcut(G12)	446+	472+	535−	529−	528−
Maxcut(G17)	2939+	2953+	2991+	2986+	3010∼
Maxcut(G18)	880+	924+	920+	929+	946+
Maxcut(G19)	800+	842+	830+	839+	859+
Maxcut(G43)	6448+	6551+	6504+	6545+	6571∼
+	19	17	12	16	10
∼	1	4	7	3	11
−	2	1	3	3	1

Tabla B.6: Resultados de los algoritmos BVV

Problema	**BL-Mejor**	**BL-PM**	**BL-Kopt**	**BL-PKopt**	**AGLB**
Deceptive	389−	383∼	390−	376+	383
Trap	220∼	218+	220∼	205+	220
M-Sat(100,1200,3)	0,957+	0,959 ∼	0,958 ∼	0,959 ∼	0,959
M-Sat(100,2400,3)	0,936 ∼	0,937 ∼	0,937 ∼	0,937 ∼	0,937
NKLand(48,4)	0,769 ∼	0,773 ∼	0,773 ∼	0,768+	0,774
NKLand(48,12)	0,755 ∼	0,754 ∼	0,764−	0,751 ∼	0,752
PPeaks(50,50)	1∼	1∼	0,999+	0,990+	1
PPeaks(50,100)	0,994+	1,000 ∼	0,976+	0,919+	1,000
PPeaks(50,150)	0,981+	0,998 ∼	0,936+	0,855+	0,999
PPeaks(50,200)	0,946+	0,997 ∼	0,921+	0,838+	0,999
PPeaks(50,250)	0,869+	0,994 ∼	0,884+	0,827+	0,996
PPeaks(100,100)	0,994+	1,000 ∼	0,978+	0,933+	1,000
BQP(50)	2098∼	2098∼	2098∼	2096∼	2098
BQP(100)	7880+	7909+	7934∼	7876+	7951
BQP(250)	45356+	45573+	45534+	45490+	45589
BQP(500)	115475+	116477−	115650+	116228∼	116079
Maxcut(G10)	1826+	1920−	1869+	1886+	1905
Maxcut(G12)	458+	501+	535−	529−	523
Maxcut(G17)	2953+	2978+	2991+	2986+	3006
Maxcut(G18)	894+	937+	920+	929+	948
Maxcut(G19)	806+	854∼	830+	839+	857
Maxcut(G43)	6472+	6569∼	6504+	6545+	6572
+	15	6	13	16	0
∼	6	14	6	5	0
−	1	2	3	1	0

ejecuciones). Estos datos se utilizaron en el estudio de la Sección 3.5.1. Se ha destacado en negrita el mayor valor de cada fila.

Tabla B.7: Número de reinicios con éxito por cada MH basada en BL

	BL-Mejor			**BL-PM**			**BL-Kopt**			**BL-PKopt**			**AGLB**		
Problema	**BLMA**	**BLI-**0,25	**BVV**	**BLMA**	**BLI-**0,1	**BVV**	**BLMA**	**BLI-**0,5	**BVV**	**BLMA**	**BLI-**0,5	**BVV**	**BLMA**	**BLI-**0,5	**BVV**
1	3,2	5,5	5,2	3,8	8,1	5,0	0,0	0,0	0,0	1,9	2,2	2,5	3,8	3,7	5,1
2	3,2	3,6	4,0	4,5	6,9	6,4	0,0	0,0	0,0	2,2	2,4	3,0	5,3	5,3	6,3
3	2,5	5,0	4,8	3,7	6,7	6,2	1,0	1,0	1,3	1,1	0,9	1,0	6,1	6,1	7,3
4	2,1	4,4	4,2	4,5	5,0	5,0	0,9	1,0	1,2	1,0	1,2	1,5	6,7	6,3	6,5
5	4,2	7,3	6,9	6,0	10,0	8,4	2,5	2,7	3,0	2,3	2,5	2,9	9,6	9,3	9,8
6	5,2	6,2	6,2	5,4	8,2	6,9	3,0	2,8	3,2	2,6	3,1	2,6	6,5	6,0	7,1
7	2,2	1,8	2,3	2,8	0,2	2,8	1,9	2,0	1,8	2,7	2,4	2,3	3,1	3,3	3,2
8	2,2	0,1	2,1	3,0	0,0	3,0	1,5	1,6	0,9	1,6	1,7	0,7	3,0	2,9	3,7
9	2,0	0,0	1,3	3,1	0,0	2,7	1,1	0,9	0,2	1,1	1,0	0,0	3,5	3,1	3,3
10	1,1	0,0	0,7	3,1	0,0	2,5	0,7	0,5	0,0	0,4	0,5	0,0	3,1	2,6	3,4
11	0,8	0,0	0,2	3,2	0,0	2,8	0,4	0,3	0,0	0,4	0,5	0,0	3,1	2,9	3,1
12	2,5	0,2	2,6	3,6	0,0	3,3	1,5	1,6	1,1	1,8	1,6	0,7	3,7	3,3	3,9
13	3,3	3,4	3,6	4,0	3,8	3,5	0,3	0,4	0,4	1,3	1,5	1,4	4,2	3,6	4,0
14	3,2	4,3	3,3	4,6	5,5	5,4	1,2	1,2	1,0	1,6	1,6	1,6	6,4	6,7	7,7
15	0,9	2,2	2,4	3,4	5,2	4,3	0,0	0,0	0,0	0,2	0,1	0,1	5,6	5,8	5,7
16	1,6	5,8	6,7	4,6	11,7	9,5	0,2	0,3	0,4	0,5	0,6	0,9	15,3	14,6	17,3
17	1,0	2,9	6,6	3,9	18,3	14,1	0,0	0,0	0,0	0,0	0,0	0,0	26,8	29,5	39,5
18	1,3	3,5	4,7	4,3	20,1	14,7	0,0	0,0	0,0	0,0	0,0	0,0	13,2	14,3	23,3
19	1,2	2,9	5,4	4,3	17,5	11,8	0,0	0,0	0,0	0,0	0,0	0,1	20,5	20,5	32,0
20	1,4	3,5	7,8	4,6	25,3	16,7	0,0	0,0	0,0	0,0	0,0	0,0	25,9	28,0	38,4
21	1,2	3,4	7,6	4,4	22,8	17,0	0,0	0,0	0,0	0,0	0,0	0,0	28,1	26,6	36,8
22	0,5	1,8	5,1	4,0	22,8	15,6	0,0	0,0	0,0	0,0	0,0	0,0	30,2	30,1	45,6

B.2. Resultados en Optimización Continua

En esta sección, presentamos los resultados de los algoritmos utilizados en el Capítulo 4. La Sección B.2.1 contiene los resultados de las algoritmos PDS-S&E con diferentes combinaciones de valores de sus parámetros N_F y N_M (ver Sección 4.4). La Sección B.2.2 muestra los resultados de la comparación de algoritmos propuestos en la literatura de la Sección 4.5.3

B.2.1. Resultados del Algoritmo PDS-S&E

La Tabla B.8 muestra los resultados del algoritmo PDS-S&E usando diferentes valores para sus parámetros N_F y N_M en las funciones de prueba: f_{esf}, f_{Ros}, f_{Sch}, f_{Ras} y f_{Gri}, y el problema real P_{sle}.

B.2.2. Resultados de la Comparación de GL-25 con Propuestas de la Literatura

Las Tablas B.9 a B.11 muestran los resultados de los algoritmos comparados en la Sección 4.5.3. Se ha añadido el resultado del test de *Student*, cuando se cumplen las condiciones de normalidad e igualdad de varianzas, o el de *Mann-Whitney* en otro caso, para detectar diferencias significativas entre los resultados de GL-25 y el algoritmo correspondiente, en cada una de las funciones de prueba y problemas reales.

Tabla B.8: Resultados de PDS-S&E con las diferentes valores para N_F y N_M

N_F	N_M	f_{esf}	f_{Ros}	f_{Sch}	f_{Ras}	f_{Gri}	P_{sle}
1	25	1,90e+002	3,36e+05	5,70e+06	5,04e+02	5,90e+02	7,59e+03
1	50	3,19e+001	1,85e+04	1,94e+06	4,61e+02	1,46e+02	4,11e+03
1	100	2,86e−005	4,92e+01	2,29e+05	4,41e+02	2,80e+01	2,21e+03
1	200	2,46e−144	2,13e+01	2,61e+04	4,34e+02	3,57e+00	9,85e+02
1	300	4,92e−116	1,59e+01	8,34e+03	4,16e+02	5,21e+00	7,03e+02
1	400	5,10e−095	1,95e+01	3,06e+03	4,19e+02	7,44e−02	3,90e+02
5	25	9,17e+001	1,75e+05	1,17e+05	4,00e+02	3,14e+02	2,70e+03
5	50	6,21e−146	2,89e+01	2,81e+03	3,50e+02	1,04e+01	5,94e+02
5	100	9,98e−187	1,56e+00	1,74e−09	2,90e+02	1,27e−02	1,64e+02
5	200	2,76e−129	2,33e+00	3,09e−09	2,22e+02	1,41e−02	8,86e+01
5	300	8,99e−098	7,67e+00	1,05e−06	2,06e+02	1,18e−02	5,09e+01
5	400	2,53e−079	1,16e+01	7,60e−05	1,85e+02	1,49e−02	3,82e+01
25	25	9,24e−007	3,27e+03	3,69e+03	2,02e+02	2,22e+01	5,78e+02
25	50	1,64e−130	1,33e+01	1,01e−12	1,34e+02	1,07e−02	1,47e+02
25	100	3,31e−104	9,48e+00	2,68e−09	6,62e+01	6,70e−03	6,50e+01
25	200	1,05e−074	9,80e+00	6,66e−05	4,48e+01	2,56e−03	3,85e+01
25	300	1,24e−059	1,08e+01	3,82e−03	2,37e+01	3,45e−03	2,10e+01
25	400	4,18e−050	1,23e+01	3,43e−02	1,84e+01	2,51e−03	1,46e+01
50	25	5,98e−076	2,53e+01	1,74e−03	1,21e+02	2,67e+00	4,50e+02
50	50	6,19e−095	1,46e+01	3,54e−10	4,97e+01	6,45e−03	8,96e+01
50	100	1,29e−072	1,37e+01	3,85e−06	2,39e+01	4,73e−03	5,61e+01
50	200	1,15e−053	1,38e+01	4,15e−03	1,26e+01	2,86e−03	2,77e+01
50	300	9,83e−044	1,45e+01	9,00e−02	1,04e+01	2,71e−03	1,43e+01
50	400	1,82e−037	1,51e+01	5,36e−01	7,80e+00	1,38e−03	8,59e+00
100	25	1,15e−088	2,45e+01	2,45e−09	5,37e+01	1,84e−02	2,50e+02
100	50	1,29e−071	1,55e+01	1,00e−06	1,94e+01	5,07e−03	1,19e+02
100	100	2,28e−049	1,60e+01	2,20e−03	1,10e+01	3,30e−03	3,32e+01
100	200	5,59e−037	1,65e+01	2,34e−01	6,27e+00	1,58e−03	1,79e+01
100	300	9,01e−031	1,71e+01	1,45e+00	5,32e+00	9,37e−04	1,33e+01
100	400	1,06e−026	1,75e+01	4,58e+00	4,24e+00	1,18e−03	1,05e+01
200	25	2,19e−078	2,53e+01	3,96e−07	1,86e+01	6,99e−03	1,49e+02
200	50	8,56e−052	1,65e+01	6,07e−04	7,92e+00	4,04e−03	6,90e+01
200	100	6,31e−035	1,73e+01	9,10e−02	5,15e+00	2,76e−03	2,02e+01
200	200	1,22e−024	1,83e+01	3,40e+00	3,58e+00	1,53e−03	1,61e+01
200	300	9,95e−021	1,88e+01	1,45e+01	6,64e+00	1,04e−03	1,07e+01
200	400	2,95e−018	1,91e+01	3,12e+01	1,92e+01	4,93e−04	5,45e+00
300	25	3,22e−068	1,85e+01	1,37e−05	1,13e+01	4,78e−03	1,28e+02
300	50	2,72e−044	1,70e+01	1,40e−02	6,81e+00	3,74e−03	5,04e+01
300	100	1,09e−027	1,81e+01	6,89e−01	3,33e+00	1,23e−03	1,94e+01
300	200	5,19e−020	1,89e+01	1,18e+01	6,32e+00	9,86e−04	1,16e+01
300	300	3,17e−016	1,96e+01	3,76e+01	3,80e+01	5,42e−04	9,13e+00
300	400	2,53e−014	1,98e+01	7,35e+01	6,58e+01	5,92e−04	7,45e+00
400	25	3,47e−061	1,87e+01	2,96e−04	8,08e+00	7,44e−03	1,10e+02
400	50	1,28e−039	1,74e+01	9,27e−02	4,16e+00	2,86e−03	3,73e+01
400	100	1,71e−024	1,85e+01	2,53e+00	2,60e+00	3,10e−03	2,43e+01
400	200	7,73e−017	1,94e+01	2,35e+01	3,45e+01	6,41e−04	1,01e+01
400	300	4,93e−014	1,99e+01	7,18e+01	6,72e+01	3,48e−04	8,83e+00
400	400	5,35e−012	2,03e+01	1,38e+02	9,71e+01	1,77e−03	7,84e+00

Tabla B.9: Resultados de los algoritmos en las funciones f_{esf} a f_{Sch}

	f_{esf}	f_{cig}	$f_{Sch2,22}$	$f_{Sch2,21}$	f_{Ros}	f_{Sch}
SPC-PNX-40	5,42e−040+	8,49e−37+	4,95e−25+	8,92e+01+	2,22e+01+	2,43e−01+
SPC-PNX-60	2,37e−029+	1,35e−27+	1,20e−17+	8,62e+01+	1,99e+01+	9,75e−01+
SPC-PNX-100	8,62e−018+	4,91e−15+	4,65e−10+	7,62e+01+	2,20e+01+	1,62e+01+
SPC-PNX-200	2,94e−008+	5,91e−05+	2,96e−04+	6,74e+01+	2,09e+01+	3,55e+02+
G3-PCX-1	4,21e−203−	6,81e+01+	1,63e+02+	9,26e+01+	9,83e+00+	3,74e−29−
G3-PCX-2	3,55e−178−	5,41e+01+	1,56e+02+	9,23e+01+	3,30e+00+	5,78e−31−
G3-PCX-3	8,10e−148−	7,40e+01+	1,54e+02+	9,14e+01+	4,84e−01∼	8,88e−32−
G3-PCX-4	4,47e−127+	7,60e+01+	1,46e+02+	9,07e+01+	7,18e−01∼	2,95e−30−
PEMLA-1	1,15e−003∼	1,62e+03+	1,47e+00+	9,77e+01+	4,05e+01+	1,35e+03+
PEMLA-2	1,55e+000+	7,47e+04+	9,37e+00+	9,38e+01+	9,92e+02∼	1,51e+04+
PEML-1	3,31e−002+	1,08e+03+	2,65e+00+	9,89e+01+	4,17e+01+	1,70e+03+
PEML-2	3,60e+000+	1,39e+05+	2,21e+01+	9,45e+01+	7,50e+02+	3,21e+04+
EE-AMC	3,18e−293−	5,2e−243−	1,3e−124−	3,17e−05−	1,59e−01−	8,7e−267−
AMCR-OACC	9,47e−100+	5,73e−94+	3,56e−41+	7,67e+01+	2,85e+00+	9,48e−07+
CHC-SW-25	3,41e−322−	5,01e−05+	7,02e−05+	8,42e+01+	5,59e+00+	1,91e−31−
CHC-SW-50	3,80e−322−	2,98e−13+	4,79e−10+	8,31e+01+	8,99e+00+	8,51e−22−
CHC-SW-75	5,30e−205−	1,66e−22+	1,13e−15+	8,32e+01+	1,31e+01+	3,97e−12−
ESM	1,46e−023+	3,48e−18+	6,73e−09+	2,12e+01−	9,40e+00+	1,80e−05+
OPAE	2,21e−028+	9,68e−26+	4,05e−17+	9,44e+01+	6,69e+00+	5,80e+02+
OPAE-30	1,03e−011+	2,71e−07+	1,07e−05+	9,61e+01+	2,34e+01+	2,47e+03+
ED	2,86e+000+	1,04e+05+	5,21e+01+	9,84e+01+	2,94e+03+	1,07e+05+
ED-60	6,81e−024+	6,68e−10+	2,63e−11+	8,97e+01+	3,38e+00+	2,28e−04+
ED-100	1,03e−011+	1,83e−07+	4,34e−05+	8,71e+01+	8,52e+00+	2,84e+00+
GL-25	4,36e−147	1,03e−122	9,85e−77	3,25e+01	8,22e−01	3,43e−09

Tabla B.10: Resultados de los algoritmos en las funciones f_{QNoise} a f_{Gri}

	f_{QNoise}	f_{Ras}	f_{RRas}	f_{SRas}	f_{Gri}	f_{MSch}
SPC-PNX-40	8,33e−03+	1,29e+02+	1,66e+02+	1,05e+02+	2,62e−02+	-9,16e+3∼
SPC-PNX-60	7,80e−03+	7,30e+01+	8,81e+01+	6,09e+01+	1,96e−02+	-9,47e+3∼
SPC-PNX-100	9,81e−03+	3,68e+01+	3,80e+01+	3,12e+01+	2,02e−02+	-1,01e+4−
SPC-PNX-200	1,53e−02+	3,13e+01+	4,76e+01+	2,51e+01+	9,25e−03+	-1,12e+4−
G3-PCX-1	1,03e+00+	4,86e+02+	4,78e+02+	1,56e+03+	1,44e−01+	-7,89e+3+
G3-PCX-2	5,83e−01+	4,85e+02+	4,72e+02+	1,18e+03+	7,32e−02+	-7,97e+3+
G3-PCX-3	3,41e−01+	4,88e+02+	4,76e+02+	9,94e+02+	4,98e−02+	-8,20e+3+
G3-PCX-4	2,63e−01+	4,83e+02+	4,73e+02+	7,42e+02+	4,51e−02+	-8,25e+3+
PEMLA-1	5,92e−02+	4,84e+01+	7,72e+01+	5,92e+01+	6,79e−02+	-9,82e+3−
PEMLA-2	9,87e−02+	4,59e+01+	1,26e+02+	1,09e+02+	8,66e+01+	-8,80e+3+
PEML-1	4,55e−02+	6,94e+01+	9,09e+01+	1,18e+02+	1,12e−01+	-9,31e+3∼
PEML-2	5,14e−02+	8,39e+01+	1,65e+02+	2,40e+02+	2,38e+02+	-7,30e+3+
EE-AMC	2,23e−01+	5,24e+01+	5,15e+01+	6,77e+01+	4,14e−03+	-6,89e+3+
AMCR-OACC	5,55e−03+	1,24e+01∼	1,93e+01∼	1,05e+01−	4,71e−02+	-1,06e+4−
CHC-SW-25	1,79e−02+	2,61e+01+	3,60e+01+	2,56e+01+	5,27e−03+	-1,11e+4−
CHC-SW-50	1,10e−02+	2,72e+01+	3,42e+01+	2,29e+01+	4,93e−03+	-1,11e+4−
CHC-SW-75	7,93e−03+	2,15e+01+	2,76e+01+	2,16e+01+	5,91e−03+	-1,08e+4−
ESM	2,37e−02+	1,99e−02−	2,46e+01+	0,00e+00−	3,14e−02+	-5,58e+3+
OPAE	6,12e−03+	7,20e+00−	3,64e+01+	4,40e+00−	9,86e−04+	-1,20e+4−
OPAE-30	1,05e−02+	5,57e−01−	3,29e+01+	2,32e−06−	6,09e−09+	-1,25e+4−
ED	6,16e−01+	1,63e+02+	1,93e+02+	2,41e+02+	3,37e−01+	-6,28e+3+
ED-60	7,00e−03+	1,23e+02+	1,30e+02+	1,23e+02+	4,43e−04+	-6,70e+3+
ED-100	1,40e−02+	1,31e+02+	1,39e+02+	1,32e+02+	3,24e−03+	-5,85e+3+
GL-25	1,67e−03	1,38e+01	1,90e+01	1,35e+01	2,26e−18	-9,36e+3

Tabla B.11: Resultados de los algoritmos en las funciones P_{sle} a $f_{Schaffer}$

	P_{sle}	$EF10$	$F8F2$	P_{Sound}	f_{Boh}	$f_{Schaffer}$
SPC-PNX-40	2,64e+02+	3,31e+00+	8,48e−01∼	8,80e+00+	2,41e+00+	2,65e+01+
SPC-PNX-60	1,71e+02+	4,10e−01+	7,78e−01∼	7,26e+00+	8,17e−01+	4,20e+00+
SPC-PNX-100	1,11e+02+	5,65e−03−	9,34e−01+	3,96e+00+	1,14e−01+	1,03e−01−
SPC-PNX-200	4,32e+01+	2,86e−02−	1,08e+00+	1,86e+00+	9,42e−06+	2,06e+00−
G3-PCX-1	3,07e+02+	9,62e+01+	5,07e+00+	2,20e+01+	1,53e+01+	2,42e+02+
G3-PCX-2	1,89e+02+	9,50e+01+	3,69e+00+	2,21e+01+	1,33e+01+	2,41e+02+
G3-PCX-3	1,30e+02+	9,07e+01+	2,90e+00+	2,28e+01+	1,27e+01+	2,39e+02+
G3-PCX-4	1,22e+02+	8,58e+01+	2,54e+00+	2,16e+01+	1,25e+01+	2,43e+02+
PEMLA-1	3,62e+02+	4,14e+00+	6,91e−01∼	1,39e+01+	2,16e+00+	5,26e+01+
PEMLA-2	1,33e+03+	5,93e+01+	1,74e+00+	2,22e+01+	1,59e+02+	1,39e+02+
PEML-1	1,73e+02+	8,07e+00+	8,63e−01∼	7,88e+00+	6,60e+00+	5,20e+01+
PEML-2	1,76e+03+	8,30e+01+	9,20e−01+	2,18e+01+	3,55e+02+	1,62e+02+
EE-AMC	3,23e−13−	2,02e+01+	1,13e+00+	2,24e+01+	2,65e+00+	3,24e+01+
AMCR-OACC	1,87e+02+	5,62e−01+	3,92e−01−	7,73e+00+	4,13e−02+	1,09e+01+
CHC-SW-25	8,13e+01+	1,31e−02−	9,25e−01+	1,59e+00+	1,46e−14+	2,04e+00−
CHC-SW-50	6,98e+01+	4,43e−07−	8,80e−01+	6,63e−01∼	7,70e−15+	6,73e−02−
CHC-SW-75	8,60e+01+	3,88e−07−	8,78e−01+	2,51e−01∼	0,00e+00∼	1,30e−02−
ESM	1,52e+03+	9,06e−09−	4,23e−01−	2,46e+01+	0,00e+00∼	2,96e−04−
OPAE	1,37e+02+	9,86e−04−	1,23e−01−	4,16e+00+	0,00e+00∼	1,62e−02−
OPAE-30	1,90e+02+	1,86e−02−	6,06e−02−	2,58e+00+	2,76e−09+	2,36e+00−
ED	7,10e+02+	9,58e+00+	1,46e+00+	1,27e+01+	8,11e+01+	1,33e+02+
ED-60	1,26e−01−	5,62e−08−	1,58e+00+	1,52e+00∼	0,00e+00∼	3,54e−02−
ED-100	1,77e+00−	1,14e−02−	1,70e+00+	1,66e+00+	2,05e−08+	1,81e+00−
GL-25	4,70e+00	1,99e−01	7,48e−01	2,96e−01	-1,19e−15	2,70e+00

Apéndice C

Análisis Estadístico

En la actualidad, el uso de análisis estadísticos para comparar resultados de diferentes algoritmos se está convirtiendo en una necesidad (Garcia y otros, 2008), y así lo están manifestando muchos revisores.

Sheskin (2000) hace una distinción entre tests paramétricos y no paramétricos que se basa en el nivel de medida representado por los datos que se van a analizar. La Sección C.1 se centra en los tests paramétricos utilizados. La Sección C.2, por su parte, describe los tests no paramétricos.

C.1. Tests Paramétricos

Un test paramétrico es aquél que utiliza datos con valores reales pertenecientes a un intervalo. Esto no implica que siempre que dispongamos de este tipo de datos haya que usar un test paramétrico. Puede darse el caso de que una o más suposiciones iniciales para el uso de los tests paramétricos se incumplan, haciendo que el análisis estadístico pierda credibilidad.

C.1.1. Condiciones para su Aplicación

Para utilizar los tests paramétricos es necesario que se cumplan las siguientes condiciones (Sheskin, 2000; Zar, 1999):

- *Independencia*: En estadística, dos sucesos son independientes cuando el que haya ocurrido uno de ellos no modifica la probabilidad de ocurrencia del otro. Esta condición, en el caso de comparar los resultados de dos algoritmos, suele ser siempre cierta.

- *Normalidad*: Una observación es normal cuando su comportamiento sigue una distribución normal o de *Gauss* con una determinada media μ y varianza σ. Un test de normalidad sobre una muestra nos indica la presencia o no de esta condición sobre los datos observados. Utilizaremos el

test de *Shapiro-Wilk* (Shapiro y Wilk, 1965), que analiza los datos observados para calcular el nivel de simetría y curtosis (o forma de la curva) para después calcular su diferencia con respecto a los de una distribución *Gaussiana*, obteniendo el valor de p a partir de la suma de cuadrados de esas discrepancias:

$$W = \frac{\left(\sum_{i=1}^{n} a_i x_{(i)}\right)^2}{\sum_{i=1}^{n}(x_i - \overline{x})^2} \tag{C.1}$$

donde $x_{(i)}$ es la i-ésima menor observación en la muestra, $\overline{x}$ es la media y las constantes a_i se obtienen mediante:

$$(a_1, \ldots, a_n) = \frac{m^T V^{-1}}{(m^T V^{-1} V^{-1} m)^{1/2}} \tag{C.2}$$

$m = (m_1, \ldots, m_n)^T$ son los valores esperados, de menor a mayor, de observaciones de una muestra de una distribución normal idéntica, y V es la matriz de covarianza de éstos.

- *Homocedasticidad*: Esta propiedad indica que las varianzas de las muestras deben ser iguales. El test de *Levene* se utiliza para comprobar si k muestras presentan o no esta homogeneidad en las varianzas. Este test tiene menos dependencia con la normalidad de las muestras que otros test como el de *Bartlett*. Se calcula como sigue:

$$\begin{aligned} W &= \frac{(N-k)\sum_{i=1}^{k} N_i (Z_{i.} - Z_{..})^2}{(k-1)\sum_{i=1}^{k}\sum_{j=1}^{N_i}(Z_{ij} - Z_{i.})^2} \\ Z_{ij} &= |Y_{ij} - \overline{Y_{i.}}| \text{ con } \overline{Y_{i.}} \text{ la media del grupo } i, \\ Z_{..} &= \frac{1}{N}\sum_{i=1}^{k}\sum_{j=1}^{N_i} Z_{ij} \text{ es la media de todos los } Z_{ij}, \\ Z_{i.} &= \frac{1}{N_i}\sum_{j=1}^{N_i} Z_{ij} \text{ es la media de } Z_{ij} \text{ para el grupo } i, \end{aligned} \tag{C.3}$$

con N, el número total de observaciones, N_i el número de observaciones de la muestra i, k es el número de muestras e Y_{ij} es la observación j-ésima de la muestra i.

C.1.2. Test de *Student*

El test de *Student* (Gosset, 1908) se utiliza para determinar si dos variables aleatorias normales y con la misma varianza tienen medias iguales o diferentes. Se aplica ampliamente para determinar si los resultados de dos algoritmos de optimización sobre un problema son, en media, iguales. Para ello, se calcula el estadístico t:

$$\begin{aligned} t &= \frac{\overline{x_1} - \overline{x_2}}{\sigma_{\overline{x_1} - \overline{x_2}}} \\ \sigma_{\overline{x_1} - \overline{x_2}} &= \sqrt{\frac{(n_1-1)\sigma_1^2 + (n_2-1)\sigma_2^2}{n_1 + n_2 - 2}\left(\frac{1}{n_1} + \frac{1}{n_2}\right)} \end{aligned} \tag{C.4}$$

donde n_1 y n_2 son el número de resultados disponibles de cada algoritmo, $\overline{x_1}$ y $\overline{x_2}$ son los resultados medios de cada uno, y σ_1 y σ_2, las varianzas de los resultados de cada algoritmo.

Bajo la suposición de la hipótesis nula (las dos variables aleatorias tienen la misma media), el estadístico t sigue una distribución t *de Student* con n_1+n_2-2 grados de libertad.

C.2. Tests No Paramétricos

Para diferenciar a un test no paramétrico del paramétrico hay que comprobar el tipo de datos que el test utiliza. Un test no paramétrico es aquél que utiliza datos de tipo nominal o que representan un orden de forma de ranking. Como norma general, un test no paramétrico es menos restrictivo que uno paramétrico, y menos robusto que uno paramétrico cuya aplicación se realiza sobre datos que cumple las condiciones de independencia, normalidad y homocedasticidad.

En las siguientes secciones, explicamos la funcionalidad básica de cada test no paramétrico utilizado en esta memoria junto al objetivo que se persigue con su utilización.

C.2.1. Test de *Friedman*

El test de *Friedman* (Friedman, 1937, 1939, 1940) es un equivalente no paramétrico al test de medidas repetidas ANOVA. Calcula el ranking de los resultados observados por el algoritmo j (r_j para el algoritmo j con k algoritmos) para cada función, asignando al mejor de ellos el ranking 1, y al peor el ranking k. Bajo la hipótesis nula, que supone que los resultados de los algoritmos son equivalentes y, por tanto, sus rankings son similares. El estadístico de *Friedman* es:

$$\chi_F^2 = \frac{12N}{k(k+1)} \left[\sum_j^k R_j^2 - \frac{k(k+1)^2}{4} \right], \tag{C.5}$$

siendo $R_j = 1/N \sum_i^N r_j^i$, y N, el número de funciones. Los valores críticos del estadístico de *Friedman* coinciden exactamente con los establecidos en la distribución χ^2 cuando $N > 10$ y $k > 5$. En caso contrario, los valores exactos pueden consultarse en Sheskin (2000) y Zar (1999).

Cuando el valor del test de *Friedman* es mayor que su valor crítico, se rechaza la hipótesis nula (todos los algoritmos son equivalentes), obteniendo, por tanto, que existe al menos una diferencia significativa entre los resultados de los algoritmos. El test de *Friedman* sólo indica que existe alguna diferencia significativa, no qué algoritmos ofrecen diferencias estadísticamente relevantes. Para eso, existe una serie de tests *post-hoc* que se pueden aplicar para identificar entre qué algoritmos existe una diferencia relevante.

C.2.2. Test de *Iman-Davenport*

El test de *Iman-Davenport* (Iman y Davenport, 1980) es una medida derivada de la de *Friedman* que no presenta el efecto conservador indeseado de éste. Por ello, en nuestros estudios hemos utilizado este test en vez de el de *Friedman*. El estadístico es:

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2}, \tag{C.6}$$

y se distribuye acorde a una distribución F con $k-1$ y $(k-1)(N-1)$ grados de libertad.

C.2.3. Test de *Holm*

Si se rechaza la hipótesis nula en el test de *Friedman* o de *Iman-Davenport*, podemos proceder a realizar un análisis *post-hoc*. El test de *Holm* (Holm, 1979) se utiliza cuando queremos comparar un algoritmo de control (aquél que obtiene el mejor ranking definido en el test de *Friedman*) frente a los demás. Éste prueba secuencialmente las hipótesis ordenadas según su significación. Denominaremos a los valores de p, ordenados, por $p_1, p_2, \ldots$, de tal forma que $p_1 \leq p_2 \leq \ldots \leq p_{k-1}$. El método de *Holm* compara cada p_i con $\alpha/(k-i)$ comenzando desde el valor de p más significante. Si p_1 es menor que $\alpha/(k-1)$, la correspondiente hipótesis se rechaza y nos permite comparar p_2 con $\alpha/(k-2)$. Si la segunda hipótesis se rechaza, continuamos el proceso. En cuanto una determinada hipótesis no se puede rechazar, todas las restantes se aceptan. El estadístico para comparar el algoritmo i-ésimo con el j-ésimo es:

$$z = \frac{(R_i - R_j)}{\sqrt{\frac{k(k+1)}{6N}}} \tag{C.7}$$

El valor z se utiliza para encontrar la probabilidad correspondiente a partir de la tabla de la distribución normal, que se compara con el correspondiente valor de α.

C.2.4. Test de Ranking de Signos de *Wilcoxon*

Este test (Wilcoxon, 1945) permite identificar si existe una diferencia significativa entre dos algoritmos para un conjunto de funciones, y determinar, en ese caso, cuál de ellos es mejor.

El procedimiento que se aplica es el siguiente:

1. Calcula para cada función, la diferencia entre los valores obtenidos por el algoritmo a comparar y el de referencia.

2. Se dividen las funciones en dos grupos: Funciones para las que el algoritmo de referencia es el mejor y funciones para los que el algoritmo de referencia

ofrece el peor resultado. Aquellas funciones en donde ambos algoritmos obtengan el mismo resultado se dividirán por igual entre ambos grupos (la mitad como positivas y las otras como negativas).

3. Se utilizan los valores absolutos de todas las diferencias para crear un orden de funciones: se asigna el menor valor a la diferencia con menor valor absoluto. Dado que las funciones utilizadas tienen diferentes rangos, hemos normalizado, previamente, los resultados de cada algoritmo al intervalo $[0, 1]$ según el mejor y peor resultado de todos los algoritmos en cada función.

4. Se suman los rankings de las funciones para cada uno de los grupos anteriores (*Ranking Positivo*, R^+, para el primer grupo, y *Ranking Negativo*, R^-, para el segundo).

5. Se calcula $T = min(R^+, R^-)$.

6. Se calcula el valor de la distribución T de *Wilcoxon* para N grados de libertad (Tabla B.123 en Zar (1999)) como valor crítico.

7. Se compara el valor T con el valor crítico. Cuando T es menor al valor crítico, se rechaza la hipótesis nula (es decir, la diferencia es estadísticamente significativa).

8. Si se encuentra una diferencia relevante entre el algoritmo a comparar y el de referencia, se identifica aquél que presenta mejor comportamiento según las siguientes reglas:

   - Si $T = R^+$ $(R^+ < R^-)$, es mejor el algoritmo comparado.
   - Si $T = R^-$ $(R^- > R^+)$, es mejor el algoritmo de referencia.

C.2.5. Test de *Mann-Whitney*

El test de *Mann-Whitney* (Mann y Whitney, 1947) es el equivalente no paramétrico al test de *Student*, comprueba si dos muestras provienen de la misma distribución, que es la hipótesis nula. Es útil cuando queremos comparar los resultados de dos algoritmos sobre una misma función. Para ello, calculamos el estadístico, normalmente llamado U, de la siguiente forma:

1. Ordenar los resultados de ambos algoritmos.

2. Sumar los rankings asociados a los resultados del primer algoritmo (R_1).

3. Calcular el estadístico U como:

$$U = R_1 - \frac{n_1(n_1 + 1)}{2} \tag{C.8}$$

donde n_1 es el número de resultados del primer algoritmo.

Cuando se cuenta con menos de 20 resultados por algoritmo, los valores críticos de U se pueden consultar en Hollander y Wolfe (1999). Con muestras mayores, se realiza la siguiente aproximación normal:

$$\begin{aligned} z &= (U - \mu_U)/\sigma_U \\ \mu_U &= n_1 n_2/2 \\ \sigma_U &= \sqrt{\frac{n_1 n_2 (n_1 + n_2 + 1)}{12}} \end{aligned} \tag{C.9}$$

donde n_1 y n_2 son el número de muestras de cada algoritmo. Bajo la suposición de la hipótesis nula, la aproximación z sigue una distribución normal.

C.3. Proceso del Análisis Estadístico

Dado un conjunto de algoritmos que se han evaluado sobre un conjunto de problemas de prueba un número elevado de ejecuciones, procederemos a realizar un análisis estadístico para conocer cuál de ellos obtiene los mejores resultados y si éstos son significativamente diferentes a los del resto de algoritmos. Para ello, seguiremos los siguientes pasos:

1. Aplicar el test de *Iman-Davenport* para confirmar que existen diferencias significativas entre los resultados de los algoritmos comparados. En caso afirmativo, avanzaremos al siguiente paso para aplicar los tests *post-hoc*. En caso contrario, el proceso se detendrá y conluiremos que no hay diferencias significativas entre los resultados de los algoritmos, en particular, ningún algoritmo del conjunto destaca por conseguir los mejores resultados.

2. Aplicar el test de *Holm* para comparar varios algoritmos con el que obtiene el mejor ranking. Este test nos permite identificar los algoritmos que, obteniendo rankings elevados, son estadísticamente peores que el de mejor ranking. De esta forma obtenemos dos grupos, uno con el algoritmo que obtiene el mejor ranking y los que pudieran alcanzar resultados equivalentes, y otro con los algoritmos que no alcanzan los buenos resultados que ofrece el de mejor ranking.

3. Aplicar el test de *Wilcoxon* para comparar los algoritmos para los que *Holm* no detecta diferencias con el algoritmo de mejor ranking. Este test nos permite identificar diferencias significativas entre los resultados de dos algoritmos aun cuando sus rankings son comparables. De esta forma, damos un paso más en el filtrado del conjunto de algoritmos que parecen equivalentes al de mejor ranking y comprobar si este último es realmente superior al resto de algoritmos.

Por otro lado, en los casos en los que nos interese analizar el comportamiento de algún algoritmo en cada uno de los problema de prueba, con respecto al resto de algoritmos comparados, aplicaremos los siguientes pasos:

1. Aplicar los tests de normalidad y homocedasticidad (Sección C.1.1), sobre los resultados de los algoritmos en cada uno de los problemas de prueba, con la intención de comprobar si podemos aplicar posteriormente un test paramétrico, o debemos aplicar uno no paramétrico.

2. Si se dan las condiciones de normalidad y homocedasticidad, entonces aplicaremos el test de *Student* (Sección C.1.2). En otro caso, aplicaremos el de *Mann-Whitney* (Sección C.2.5). Con estos resultados, presentaremos unas gráficas con el porcentaje de funciones o algoritmos para los que una propuesta obtiene resultados mejores, peores o sin diferencia significativa.

Bibliography

(2007) IEEE congress on Evolutionary Computation

Aarts EHL, van Laarhoven PJM (1992) Local search in coding theory. Discret. Math. 106/107:11–18

Aarts EHL, Korst JHM, Laarhoven PJM (1997) Simulated annealing. In: Aarts EHL, Lenstra JK (eds) Local Search in Combinatorial Optimization, Wiley-Interscience, Chichester, England, pp 91–120

Affenzeller M, Wagner S (2004) Sasegasa: A new generic parallel evolutionary algorithm for achieving highest quality results. J. Heuristics 10(3):243–267

Ahuja RK, Orlin JB, Tiwari A (2000) A greedy genetic algorithm for the quadratic assignment problem. Comput. Oper. Res. 27(10):917–934

Alba E, Tomassini M (2002) Parallelism and evolutionary algorithms. IEEE Trans. Evol. Comput. 6(5):443–462

Alba E, Troya JM (1999) A survey of parallel distributed genetic algorithms. Complexity 4(4):31–52

Andre J, Siarry P, Dognon T (2001) An improvement of the standard genetic algorithm fighting premature convergence in continuous optimization. Adv. Eng. Softw. 32(1):49–60

Angel E (2006) A survey of approximation results for local search algorithms. In: Bampis E, Jansen K, Kenyon C (eds) Efficient Approximation and Online Algorithms LNCS 3484, Springer Berlin, Heidelberg, pp 30–73

Antonisse J (1989) A new interpretation of schema notation that overturns the binary encoding constraint. In: Schaffer JD (ed) Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Mateo, pp 86–91

Arabas J, Michalewicz Z, Mulawka J (1994) Gavaps- a genetic algorithm with varying population size. In: IEEE Conf. on Evolutionary Computation, pp 73–78

Auger A, Hansen N (2005) Performance evaluation of an advanced local search evolutionary algorithm. In: Proc. of the IEEE Int. Conf. Evolutionary Computation, vol 2, pp 1777–1784

Ausiello G, Protasi M (1995) Local search, reducibility and approximability of NP-optimization problems. Information Processing Letters 54(2):73–79

Bäck T (1992a) The interaction of mutation rate, selection, and self-adaptation within genetic algorithm. In: Männer R, Manderick B (eds) Parallel Problem Solving from Nature, Elsevier Science Publishers, Amsterdam, vol 2, pp 85–94

Bäck T (1992b) Self-adaptation in genetic algorithms. In: Varela FJ, Bourgine P (eds) European Conf. on Artificial Life, MIT Press, Cambridge, pp 263–271

Bäck T (1994) Selective pressure in evolutionary algorithms: A characterization of selection mechanisms. In: IEEE Conf. on Evolutionary Computation, pp 57–62

Bäck T (1996) Evolutionary Algorithms in Theory and Practice. Oxford University Press

Bäck T, Fogel DB, Michalewicz Z (1997) Handbook of Evolutionary Computation. Institute of Physics Publishers

Baker JE (1987a) Adaptive selection methods for genetic algorithms. In: Grefenstette JJ (ed) Int. Conf. on Genetic Algorithms Applications and Their Application, Erlbaum Associates, Hillsdale, pp 14–21

Baker JE (1987b) Reducing bias and inefficiency in the selection algorithm. In: Grefenstette JJ (ed) Int. Conf. on Genetic Algorithms Applications and Their Application, Erlbaum Associates, Hillsdale, pp 14–21

Ballester PJ, Carter JN (2003) Real-parameter genetic algorithms for finding multiple optimal solutions in multi-modal optimization. In: Cantú-Paz E (ed) Proc. of the Genetic and Evolutionary Computation Conf., Springer Berlin, Heidelberg, LNCS, vol 2723, pp 706–717

Ballester PJ, Carter JN (2004) An effective real-parameter genetic algorithm with parent centric normal crossover for multimodal optimisation. In: Deb K (ed) Proc. of the Genetic and Evolutionary Computation Conf., Springer Berlin, Heidelberg, LNCS, vol 3102, pp 901–913

Ballester PJ, Richards WG (2006) A multiparent version of the parent-centric normal crossover for multimodal optimization. In: Proc. of the Congress on Evolutionary Computation, pp 2999–3006

Bandyopadhyay S, Pal SK, Maulik U (1998) Incorporating chromosoma differentiation in genetic algorithms. Inf. Sci. (104):901–913

Beasley JE (1990) OR-Library: distributing test problems by electronic mail. J. Oper. Res. Soc. 41(11):1069–1072, URL http://people.brunel.ac.uk/~mastjjb/jeb/info.html

Beasley JE (1998) Heuristic algorithms for the unconstrained binary quadratic programming problem. Tech. rep., The Management School, Imperial College

Beasley JE, Bull DR, Martin RR (1993a) A Sequential Niche Technique for Multimodal Function Optimization. Evol. Comput. 1(2):101–125

Beasley JE, Bull DR, Martin RR (1993b) An Overview of Genetic Algorithms: Part 2, Research Topics. Univ. Comput. 15(4):170–181

Beyer HG, Deb K (2001) On self-adaptive features in real-parameter evolutionary algorithms. IEEE Trans. Evol. Comput. 5(3):250–270

Beyer HG, Schwefel HP (2002) Evolution strategies–A comprehensive introduction. Natural Computing 1(1):3–52

Blum C (2002) Aco applied to group shop scheduling: A case study on intensification and diversification. In: Dorigo M, et al (eds) ANTS, Springer-Verlag, Lecture Notes in Computer Science, 2463, pp 14–27

Blum C, Roli A (2003) Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Comput. Surv. 35(3):268–308

Boender CGE, Rinnooy-Kan AHG, Stougie L, Timmer GT (1982) A stochastic method for global optimization. Math. Program. 22:125–140

Bonham CR, Parmee IC (1999) An investigation of exploration and exploitation within cluster oriented genetic algorithms (COGAs). In: Proc. of the Conf. on Genetic and Evolutionary Computation, Morgan Kaufmann, San Francisco, California, pp 1491–1497

Booker L (1987) Improving Search in Genetic Algorithms, Morgan Kaufmann, Los Altos, pp 61–73. Genetic Algorithms and Simulated Annealing

Bramlette MF (1991) Initialization, mutation and selection methods in genetic algorithms for function optimization. In: Belew R, Booker LB (eds) Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Mateo, pp 100–107

Bramlette MF, Bouchard EE (1991) Genetic Algorithms in Parametric Design of Aircraft, Van Nostrand Reinhold, New York, pp 109–123. Handbook of Genetic Algorithms

Branke J, Cutaia M, Dold H (1999) Reducing genetic drift in steady state evolutionary algorithms. In: Banzhaf W, et al (eds) Genetic and Evolutionary Computation Conf., Morgan Kaufmann, San Francisco, pp 68–74

Brito J (2007) Genetic learning of vocal tract area functions for articulatory synthesis of spanish vowels. App. Soft Comput. 7(3):1035–1043

Caruana RA, Schaffer JD (1988) Representation and Hidden Bias: Gray versus Binary Coding for Genetic Algorithms. In: Int. Conf. on Machine Learning, pp 153–162

Cedeño W, Vemuri V, Slezak T (1995) Multi-niche crowding in genetic algorithms and its application to the assembly of dna restriction-fragments. Evol. Comput. 2(4):321–345

Chankong V, Haimes YY (1983) Multiobjective Decision Making Theory and Methodology. North-Holland

Charon I, Hudry O (2001) The noising methods: A generalization of some metaheuristics. Eur. J. Oper. Res. 135(1):86–101

Chelouah R, Siarry P (2000) A continuous genetic algorithm designed for the global optimization of multimodal functions. J. Heuristics 6(2):191–213

Chelouah R, Siarry P (2003) Genetic and Nelder-Mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions. Eur. J. Oper. Res. 148(2):335–348

Chen Z, Kang L (2005) Steady-state evolutionary algorithm for multimodal function global optimization. In: Hao Y, Liu J, Wang Y, m Cheung Y, Yin H, Jiao L, Ma J, Jiao Y-C (eds) Proc. of the Int. Conf. on Computation Ingelligence and Security, LNCS, vol 3801, pp 200–207

Chiang T-C, Liu C-H, Huang Y-M (2007) A near-optimal multicast scheme for mobile ad hoc networks using a hybrid genetic algorithm. Exp. Syst. App. 33(3):734–742

Coello CA, Van Veldhuizen DA, Lamont GB (2007) Evolutionary algorithms for solving multi-objective problems. Springer, New York

Cohoon JP, Hedge S, Martin W, Richards D (1987) Punctuated Equilibria: A Parallel Genetic Algorithm. In: Grefenstette JJ (ed) Int. Conf. on Genetic Algorithms Applications and Their Application, Erlbaum Associates, Hillsdale, pp 148–154

Collins RJ (1992) Studies in artificial evolution. PhD thesis, Universidad de California, Los Angeles

Corcoran AL, Sen S (1994) Using Real-Valued Genetic Algorithms to Evolve Sets for Classification. In: IEEE Conf. on Evolutionary Computation, pp 120–124

Craighurst R, Martin W (1995) Enhancing GA performance through crossover prohibitions based on ancestry. In: Eshelman LJ (ed) Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Mateo, pp 130–137

Crauwels HAJ, Potts CN, Van Wassenhove LN (1998) Local search heuristics for the single machine total weighted tardiness scheduling problem. INFORMS J. on Computing 10(3):341–350

Darwin C (1859) On the Origin of Species by Means of Natural Selection. John Murray

Davidor Y (1991) A Naturally Occurring Niche & Species Phenomenon: The Model and First Results. In: Belew R, Booker LB (eds) Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Mateo, pp 257–263

Davis L (1985) Job Shop Scheduling with Genetic Algorithms. In: Grefenstette JJ (ed) Int. Conf. on Genetic Algorithms Applications and Their Application, Erlbaum Associates, Hillsdale, pp 136–140

Davis L (1989) Adapting operator probabilities in genetic algorithms. In: Schaffer JD (ed) Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Mateo, pp 61–69

Davis L (1991a) Bit-climbing, representational bias, and test suite design. In: Proc. of the Int. Conf. on Genetic Algorithms, pp 18–23

Davis L (1991b) Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York

Deb K (2000) An efficient constraint handling method for genetic algorithms. Comput. Meth. Appl. Mech. Eng. 186(2):311–338

Deb K (2005) A population-based algorithm-generator for real-parameter optimization. Soft Computing 9:236–253

Deb K, Agrawal RB (1995) Simulated binary crossover for continuous search space. Complex Syst. 9(2):115–148

Deb K, Beyer H (2001) Self-adaptive genetic algorithms with simulated binary crossover. Evol. Comput. 9(2):195–219

Deb K, Tiwari S (2008) Omni-optimizer: A generic evolutionary algorithm for single and multi-objective optimization. Eur. J. Oper. Res. 185(3):1062–1087

Deb K, Anand A, Joshi D (2002a) A computationally efficient evolutionary algorithm for real-parameter optimization. Evol. Comput. 10(4):371–395

Deb K, Pratap A, Agarwal S, Meyarivan T (2002b) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6(2):182–197

De Jong ED, Watson RA, Pollack JB (2001) Reducing bloat and promoting diversity using multi-objective methods. In: Spector L, et al (eds) Proc. of the Conf. on Genetic and Evolutionary Computation, Morgan Kaufmann, San Francisco, California, pp 11–18

De Jong K (1975) An Analysis of the Behavior of a Class of Genetic Adaptative Systems. PhD thesis, Univ. of Michigan, Ann Arbor

De Jong KA, Sarma J (1993) Generation gaps revisited, Foundations of Genetic Algorithms, vol 2, Morgan Kaufmann, San Mateo, pp 19–28

De Jong KA, Spears WM (1992) A formal analysis of the role of multi-point crossover in genetic algorithms. Ann. Math. Artif. Intell. 5(1):1–26

De Jong K, Potter MA, Spears WM (1997) Using problem generators to explore the effects of epistasis. In: Bäck T (ed) Proc. of the Int. Conf. on Genetic Algorithms, Morgan Kaufmann, pp 338–345

de la Maza M, Tidor B (1992) Increased flexibility in genetic algorithms: The use of variable boltzmann selective preassure to control propagation. In: ORCA CSTS Conf.: Computer Science and Operations Research: New Developments in Their Interfaces, pp 425–440

Dietzfelbinger M, Naudts B, Van Hoyweghen C, Wegener I (2003) The analysis of a recombinative hill-climber on H-IFF. IEEE Trans. Evol. Comput. 7(5):417–423

Dorigo M, Stützle T (2004) Ant Colony Optimization. MIT Press

Dorigo M, Maniezzo V, Colorni A (1996) The Ant System: Optimization by a colony of cooperating agents. IEEE Trans. Syst., Man, Cybern. B 26(1):29–41, URL citeseer.ist.psu.edu/dorigo96ant.html

Dunham B, Fridshal D, Fridshal R, North JH (1963) Design by natural selection. Synthese 15(1):254–259

Eberhart RC, Kennedy J (1995) A new optimizer using particle swarm theory. In: Int. Symposium on Micromachine and Human Sicence, pp 39–43

Eiben AE, Smith JE (2003) Introduction to Evolutionary Computing. Springer-Verlag

Eiben AE, Hinterding R, Michalewicz Z (1999) Parameter control in evolutionary algorithms. IEEE Trans. Evol. Comput. 3(2):124–141

Eiben G, Schut MC (2008) New ways to calibrae evolutionary algorithms. In: Siarry P, Michalewicz Z (eds) Advances in Metaheuristics for Hard Optimization, Springer-Verlag, pp 153–177

Eldredge N, Gould SJ (1972) Punctuated Equilibria: An Alternative to Phyletic Gradualism, Freeman, Cooper, San Francisco, pp 82–115. Models of Paleobiology

Elliott L, Ingham DB, Kyne AG, Mera NS, Pourkashanian M, Wilson CW (2004) An informed operator based genetic algorithm for tuning the reaction rate parameters of chemical kinetics mechanisms. In: Deb K (ed) Proc. of the Genetic and Evolutionary Computation Conf., Springer Berlin, Heidelberg, LNCS, vol 3103, pp 945–956

Eshelman LJ (1991) The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination, Foundations of Genetic Algorithms, vol 1, Morgan Kaufmann, San Mateo, pp 265–283

Eshelman LJ, Schaffer JD (1991) Preventing premature convergence in genetic algorithms by preventing incest. In: Belew R, Booker LB (eds) Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Mateo, pp 115–122

Eshelman LJ, Schaffer JD (1993) Real-coded genetic algorithms and interval-schemata. In: Whitley LD (ed) Foundations of Genetic Algorithms 2, Morgan Kaufmann, pp 187–202

Eshelman LJ, Caruana A, Schaffer JD (1989) Biases in the crossover landscape. In: Schaffer JD (ed) Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Mateo, pp 86–91

Eshelman LJ, Mathias KE, Schaffer JD (1997) Convergence controlled variation. In: Belew R, Vose M (eds) Foundations of Genetic Algorithms 4, Morgan Kaufmann, San Mateo, California, pp 203–224

Feng J, Li Wenjuan, Shi X, Chen J (2006) A hybrid genetic algorithm with fitness sharing based on rough sets theory. In: Proc. of the World Congress on Intelligent Control and Automation, vol 1, pp 3340–3344

Feo T, Resende M (1995) Greedy randomized adaptive search. J. Global Optim. 6:109–133

Fernandes C, Rosa A (2001) A study on non-random mating and varying population size in genetic algorithms using a royal road function. In: Proc. of the Congress on Evolutionary Computation, IEEE Press, Piscataway, New Jersey, pp 60–66

Festa P, Pardalos PM, Resende MGC, Ribeiro CC (2002) Randomized heuristics for the max-cut problem. Optim. Method Softw. 17(6):1033–1058

Fischer I, Gruber G, Rendl F, Sotirov R (2006) Computational experience with a bundle approach for semidefinite cutting plane relaxations of max-cut and equipartition. Math. Program. 105(2–3):451–469

Fisher R (1930) The General Theory of Natural Selection. Clarendon Press, Oxford

Fogarty TC (1989) Varying the probability of mutation in genetic algorithm. In: Schaffer JD (ed) Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Mateo, pp 104–109

Fogel DB (1992) Evolving Artificial Intelligence. PhD thesis, Univ. of California, San Diego, CA

Fogel DB (1994) Evolutionary programming: An introduction and some current directions. Stat. Comput. 4:113–129

Fogel DB (1995) Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. IEEE Press, Piscataway, New Jersey

Fogel LJ, Owens AJ, Walsh MJ (1966) Artificial Intelligence Through Simulated Evolution. John Wiley, New York

Fowler JW, Wirojanagud P, Gel ES (2008) Heuristics for workforce planning with worker differences. Eur. J. Oper. Res. 190(3):724–740

Fox BR, McMahon MB (1991) Genetic Operators for Sequencing Problems, Foundations of Genetic Algorithms, vol 1, Morgan Kaufmann, San Mateo, pp 284–300

Friedman M (1937) The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J. Am. Stat. Assoc. 32(200):675–701

Friedman M (1939) A correction: The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J. Am. Stat. Assoc. 34(205):109

Friedman M (1940) A comparison of alternative tests of significance for the problem of m rankings. Ann. Math. Stat. 11(1):86–92

Gang Peng, Iimura Ichiro, Tsurusawa Hidenobu, Nakayama Shigeru (2004) A local search algorithm based on genetic recombination. In: Proc. of the Conf. on Genetic and Evolutionary Computation, pp 1–11

Gang Peng, Iimura Ichiro, Nakayama Shigeru (2007) Application of genetic recombination to genetic local search in TSP. Int. J. Inf. Technol. 13(1):57–66

Garcia S, Molina D, Lozano M, Herrera F (2008) A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC'2005 special session on real parameter optimization. J. Heuristics p En prensa

Ghosh A, Tsutsui S, Tanaka H (1998) Function optimization in nonstationary environment using steady state genetic algorithms wit aging of individuals. In: IEEE Conf. on Evolutionary Computation, IEEE Press, New York, pp 666–671

Gilbert JC, Nocedal J (1992) Global convergence properties of conjugate gradient methods. SIAM J. Optim. 2:21–42

Glover F (1986) Future paths for integer programming and links to artificial intelligence. Comput. Oper. Res. 13:533–549

Glover F (1999) Scatter search and path relinking. In: Corne D, Dorigo M, Glover F, Dasgupta D, Moscato P, Poli R, Price KV (eds) New Ideas in Optimization, McGraw-Hill Ltd., UK, pp 297–316

Glover F, Kochenberger G (eds) (2003) Handbook of metaheuristics. Kluwer Academic Publishers, Massachusetts

Glover F, Laguna M (1997) Tabu Search. Kluwer Academic Publishers

Goh KS, Lim A, Rodrigues B (2003) Sexual selection for genetic algorithms. Artif. Intell. Rev. 19:123–152

Goldberg DE (1989a) Genetic algorithms and walsh functions: Part ii, deception and its analysis. Complex Syst. 3:153–171

Goldberg DE (1989b) Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA

Goldberg DE (1989c) Sizing populations for serial and parallel genetic algorithms. In: Schaffer JD (ed) Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Mateo, pp 70–79

Goldberg DE (1991) Real-Coded Genetic Algorithms, Virtual Alphabets, and Blocking. Complex Syst. 5:139–167

Goldberg DE, Deb K (1991) A Comparative study of Selection Schemes Used in Genetic Algorithms, Foundations of Genetic Algorithms, vol 1, Morgan Kaufmann, San Mateo, pp 69–93

Goldberg DE, Richardson J (1987) Genetic algorithms with sharing for multimodal function optimization. In: Grefenstette JJ (ed) Proc. of the Int. Conf. Genetic Algorithms, L. Erlbraum Associates, Hillsdale, pp 41–49

Goldberg DE, Korb B, Deb K (1989) Messy genetic algorithms: motivation, analysis, and first results. Complex Syst. 3:493–530

Gosset W (1908) The probable error of a mean. Biometrika 6(1):1–25

Grefenstette JJ (1986) Optimization of control parameters for genetic algorithms. IEEE Trans. Syst., Man, Cybern. 16:122–128

Guanqi G, Shouyi Y (2003) Evolutionary parallel local search for function optimization. IEEE Trans. Syst., Man, Cybern. B 33(6):864–876

Gulati VP, Gupta SK, Mittal AK (1984) Unconstrained quadratic bivalent programming problem. Eur. J. Oper. Res. 15:121–125

Hansen N, Ostermeier A (1996) Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In: IEEE Conf. on Evolutionary Computation, pp 312–317

Hansen N, Ostermeier A (2001) Completely derandomized self-adaptation in evolution strategies. Evol. Comput. 9(2):159–195

Hansen N, Müller SD, Koumoutsakos P (2003) Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). Evol. Comput. 11(1):1–18

Hansen P, Mladenović N (2001) Variable neighborhood search: Principles and applications. Eur. J. Oper. Res. 130(3):449–467

Hansen P, Mladenović N (2002) Variable neighborhood search. In: Glover F, Kochenberger G (eds) Handbook of Metaheuristics, Kluwer Academic Publishers, Boston, MA, USA, pp 145–184

Harada K, Ikeda K, Kobayashi S (2006) Hybridization of genetic algorithm and local search in multiobjective function optimization: recommendation of ga then ls. In: Proc. of the Genetic and Evolutionary Computation Conf., ACM Press, New York, NY, USA, pp 667–674, DOI http://doi.acm.org/10.1145/1143997.1144116

Harik G (1995) Finding multimodal solutions using restricted tournament selection. In: Eshelman LJ (ed) Proc. of the Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Mateo, California, pp 24–31

Hart E, Smith JE, Krasnogor N (2004) Recent Advances in Memetic Algorithms. Springer

Helmberg C, Rendl F (2000) A spectral bundle method for semidefinite programming. SIAM J. Optim. 10(3):673–696

Herrera F, Lozano M (2000a) Gradual distributed real-coded genetic algorithms. IEEE Trans. Evol. Comput. 4(1):43–63

Herrera F, Lozano M (2000b) Two-loop real-coded genetic algorithms with adaptive control of mutation step sizes. App. Intell. 13(3):187–204

Herrera F, Lozano M, Verdegay JL (1995) Runing fuzzy logic controllers by genetic algorithms. Int. J. Approx. Reasoning 12:299–315

Herrera F, Lozano M, Verdegay JL (1998) Tackling real-coded genetic algorithms: operators and tools for behavioral analysis. Artif. Intell. Rev. 12(4):265–319

Herrera F, Lozano M, Moraga C (1999) Hierarchical distributed genetic algorithms. Int. J. Intell. Syst. 14(11):1099–1121

Herrera F, Lozano M, Sánchez AM (2003) A taxonomy for the crossover operator for real-coded genetic algorithms. an experimental study. Int. J. Intell. Syst. 18(3):309–338

Herrera F, Lozano M, Sánchez AM (2005) Hybrid crossover operators for real-coded genetic algorithms: An experimental study. Soft Comput. 9(4):280–298

Hervás-Martínez C, Ortiz-Boyer D (2005) Analyzing the statistical features of CIXL2 crossover offspring. Soft Comput. 9(4):270–279

Hoffmeister F, Bäck T (1990) Genetic Algorithms and Evolution Strategies - Similarities and Differences. In: Parallel Problem Solving from Nature, Springer-Verlag, London, vol 1, pp 455–469

Holland JH (1975) Adaptation in Natural and Artificial Systems. The University of Michigan Press

Hollander M, Wolfe DA (1999) Nonparametric Statistical Methods, 2nd edn. Wiley-Interscience

Holm S (1979) A simple sequentially rejective multiple test procedure. Scand. J. Stat. 6:65–70

Horner JL, White RG, Hooke R, Jeeves TA (1961) Direct search solution of numerical and statistical problems. J. Sound Vib 147(1):87–103

Hrstka O, Kučerová A (2004) Improvements of real coded genetic algorithms based on differential operators preventing premature convergence. Adv. Eng. Softw. 35(3–4):237–246

Hutter M (2002) Fitness uniform selection to preserve genetic diversity. In: IEEE Cong. on Evolutionary Computation, IEEE Press, pp 783–788

Hutter M, Legg S (2006) Fitness uniform optimization. IEEE Trans. Evol. Comput. 10(5):568–589

Ichikawa Y, Ishii Y (1993) Retaining diversity of genetic algorithms for multivariable optimization and neural network learning. In: IEEE Int. Conf. on Neural Networks, pp 1110–1114

Iman RL, Davenport JM (1980) Approximations of the critical region of the Friedman statistic. In: Communications in Statistics, pp 571–595

Ishibuchi H, Yoshida T, Murata T (2003) Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. IEEE Trans. Evol. Comput. 7(2):204–223

Janikow CZ, Michalewicz Z (1991) An experimental comparison of binary and floating point representation in genetic algorithms. In: Belew R, Booker LB (eds) Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Mateo, pp 31–36

Jassadapakorn C, Chongstitvatana P (2004) Diversity control to improve convergence rate in genetic algorithms. In: Intelligent Data Engineering and Automated Learning, vol 2690, Springer, Berlín, pp 421–425

Johnson DS, Papadimtriou CH, Yannakakis M (1988) How easy is local search? Journal of Computer and System Sciences 37(1):79–100

Jones T (1995) Crossover, macromutation, and population-based search. In: Eshelman Larry (ed) Proceedings of the Sixth International Conference on Genetic Algorithms, Morgan Kaufmann, San Francisco, CA, pp 73–80

Julstrom BA (1995) What have you done for me lately? adapting operator probabilities in a steady-state genetic algorithm. In: Eshelman L (ed) Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Francisco, pp 81–87

Karlin S (1968) Equilibrium beahvior of population genetic models with non-random mating. J. Appl. Probab. 5:231–313

Karp RM (1972) Reducibility among combinatorial problems. In: Miller R, Thatcher J (eds) Complexity of Computer Computations, Plenum Press, New York, pp 85–103

Katayama K, Narihisa H (2001) A variant k-opt local search heuristic for binary quadratic programming. Trans. IEICE (A) J84-A(3):430–435

Kauffman SA (1989) Adaptation on rugged fitness landscapes 1:527–618

Kazarlis SA, Papadakis SE, Theocharis JB, Petridis V (2001) Microgenetic algorithms as generalized hill-climbing operators for GA optimization. IEEE Trans. Evol. Comput. 5(3):204–217

Kelly J, Davis L (1991) Hybridizing the GA and the K Nearest Neighbors Classification Algorithm. In: Belew R, Booker LB (eds) Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Mateo, pp 377–383

Kelner V, Capitanescu F, Léonard O, Wehenkel L (2008) A hybrid optimization technique coupling an evolutionary and a local search algorithm. J. Comput. Appl. Math. 215(2):448–456

Kennedy J, Eberhart RC (2001) Swarm Intelligence. Morgan Kaufmann

Kernighan BW, Lin S (1970) An efficient heuristic procedure for partitioning graphs. Bell System Technical Journal 49(2):291–307

Kirkpatrick S, Gelatt Jr CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680

Kita H (2001) A comparison study of self-adaptation in evolution strategies and real-coded genetic algorithms. Evol. Comput. 9(2):223–241

Kita H, Ono I, Kobayashi S (1999) Multi-parental extension of the unimodal normal distributioncrossover for real-coded genetic algorithms. In: Proc. of the Congress on Evolutionary Computation, vol 2, pp 1581–1588

Knowles JD, Corne DW (2000) M-PAES: a memetic algorithm for multiobjective optimization. In: IEEE Conf. on Evolutionary Computation, pp 325–332

Kong M, Tian P, Kao Y (2008) A new ant colony optimization algorithm for the multidimensional knapsack problem. Comput. Oper. Res. 35(8):2672–2683

Koumousis VK, Katsaras CP (2006) A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. IEEE Trans. Evol. Comput. 10(1):19–28

Koza JR (1991) Evolving a computer program to generate random numbers using the genetic programming paradigm. In: Belew R, Booker LB (eds) Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Mateo, pp 37–44

Koza JR (1992) Genetic Programming. MIT Press, Cambridge

Krasnogor N, Smith J (2005) A tutorial for competent memetic algorithms: Model, taxonomy and design issues. IEEE Trans. Evol. Comput. 9(5):474–488

Krishnakumar K (1989) Micro-genetic algorithms for stationary and non-stationary function optimization. In: SPIE: Intelligent Control and Adaptive Systems, vol 1196, pp 289–296

Kuo T, Hwang SY (1996) A genetic algorithm with disruptive selection. IEEE Trans. Syst., Man, Cybern. 26(2):299–307

Kwon Young-Doo, Kim Jae-Yong (2003) Convergence enhanced genetic algorithm with successive zooming method for solving continuous optimization problems. Comput. Struct. 81(17):1715–1725

Laarhoven PJM, Aarts EHL (1987) Simulated Annealing: Theory and Applications. Kluwer Academic Publishers Norwell, MA, USA

Laguna M (2003) Scatter Search. Kluwer Academic Publishers Boston, Mass

Larrañaga P, Lozano JA (2001) Estimation of Distribution Algorithms. Kluwer Academic Publishers

Larrañaga P, Kuijpers CMH, Murga RH, Inza I, Dizdarevic S (1999) Genetic algorithms for the travelling salesman problem: a review of representations and operators. Artificial Intelligence Review 13(2):129–170

Lazzerini B, Marcelloni F (2000) A genetic algorithm for generatin optimal assembly plans. Artif. Intell. Eng. 14(4):319–329

Lee C-Y (2003) Entropy-Boltzmann selection in the genetic algorithms. IEEE Trans. Syst., Man, Cybern. B 33(1):138–149

Lee C-Y, Yao X (2004) Evolutionary programming using mutations based on the Levy probability distribution. IEEE Trans. Evol. Comput. 8(1):1–13

Li TH, Lucasius CB, Kateman G (1992) “optimization of the calibration data with the dynamic genetic algorithm”. Anal. Chim. Acta 2768:123–134

Liang JJ, Qin AK, Suganthan PN, Baskar S (2004a) Evaluation of comprehensive learning particle swarm optimizer. In: Int. Conf. on Neural Information Processing, Lecture Notes in Computer Science, vol 3316, pp 230–235

Liang JJ, Qin AK, Suganthan PN, Baskar S (2004b) Particle swarm optimization algorithms with novel learning strategies. In: IEEE Int. Conf. on Systems, Man, and Cybernetics, pp 3659–3664

Liberti L, Dražić M (2005) Variable neighbourhood search for the global optimization of constrained NLPs. In: Proc. of GO, pp 1–5

Liepins GE, Vose MD (1990) Representational issues in genetic optimization. J. Exp. Theor. Artif. Intell. 2:101–115

Liepins GE, Vose MD (1992) Characterizing crossover in genetic algorithms. Ann. Math. Artif. Intell. 5(1):123–134

Lima CF, Pelikan M, Sastry K, Butz M, Goldberg DE, Lobo FG (2006) Substructural neighborhoods for local search in the bayesian optimization algorithm. In: Proc. of the Int. Conf. on Parallel Problem Solving from Nature, Lecture Notes in Computer Science 4193, pp 232–241

Lin F, Yang Q (2002) Improved genetic operator for genetic algorithm. J. Zhejinag Univ.: Sci. 3(4):431–434

Lin S, Kernighan BW (1973) An effective heuristic algorithm for the traveling-salesman problem. Oper. Res. 21(2):498–516

Lin S-C, Punch III WF, Goodman ED (1994) Coarse-grain genetic algorithms: Categorization and new approach. In: IEEE Parallel and Distributed Processing, pp 28–37

Lobo FG, Goldberg DE (2004) The parameter-less genetic algorithm in practice. Inf. Sci. 167:217–232

Lobo FG, Lima CF, Michalewicz Z (eds) (2007) Parameter setting in evolutionary algorithms, Studies in Computational Intelligence, vol 54. Springer

Lourenço HR, Martin O, Stützle T (2003) Iterated local search. In: Glover F, Kochenberger G (eds) Handbook of Metaheuristics, Kluwer Academic Publishers, Boston, MA, USA, pp 321–353

Lozano M, Herrera F, Krasnogor N, Molina D (2004) Real-coded memetic algorithms with crossover hill-climbing. Evol. Comput. 12(3):273–302

Lucasius CB, Kateman G (1989) Applications of genetic algorithms in chemometrics. In: Schaffer JD (ed) Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Mateo, pp 170–176

Mahfoud SW (1992) Crowding and preselection revised. In: Männer R, Manderick B (eds) Parallel Problem Solving from Nature, Elsevier Science Publishers, Amsterdam, vol 2, pp 27–36

Mahfoud SW (1995) Niching methods for genetic algorithms. PhD thesis, University of Illinois, Urbana-Champaign

Mandelbrot BB (1983) The Fractal Geometry of Nature. W. H. Freeman

Manderick B, Spiessens P (1989) Fine-grained parallel genetic algorithms. In: Schaffer JD (ed) Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Mateo, pp 428–433

Mann HB, Whitney DR (1947) On a test of whether one of two random variables is stachastically larger than the other. Ann. Math. Stat. 18:50–60

Maresky J, Davidor Y, Gitler D, Aharoni G, Barak A (1995) Selectively desctructive re-start. In: Eshelman L (ed) Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Francisco, pp 144–150

Marti R (2003) Multi-start methods. In: Glover F, Kochenberger G (eds) Handbook of Metaheuristics, Kluwer Academic Publishers, Boston, MA, USA, pp 355–368

Matsui K (1999) New selection method to improve the population diversity in genetic algorithms. In: IEEE Int. Conf. on Systems, Man, and Cybernetics, pp 625–630

Mattiussi C, Waibel M, Floreano D (2004) Measures of diversity for populations and distances between individuals with highly reorganizable genomes. Evol. Comput. 12(4):495–515, cited By (since 1996): 6

Mauldin ML (1984) Maintaining diversity in genetic search. In: Nat. Conf. on Art. Int., Austin, pp 247–250

Meloni C, Naso D, Turchiano B (2003) Multi-objective evolutionary algorithms for a class of sequencing problems in manufacturing environments. In: Proc. of the IEEE Int. Conf. on Systems, Man and Cybernetics 1, pp 8–13

Merz P (2000) Memetic Algorithms for Combinatorial Optimization Problems: Fitness Landscapes and Effective Search Strategies. PhD thesis, Department of Electrical Engineering and Computer Science, University of Siegen, Germany

Merz P, Katayama K (2004) Memetic algorithms for the unconstrained binary quadratic programming problem. Biosystems 79(1–3):99–118

Michalewicz A, Schoenauer M (1996) Evolutionary algorithms for constrained parameter optimization problems. Evol. Comput. 4(1):1–32

Michalewicz Z (1991) A genetic algorithm for the linear transportation problem. IEEE Trans. Syst., Man, Cybern. 21(2):445–452

Michalewicz Z (1992) Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, New York

Michielssen E, Ranjithan S, Mittra R (1992) Optimal multilayer filter design using real coded genetic algorithms. Proc. IEEE 139(6):413–419

Mladenovic N, Hansen P (1997) Variable neighborhood search. Comput. Oper. Res. 24:1097–1100

Mori N, Yoshida J, Tamaki H, Kita H, Nichikawa YA (1995) Thermodynamical selection rule for genetic algorithms. In: IEEE Cong. on Evolutionary Computation, IEEE Press, New York, pp 188–192

Moscato P (1999) Memetic algorithms: a short introduction. In: Corne D, Dorigo M, Glover F (eds) New Ideas in Optimization, McGraw-Hill Ltd., UK, Maidenhead, UK, England, pp 219–234

Muhando E, Kinjo H, Yamamoto T (2006) Enhanced performance for multivariable optimization problems by use of gas with recessive gene structure. Artif. Life Robotics 10:11–17

Mühlenbein H (1989) Parallel genetic algorithms, population genetics and combinatorial optimization. In: Schaffer JD (ed) Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Mateo, pp 416–421

Mühlenbein H (1991) Evolution in Time and Space - The Parallel Genetic Algorithm as Function Optimizer, Foundations of Genetic Algorithms, vol 1, Morgan Kaufmann, San Mateo, pp 316–337

Mühlenbein H, Schlierkamp-Voosen D (1993) Predictive Models for the Breeder Genetic Algorithm I. Continuous Parameter Optimization. Evol. Comput. 1:25–49

Mühlenbein H, Schomisch M, Born J (1991) The parallel genetic algorithm as function optimizer. In: Belew R, Booker LB (eds) Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Mateo, pp 271–278

Mutoh A, Tanahashi F, Kato S, Itoh H (2006) Efficient real-coded genetic algorithms with flexible-step crossover 126(5):654–660

Nelder JA, Mead R (1965) A simplex method for function minimization. Computer Journal 7(4):308–313

Nguyen HD, Yoshihara I, Yamamori K, Yasunaga M (2007) Implementation of effective hybrid GA for large-scale traveling salesman problems. IEEE Trans. Syst., Man, Cybern. B 37(1):92–99

Noman N, Iba H (2008) Accelerating differential evolution using an adaptive local search. IEEE Trans. Evol. Comput. 12(1):107–125

Nsakanda AL, Price WL, Diaby M, Gravel M (2007) Ensuring population diversity in genetic algorithms: A technical note with application to the cell formation problem. Eur. J. Oper. Res. 178(2):634–638

Obayashi S, Deb K, Poloni C, Hiroyasu T, Murata T (eds) (2007) Evolutionary Multi-Criterion Optimization, Lecture Notes in Computer Science 4403, Springer Berlin, Heidelberg

O'Neill M, Ryan C (2001) Grammatical evolution. IEEE Trans. Evol. Comput. 5(4):349–358

O'Reilly UM, Oppacher F (1995) Hybridized crossover-based search techniques for program discovery. In: Proc. of the World Conf. on Evolutionary Computation, 2, pp 573–578

Papadakis SE, Theocharis JB (2002) A GA-based fuzzy modeling approach for generating TSK models. Fuzzy Sets and Systems 131(2):121–152

Paquete L, Schiavinotto T, Stützle T (2007) On local optima in multiobjective combinatorial optimization problems. Ann. Oper. Res. 156(1):83–97

Park JH, Choi M (2004) Generation of an optimal gait trajectory for biped robots using a genetic algorithm. JSME Int. J. Series C 47(2):715–721

Petalas YG, Parsopoulos KE, Vrahatis MN (2007) Memetic particle swarm optimization. Ann. Oper. Res. 156(1):99–127

Pétrowski A (1996) A clearing procedure as a niching method for genetic algorithms. In: Proc. of the IEEE Int. Conf. Evolutionary Computation, pp 798–803

Pezzella F, Morganti G, Ciaschetli G (2008) A genetic algorithm for the flexible job-shop scheduling problem. Comput. Oper. Res. 35(10):3202–3212

Poojari CA, Varghese B (2008) Genetic algorithm based technique for solving chance constrained problems. Eur. J. Oper. Res. 185(3):1128–1154

Potts JC, Giddens TD, Yadav SB (1994) The development and evaluation of an improved genetic algorithm based on migration and artificial selection. IEEE Trans. Syst., Man, Cybern. 24:73–86

Press WH, y otros (1989) Numerical Recipes: The Art of Scientific Computing. Cambridge University Press Cambridge, UK

Price KV, Storn R, Lampinen JA (2005) Differential Evolution: A Practical Approach To Global Optimization. Springer Berlin, Heidelberg

Qi X, Palmieri F (1994) Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space part ii: Analysis of the diversification role of crossover. IEEE Trans. Neural Netw. 5(1):102–119

Radcliffe NJ (1991) Equivalence class analysis of genetic algorithms. Complex Syst. 5(2):183–205

Radcliffe NJ (1992) Non-Linear Genetic Representations. In: Männer R, Manderick B (eds) Parallel Problem Solving from Nature, Elsevier Science Publishers, Amsterdam, vol 2, pp 259–268

Raich AM, Ghaboussi J (1997) Implicit representation in genetic algorithms using redundancy. Evol. Comput. 5(3):277–302

Randall M (2006) Search space reduction as a tool for achieving intensification and diversification in ant colony optimisation. In: Ali M, Dapoigny R (eds) Lecture Notes in Artificial Intelligence, 4031, Springer-Verlag, pp 254–262

Ray SS, Bandyopadhyay S, Pal SK (2007) Genetic operators for combinatorial optimization in TSP and microarray gene ordering. App. Intell. 26(3):183–195

Rechenberg I (1973) Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution. Frommann-Holzboog, Stuttgart

Rechenberg I (1994) Evolutionsstrategie'94. Frommann-Holzboog

Reeves CR (1993) Using genetic algorithms with small populations. In: Forrest S (ed) Int. Conf. on Genetic Algorithms, Morgan Kaufmann, pp 92–99

Reijmers TH, Wehrens R, Daeyaert FD, Lewi PJ, Buydens LMC (1999) Using genetic algorithms for the construction of phylogenetic trees: application to g-protein coupled receptor sequences. Biosystems 49:31–43

Renders J-M, Flasse SP (1996) Hybrid methods using genetic algorithms for global optimisation. IEEE Trans. Syst., Man, Cybern. 26(2):243–258

Resende MGC, Ribeiro CC (2003) Greedy randomized adaptive search procedures. In: Glover F, Kochenberger G (eds) Handbook of Metaheuristics, Kluwer Academic Publishers, Boston, MA, USA, pp 219–249

Ribeiro CC, Hansen P (2002) Essays and Surveys in Metaheuristics. Kluwer Academic Publishers

Rinnoy Kan AHG, Timmer GT (1989) Global Optimization, Handbooks in operations research and management science, vol 1, North-Holland, pp 631–662

Rosenbrock HH (1960) An automatic method for finding the greatest or least value of a function. The Computer Journal 3(3):175

Ruiz R, Maroto C (2005) A comprehensive review and evaluation of permutation flowshop heuristics. Eur. J. Oper. Res. 165(2):479–494

Sakanashi H, Suzuki K (1994) Controlling dynamics of ga through filtered evaluation function. In: Davidor Y, Schwefel HP, Männer R (eds) Parallel Problem Solving from Nature, Springer-Verlag, Berlín, vol 3, pp 329–248

Sareni B, Krahenbuhl L (1998) Fitness sharing and niching methods revisited. IEEE Trans. Evol. Comput. 2(3):97–106

Sastry K, Goldberg DE (2004) Designing competent mutation operators via probabilistic model building of neighborhoods. In: Proc. of the Conf. on Genetic and Evolutionary Computation, Lecture Notes in Computer Science 3103, pp 114–125

Schaffer JD, Morishima A (1987) An adaptive crossover ditribution mechanism for genetic algorithms. In: Grefenstette JJ (ed) Int. Conf. on Genetic Algorithms Applications and their application, pp 36–40

Schaffer JD, Caruana RA, Eshelman LJ, Das R (1989) A study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization. In: Schaffer JD (ed) Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Mateo, pp 51–60

Schlierkamp-Voosen D, Mühlenbein H (1994) Strategy adapataion by competing subpopulations. In: Davidor Y, Schwefel HP, Männer R (eds) Parallel Problem Solving from Nature, Springer-Verlag, Berlín, vol 3, pp 199–208

Schraudolph NN, Belew RK (1992) Dynamic parameter encoding for genetic algorithms. Mach. Learn. 9:9–21

Schuurman P, Vredeveld T (2007) Performance guarantees of local search for multiprocessor scheduling. INFORMS J. on Computing 19(1):52–63

Schwefel HP (1995) Evolution and Optimum Seeking. Wiley

Sebag M, Schownauer M (1994) Controlling crossover through inductive learning. In: Davidor Y, Schwefel HP, Männer R (eds) Parallel Problem Solving from Nature, Springer-Verlag, Berlín, vol 3, pp 209–218

Shaefer CG (1987) The ARGOT Strategy: Adaptive Representation Genetic Optimizer Technique. In: Grefenstette JJ (ed) Int. Conf. on Genetic Algorithms Applications and their application, pp 50–55

Shapiro SS, Wilk MB (1965) An analysis of variance test for normality (complete samples). Biometrika 52(3,4):591–611

Sheskin DJ (2000) The Handbook of Parametric and Nonparametric Statistical Procedures. Chapman & Hall/CRC

Shimodaira H (1996) A new genetic algorithm using large mutation rates and population-elitist selection (galme). In: IEEE Conf. on Tools with Artificial Intelligence, IEEE Computer Society Press, pp 25–32

Shimodaira H (1999) A diversity control oriented genetic algorithm (DCGA): Development and experimental results. In: Proc. of the Conf. on Genetic and Evolutionary Computation, Morgan Kaufmann, San Francisco, California, pp 603–611

Siarry P, Michalewicz Z (eds) (2008) Advances in Metaheuristics for Hard Optimization. Natural Computing, Springer

Siarry P, Berthiau G, Durbin F, Haussy J (1997) Enhanced simulated annealing for globally minimizing functions of many-continuous variables. ACM Trans. Math. Softw. 23(2):209–228

Siddall JN (1982) Optimal Engineering Design: Principles and Applications. Marcel Dekker Ltd

Smith K, Hoos HH, Stützle T (2003) Iterated robust tabu search for MAX-SAT. In: Carbonell JG, Siekmann J (eds) Proc. of the Canadian Society for Computational Studies of Intelligence Conf., Springer, Berlin Heidelberg, LNCS, vol 2671, pp 129–144

Smith RE (1993) Adaptively resizing populations: An algorithm and analysis. In: Forrest S (ed) Int. Conf. on Genetic Algorithms, pp 653–653

Sokolov A, Whitley D, Da Motta Salles Barreto A (2007) A note on the variance of rank-based selection strategies for genetic algorithms and genetic programming. Gen. Program. Evol. Mach. 8(3):221–237

Solis FJ, Wets RJ (1981) Minimization by random search techniques. Mathematical Operations Research 6:19–30

Someya H, Yamamura M (2005) A robust real-coded evolutionary algorithm with toroidal search space conversion. Soft Comput. 9(4):254–269

Spears WM (1992) Adapting crossover in a genetic algorithm. Tech. Rep. #AIC-92-025, Navy Center for Applied Research in Artificial Intelligence, EEUU

Spears WM (2000) Evolutionary Algorithms: The Role of Mutation and Recombination. Springer

Spears WM, De Jong KA (1991) On the virtues of parameterized uniform crossover. In: Proc. of the Int. Conf. on Genetic Algorithms, Morgan Kaufmann, pp 230–236

Srinivas M, Patnaik LM (1994) Adaptive probabilities of crossover and mutation in genetic algorithms. IEEE Trans. Syst., Man, Cybern. 24(4):656–667

Storn R, Price K (1997) Differential evolution–A simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. 11(4):341–359

Streichert F, Stein G, Ulmer H, Zell A (2004) A clustering based niching ea for multimodal search spaces. In: Proc. of the Int. Conf. on Evolution Artificielle, Lecture Notes in Computer Science 2936, pp 293–304

Swann WH (1964) A report on the development of a new direct searching method of optimization. ICI CENTER Instrument Laboratory, Middlesborough

Syswerda G (1989) Uniform crossover in genetic algorithms. In: Schaffer JD (ed) Proc. of the Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Francisco, CA, USA, pp 2–9

Syswerda G (1991) Schedule Optimization Using Genetic Algorithms, Van Nostrand Reinhold, New York, pp 332–349. Handbook of Genetic Algorithms

Takahashi O, Kobayashi S (2001) An adaptive neighboring search using crossover-like mutation for multi modal function optimization. In: Proc. of the IEEE Int. Conf. on System Man and Cybernetics, vol 1, pp 261–267

Tanese R (1987) Parallel genetic algorithms for a hypercube. In: Grefenstette JJ (ed) Int. Conf. on Genetic Algorithms Applications and their application, pp 177–183

Tate DM, Smith AE (1993) Expected allele coverage and the role of mutation in genetic algorithms. In: Forrest S (ed) Int. Conf. on Genetic Algorithms, Morgan Kaufmann, pp 31–36

Tate DM, Smith AE (1995) A genetic approach to the quadratic assignment problem. Comput. Oper. Res. 22(1):73–83

Thierens D (2004) Population-based iterated local search: restricting neighborhood search by crossover. In: Deb K (ed) Proc. of the Genetic and Evolutionary Computation Conf., Springer, Berlin Heidelberg, LNCS, vol 3103, pp 234–245

Toffolo A, Benini E (2003) Genetic diversity as an objective in multi-objective evolutionary algorithms. Evol. Comput. 11(2):151–167

Tohka J, Krestyannikov E, Dinov ID, Graham AM, Shattuck DW, Ruotsalainen U, Toga AW (2007) Genetic algorithms for finite mixture model based voxel classification in neuroimaging. IEEE Trans. Med. Imag. 26(5):696–711

Tsai H, Yang J, Tsai Y, Kao C (2004a) Some issues of designing genetic algorithms for traveling salesman problems. Soft Comput. 8(10):689–697

Tsai H-K, Yang J-M, Tsai Y-F, Kao C-Y (2004b) An evolutionary algorithm for large traveling salesman problems. IEEE Trans. Syst., Man, Cybern. B 34(4):1718–1729

Tsutsui S, Fujimoto Y (1993) Forking genetic algorithm with blocking and shrinking modes. In: Forrest S (ed) Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Mateo, California, pp 206–213

Tsutsui S, Ghosh A, Corne D, Fujimoto Y (1997) A real coded genetic algorithm with an explorer and an exploiter population. In: Bäck T (ed) Proc. of the Int. Conf. on Genetic Algorithms, Morgan Kaufmann Publishers, San Francisco, pp 238–245

Tsutsui S, Yamamura M, Higuchi T (1999) Multi-parent recombination with simplex crossover in real coded genetic algorithms. In: Genetic Evolutionary Computation Conf (GECCO), pp 657–664

Valenzuela CL (2001) A study of permutation operators for minimum span frequency assignment using an order based representation. Journal of Heuristics 7(1):5–21

van Kemenade CHM, Kok JN, Eiben AE (1995) Raising ga performance by simultaneous tuning of selection pressure and recombination disruptiveness. In: IEEE Conf. on Evolutionary Computation, IEEE Press, pp 345–351

Venugopal V, Narendran TT (1992) A genetic algorithm approach to the machine-component grouping problem with multiple objectives. Comput. Ind. Eng. (4):469–480

Voigt HM, Anheyer T (1994) Modal mutations in evolutionary algorithms. In: IEEE Conf. on Evolutionary Computation, pp 88–92

Voigt HM, Mühlenbein H, Cvetkovic D (1995) Fuzzy recombination for the breeder genetic algorithm. In: Int. Conf. on Genetic Algorithms, Morgan Kaufmann, pp 104–113

Vose MD (1991) Generalizing the Notion of Schemata in Genetic Algorithm. Artif. Intell. 50:385–396

Voß S, Osman IH, Roucairol C (1999) Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization. Kluwer Academic Publishers Norwell, MA, USA

Voudouris C, Tsang E (1999) Guided local search. Eur. J. Oper. Res. 113(2):469–499

Walser JP, Kautz H (1999) Integer Optimization by Local Search: A Domain-Independent Approach. Springer

Weicai Z, Jing L, Mingzhi X, Licheng J (2004) A multiagent genetic algorithm for global numerical optimization. IEEE Trans. Syst., Man, Cybern. B 34(2):1128–1141

White T (1994) Adaptive crossover using automata. In: Davidor Y, Schwefel HP, Männer R (eds) Parallel Problem Solving from Nature, Springer-Verlag, pp 228–237

Whitley D (1989) The GENITOR algorithm and selection pressure: why rank-based allocation of reproductive trials is best. In: Schaffer JD (ed) Proc. of the Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Francisco, CA, USA, pp 116–121

Whitley D (1990) Genitor-II: A Distributed Genetic Algorithm. J. Exp. Theor. Artif. Intell. 2:189–214

Whitley D, Hanson T (1989) Optimizing Neural networks Using Faster, More Accurate Genetic Search. In: Schaffer JD (ed) Int. Conf. on Genetic Algorithms, Morgan Kaufmann, pp 391–396

Whitley D, Rowe JE (2005) Gray, binary and real valued encodings: quad search and locality proofs. In: Foundations of Genetic Algorithms, LNCS 3469, Springer Berlin, Heidelberg, pp 21–36

Whitley D, Starkweather T, Fuquay D (1989) Scheduling problems and traveling salesmen: The genetic edge recombination operator. In: Schaffer JD (ed) Int. Conf. on Genetic Algorithms, Morgan Kaufmann, pp 133–140

Whitley D, Mathias K, Fitzhorn P (1991) Delta coding: An iterative search strategy for genetic algorithms. In: Belew R, Booker LB (eds) Int. Conf. on Genetic Algorithms, Morgan Kaufmann, pp 77–84

Wilcoxon F (1945) Individual comparisons by ranking methods. Biometrics 1:80–83

Wilson SW (1986) Classifier system learning of a boolean function, research Memo RIS-27r, Rowland Institute for Science, Cambridge Mass

Winter G, Galvan B, Alonso S, Gonzalez B, Jiménez JI, Greiner D (2005) A flexible evolutionary agent: Cooperation and competition among real-coded evolutionary operators. Soft Comput. 9(4):299–323

Wright A (1991) Genetic Algorithms for Real Parameter Optimization, Foundations of Genetic Algorithms, vol 1, Morgan Kaufmann, San Mateo, pp 205–218

Wright S (1934) The roles of mutation, inbreeding, crossbreeding and selection in evolution. In: Int. Cong. of Genetics, vol 1, pp 356–366

Xiao P, Vadakkepat P, Lee TH (2008) Context-dependent dna coding with redundancy and introns. IEEE Trans. Syst., Man, Cybern. B 38(2):331–341

Yan H, Zheng J, Jiang Y, Peng C, Xiao S (2008) Selecting critical clinical features for heart diseases diagnosis with a real-coded genetic algorithm. App. Soft Comput. 8(2):1105–1111

Yang J-M, Kao C-Y (2000) Integrating adaptive mutations and family competition into genetic algorithms as function optimiser. Soft Comput. 4:89–102

Yang Y, Vincent J, Littlefair G (2004) A coarse-grained parallel genetic algorithm employing cluster analysis for multi-modal numerical optimisation. In: Proc. of the Int. Conf. on Evolution Artificielle, Lecture Notes in Computer Science 2936, pp 229–240

Yu T, Sastry K, Goldberg DE (2007) Population sizing to go: Online adaptation using noise and substructural measurements. In: Parameter Setting in Evolutionary Algorithms, vol 54, Springer, Berlín, pp 205–223

Yuret D, de la Maza M (1993) Dynamic hill climbing: Overcoming the limitations of optimization techniques. In: Proc. of the Turkish Symposium on Artificial Intelligence and Artificial Neural Networks, pp 208–212

Zar JH (1999) Biostatistical Analysis. Prentice Hall

Zhang L, Wang L, Zheng D (2006) An adaptive genetic algorithm with multiple operators for flowshop scheduling. Int. J. Adv. Manuf. Technol. 27(5–6):580–587

Zimmermann HJ (1990) Fuzzy Set Theory and its Applications. Kluwer

Zitzler E, Deb K, Thiele L, Coello CA, Corne D (eds) (2001) Evolutionary Multi-Criterion Optimization, Lecture Notes in Computer Science 1993, Springer Berlin, Heidelberg