UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
DEPARTAMENTO DE INTELIGENCIA ARTIFICIAL

TÉCNICAS NOVEDOSAS DE EMPAREJAMIENTO EN ALGORITMOS GENÉTICOS

TRABAJO FIN DE MÁSTER
EN
INTELIGENCIA ARTIFICIAL AVANZADA: FUNDAMENTOS, MÉTODOS Y APLICACIONES

RAFAEL PINTER GÓMEZ
LICENCIADO EN CIENCIAS FÍSICAS POR LA UNIVERSIDAD DE VALENCIA

DIRECCIÓN:
SEVERINO FERNÁNDEZ GALÁN
PROFESOR TITULAR DE UNIVERSIDAD
DEL DEPARTAMENTO DE INTELIGENCIA ARTIFICIAL DE LA UNED

SEPTIEMBRE DE 2011

ÍNDICE

RESUMEN

En la resolución de problemas de optimización mediante algoritmos genéticos, normalmente se realiza un emparejamiento aleatorio de padres como paso previo a la fase de recombinación. Galán y Mengshoel propusieron en 2010 [Galán y Mengshoel (2010)] un nuevo método de emparejamiento de padres en algoritmos genéticos, que hace uso de un parámetro denominado "índice de emparejamiento", para definir la estrategia de emparejamiento de los padres. Dependiendo del problema de optimización planteado, el mejor comportamiento (superior al método de emparejamiento aleatorio tradicional) se obtiene para un valor concreto del índice de emparejamiento. Dado que el valor del índice de emparejamiento que proporciona mejores resultados para un problema dado es desconocido de antemano, Galán y Mengshoel desarrollaron un método de control autoadaptativo del índice de emparejamiento. Este método autoadaptativo fue aplicado a la optimización de funciones reales mediante algoritmos genéticos con representación binaria, aunque los resultados obtenidos no fueron satisfactorios en su conjunto para el caso de funciones con muchos óptimos locales.

El presente trabajo fin de máster tiene como objetivo investigar nuevos métodos de emparejamiento de padres que permitan mejorar los resultados obtenidos mediante el método autoadaptativo de Galán y Mengshoel. En concreto, se proponen y evalúan experimentalmente dos nuevos métodos de control del índice de emparejamiento, denominados respectivamente "método espacial" y "método temporal". Por otra parte, también evaluamos experimentalmente si el uso de una representación del cromosoma más cercana al problema de optimización de funciones reales, como es el uso de genes reales en vez de los binarios empleados por Galán y Mengshoel, puede influir en un mejor comportamiento del emparejamiento autoadaptativo para el problema de optimización de funciones reales de varias variables.

El grado de consecución de los objetivos marcados en este trabajo fin de máster ha sido variado. Por una parte, la representación real introducida para el problema de optimización

de funciones reales ha resultado beneficiosa. Por otra parte, aunque el control espacial del índice de emparejamiento no ha resultado todo lo ventajoso que hubiéramos esperado, el método temporal ha aportado importantes ventajas. Todo ello queda motivado, explicado y evaluado en detalle a lo largo de la memoria.

AGRADECIMIENTOS

Quiero manifestar mi sincero agradecimiento al Prof. Dr. D. Severino Fernández Galán por su guía, apoyo y certeros consejos. Su dedicación y proximidad me ha hecho sentirme muy a gusto realizando este trabajo.

CAPÍTULO 1 INTRODUCCIÓN

Un problema que se plantea habitualmente en la actividad humana es el de la optimización. Continuamente aparece la necesidad de obtener el mejor uso posible de unos recursos limitados, o cómo combinarlos de forma que produzcan el mejor rendimiento o cómo conseguir los mejores resultados con los recursos disponibles. El presente trabajo de fin de máster trata de un problema de optimización.

1.1 Explicación del problema

El problema que planteamos es la optimización de funciones reales n-dimensionales de variable real (unimodales y multimodales) utilizando algoritmos evolutivos. Las funciones que estudiamos proporcionan un problema estándar ampliamente conocido. El grado de dificultad se puede variar con facilidad, eligiendo el tipo de función y la dimensión, desde funciones unimodales, relativamente fáciles de optimizar hasta funciones multimodales, más difíciles de optimizar debido a la existencia de muchos óptimos locales.

Los algoritmos evolutivos, inspirados en la biología, actúan, como en la evolución natural, bajo las fuerzas de la variación (recombinación y mutación) y selección de los individuos más aptos en cada generación. Dos tendencias han de quedar compensadas: la exploración de diferentes regiones del espacio de búsqueda (que puede permitirnos salir de óptimos locales) y la explotación de los mejores individuos (que nos permite avanzar rápidamente hacia la solución óptima). De la correcta compensación de estas dos tendencias depende el éxito del algoritmo evolutivo en la resolución del problema de optimización planteado.

Un algoritmo evolutivo pasa por varias fases que se repiten hasta que se cumple la condición de terminación. Estas fases son: selección de padres, recombinación, mutación, evaluación del grado adaptación* de los nuevos individuos y selección de supervivientes para la siguiente generación.

* Fitness en inglés

Entre la selección de padres y la recombinación puede haber una fase de emparejamiento*, en la que agrupamos a los padres, normalmente por parejas. En la posterior fase en recombinación se obtendrán hijos a partir de cada pareja de padres.

En este trabajo nos centraremos en la fase de emparejamiento del algoritmo evolutivo y su influencia en el resultado final del proceso de optimización.

1.2 Motivación y objetivos

En [Galán y Mengshoel (2010)] (véase el Anexo I de esta memoria), Galán y Mengshoel propusieron una nueva estrategia de emparejamiento para algoritmos evolutivos. Esta nueva estrategia permite mejorar los resultados del algoritmo evolutivo, adaptándolo al tipo de problema (unimodal o multimodal). Sin embargo, al intentar una versión del algoritmo que se adaptara él mismo a las características del problema a solucionar (autoadaptativo), los resultados no son buenos en todos los casos: aunque el algoritmo funciona bien en problemas unimodales, no tiene un comportamiento robusto en problemas multimodales.

En [Galán y Mengshoel (2010)] se usó representación binaria para el algoritmo genético. Uno de los objetivos del presente trabajo de fin de máster es ver si mejoran los resultados usando representación real. Para ello desarrollaremos un algoritmo evolutivo que use representación real y se repetirán los experimentos desarrollados en [Galán y Mengshoel (2010)] usando el nuevo software.

Un segundo objetivo es buscar una forma de controlar los parámetros de la fase de emparejamiento para obtener un método más robusto, o sea un método que obtenga buenos resultados tanto en problemas unimodales como en problemas multimodales.

1.3 Contribuciones del presente trabajo

Como resultado de este trabajo se han confirmado, usando representación real, los resultados que se obtuvieron por [Galán y Mengshoel (2010)] usando representación binaria. En el caso del método *Autoadaptativo*, con la representación real se obtienen mejores resultados que con la representación binaria, aunque no se llega al método robusto deseado.

* Mating en inglés

En la misma línea de conseguir mejorar los resultados obtenidos por el método *Autoadaptativo*, se planteó el método que hemos denominado *Espacial-1*, original de este trabajo de fin de máster. Los resultados obtenidos parecen indicar el camino de futuros trabajos en el estudio de la fase de emparejamiento en algoritmos genéticos.

Otro método original de este trabajo, el método *Temporal*, también sigue la línea de intentar mejorar los resultados obtenidos por el método *Autoadaptativo* El método *Temporal* es un método aceptablemente robusto que facilita la optimización de cualquier tipo de función. Además, su comportamiento se puede ajustar al tipo de problema planteado de forma más versátil que el resto de métodos.

1.4 Organización de la memoria

En el capítulo 2 se hace una breve descripción de los algoritmos evolutivos, con especial atención a la fase de emparejamiento, explicando el método de emparejamiento propuesto en [Galán y Mengshoel (2010)], que es el que pretendemos mejorar en el presente trabajo. En el capítulo 3 se describe nuestra propuesta, es decir el cromosoma con representación y real y los nuevos métodos espacial y temporal. En el capítulo 4 se explican las característica del software desarrollado, implementando los algoritmos descritos en los capítulos 2 y 3, para realizar los experimentos. En el capítulo 5 se presentan y discuten los resultados experimentales. Finalmente, en el capítulo 6 comentamos algunas conclusiones de los resultados obtenidos y proponemos futuros trabajos en la misma línea.

CAPÍTULO 2 LA COMPUTACIÓN EVOLUTIVA

El presente trabajo fin de máster trata de la resolución de problemas de optimización de funciones reales aplicando técnicas de computación evolutiva. Por ello, el presente capítulo presenta una introducción al campo de la Computación Evolutiva, haciendo especial énfasis en la fase en emparejamiento de un algoritmo evolutivo.

2.1 Definición y Objetivos de la Computación Evolutiva

La computación evolutiva [Holland (1992); Goldberg (1989); Eiben y Smith (2003)] trata de resolver problemas de optimización mediante algoritmos inspirados en la teoría de la evolución de Darwin: la selección natural.

En la Naturaleza, los individuos compiten por unos recursos limitados. En cada generación, los individuos mejor adaptados tienen más opciones de sobrevivir y reproducirse, transmitiendo su material genético a su descendencia. Por otra parte, en cada generación se producen variaciones genéticas (por la vía de la recombinación de genes inherente al mecanismo de reproducción sexual y por la vía de la mutación) que se pueden manifestar en individuos cada vez mejor adaptados a su entorno.

El objetivo de la computación evolutiva es la solución de problemas de optimización. Consideremos un sistema objeto de análisis formado por tres componentes: entradas al sistema, modelo (el conocimiento de cómo el sistema procesa las entradas para obtener las salidas) y las salidas del sistema. Podemos entonces distinguir entre tres tipos de problemas:

1) Problemas de optimización de la entrada. Conocemos el modelo y las salidas y hay que encontrar las entradas que conducen a esas salidas según el modelo.

Figura 2.1: Representación de un problema de optimización de la entrada

Ejemplos de este tipo de problemas son el problema del viajante, la construcción de horarios o la optimización de funciones. En este trabajo, la parte experimental trata precisamente sobre optimización de funciones reales.

2) Problemas de optimización del modelo . En este caso conocemos las entradas y sus salidas correspondientes y se trata de encontrar un modelo del sistema que sea capaz de generar las salidas correctas para cada entrada conocida.

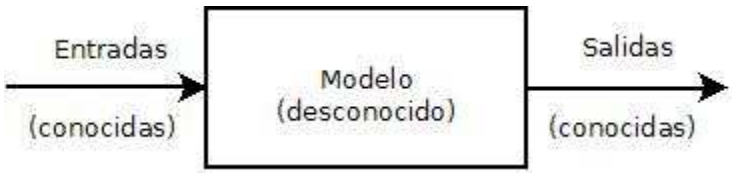

Figura 2.2: Representación de un problema de optimización del modelo

Ejemplos de este tipo de problemas son el modelado del mercado de valores o los modelos macroeconómicos.

3) Problemas de optimización de la salida. Conocemos el modelo del sistema y las entradas y necesitamos calcular las salidas correspondientes a esas entradas.

Figura 2.3: Representación de un problema de optimización de la salida

Se puede citar la simulación de circuitos electrónicos como un problema que entra dentro de esta categoría.

En los años sesenta se desarrollaron tres implementaciones diferentes de la idea básica descrita más arriba: *programación evolutiva* [Fogel, Owens y Walsh (1965); Fogel, Owens y Walsh (1966)], *algoritmos genéticos* [De Jong (1975); Holland (1973); Holland (1992)] y *estrategias evolutivas* [Rechenberg (1973); Schwefel (1995)]. En los años noventa aparece una nueva implementación denominada *programación genética* [Banzhaf,

Nordin y Keller (1999); Koza (1994)]. Actualmente estas cuatro implementaciones se consideran variantes dentro del campo que se ha venido a denominar *computación evolutiva* [Bäck (1996); Michalewicz (1996)].

2.2 Fases de un algoritmo evolutivo

De forma genérica, un algoritmo evolutivo funciona de la siguiente forma: dada una población de individuos, la competencia entre ellos por resolver mejor un determinado problema (equivalente a la mejor adaptación al entorno en el caso de la biología) hará que sobrevivan y se reproduzcan los mejores (los mejor adaptados) haciendo que, en este proceso, aumente la calidad media de la población para resolver el problema.

Como ocurre en la biología, hay dos fuerzas actuando cada generación sobre la población: por una parte, los operadores de variación (recombinación y mutación) se encargan de aumentar la diversidad de la población, promoviendo la aparición de nuevos individuos; por otra parte, la selección aumenta la calidad media de los individuos (o lo que es lo mismo de las soluciones al problema).

El esquema general de un algoritmo evolutivo es el siguiente [Eiben y Smith (2003)]:

```
INICIO
    INICIALIZA población con individuos al azar;
    EVALÚA cada individuo;
    REPITE HASTA (CONDICIÓN DE TERMINACIÓN)
        SELECCIONA padres;
        RECOMBINA pares de padres;
        MUTA la descendencia obtenida;
        EVALÚA los nuevos individuos;
        SELECCIONA individuos para la siguiente generación;
    FIN DE REPITE
FIN
```

Figura 2.4: Esquema general de un algoritmo evolutivo

El algoritmo empieza **inicializando** al azar una población de individuos. La población es un conjunto de individuos y es lo que realmente va cambiando (evolucionando) generación tras generación. Cada individuo representa una posible solución y al decir

"representa" queremos decir que el individuo es una imagen computacional de una solución en el mundo real. Normalmente, el genotipo de los individuos de la primera generación se obtiene al azar sobre todo el dominio del problema.

El siguiente paso es **evaluar** cada individuo. Para ello se utiliza una función de adaptación, que asigna a cada individuo un valor que indica el grado en que ese individuo es una buena solución para el problema. Por ejemplo, en un problema de maximización, cuanto mayor sea el valor asignado por la función al individuo, mejor adaptado estará ese individuo; en otras palabras, la solución candidata asociada al individuo será mejor ya que se aproxima más al valor de adaptación máximo buscado.

El resto de fases se repiten hasta que se cumple la **condición de terminación**. Como condición de terminación podemos tomar el hecho de que haya un determinado número de generaciones sin mejora en la solución propuesta (o con dicha mejora por debajo de un determinado valor umbral), que la diversidad de la población caiga por debajo de un valor umbral o que se hayan producido un número determinado de generaciones. Es conveniente que el número de generaciones (o equivalentemente el tiempo de CPU) aparezca como una de las condiciones para terminar, de cara a garantizar la finalización del algoritmo.

Dentro del bucle de la figura 2.4 se repiten las siguientes fases:

En primer lugar tenemos la **selección de padres**, que trata de seleccionar los mejores individuos para que sean los padres de la siguiente generación. La selección de padres, junto con la selección de supervivientes que veremos a continuación, es la responsable de realizar la presión selectiva sobre la población para mejorar la calidad de los individuos. Se trata de un proceso estocástico: los mejores individuos tienen una probabilidad mayor de convertirse en padres, pero un individuo con peor valor de adaptación tiene también cierta probabilidad (pequeña pero positiva) de convertirse en padre. El presente trabajo está centrado en la forma particular de emparejar los padres seleccionados en esta fase.

En la siguiente fase, los padres seleccionados se recombinan para obtener descendencia. En la mayoría de los casos la **recombinación** es un operador binario en el sentido de que mezcla información del genotipo de dos padres para obtener uno o dos descendientes. Detrás de la recombinación está el mismo principio que se sigue para obtener nuevas variedades de animales o plantas: si recombinamos padres con unas determinadas características que nos interesan pueden aparecer descendientes que reúnan características deseables de ambos padres.

La **mutación** es un operador unario. Se aplica a un genotipo y produce un descendiente (mutante) ligeramente modificado. Se trata de un operador que es siempre estocástico ya que debe producir un cambio imparcial, al azar. La mutación garantiza que toda solución del espacio de búsqueda tenga cierta probabilidad de ser alcanzada.

A los nuevos individuos obtenidos mediante la recombinación y la mutación se les debe asignar un valor de adaptación, de forma que al pasar a la siguiente fase todos los individuos presentes en la población tengan un valor de adaptación asignado.

La última fase del ciclo evolutivo es la **selección de supervivientes**. De la población total formada por los padres y su descendencia hay que seleccionar los individuos que formarán la población de la siguiente generación. Así como la selección de padres es estocástica, la selección de supervivientes es frecuentemente determinista y normalmente se mantiene constante el tamaño de la población en cada generación. Para ello suele realizarse una selección de los mejores individuos de entre el conjunto formado por la población original y la descendencia hasta que se mantenga un tamaño de población igual al de la población original. A veces se produce un reemplazo generacional completo, descartando todos los individuos de la población original y realizando la selección únicamente entre la descendencia.

Una vez se ha producido el reemplazamiento de la población se inciaría un nuevo ciclo hasta que se diera la condición de terminación que detuviera el proceso.

2.3 Fase de emparejamiento

Normalmente, después de la selección de padres y antes de la recombinación los individuos tienen que ser emparejados. La forma usual de emparejar padres en algoritmos genéticos consiste en tomar un padre de un conjunto de individuos y seleccionar su pareja escogiendo uniformemente al azar entre los restantes individuos del conjunto. Los padres emparejados de esta forma son extraídos fuera del conjunto y se repite el proceso hasta que no queden más padres por emparejar. El presente trabajo se centra en la fase de emparejamiento en algoritmos genéticos.

Hay otras técnicas de “emparejamiento restringido” que no seleccionan una pareja uniformemente al azar. Podemos citar las siguientes: *assortive mating genetic algorithms* [Fernandes (2001); Huang (2001); Ochoa (2005)], *correlative tournament selection* [Matsui (1999)], *seduction* [Ronald (1995)], *tabu genetic algorithm* [Ting (2003)] y *evolving agents*

[Smith (2000); Smith y Bonacina (2003); Unemi y Nagayishi (1997)]. A continuación describimos brevemente dos de las más importantes.

El *fitness sharing* [Deb y Goldberg (1989)] distribuye a los individuos en nichos de adaptación (zonas alrededor de cada óptimo local) proporcionalmente al valor de adaptación de cada nicho. Se promueve un emparejamiento al azar pero restringido dentro de los individuos de cada nicho. Para ello, dado un padre, su pareja es aceptada sólo si la distancia entre ellos es inferior a un determinado umbral. Si la distancia es superior al umbral se busca otro candidato. En caso de no encontrar ningún candidato aceptable se elige uno al azar.

Contrariamente la técnica denominada *incest prevention* [Eshelman y Schaffer (1991)] promueve un emparejamiento restringido a individuos que son lo suficientemente distintos. En este caso el candidato a pareja sería aceptado sólo si la distancia al otro padre es superior a un determinado umbral.

El nuevo método de emparejamiento del que vamos a tratar en la siguiente sección entra dentro de lo que hemos denominado “emparejamiento restringido”. Sin embargo, así como *fitness sharing* juega la baza de la explotación e *incest prevention* la de la exploración, el nuevo método va a permitir ajustar el grado de “exploración – explotación” e incluso permite, de forma sencilla, implementar un método autoadaptativo que ajuste el grado de “exploración – explotación” adaptándose al problema. Hemos decidido incluir el trabajo mencionado, [Galan y Mengshoel, 2010], como Anexo I a esta memoria, ya que es la base del presente trabajo fin de máster. De cualquier forma, en la siguiente sección realizamos un resumen exhaustivo de dicho trabajo.

2.4 Un nuevo método de emparejamiento

Los métodos de emparejamiento restringido basados en semejanza, descritos en la sección anterior, producen una exploración más efectiva del espacio de búsqueda, tanto en el caso de *fitness sharing* [Deb y Goldberg (1989)] como en *incest prevention* [Eshelman y Schaffer (1991)]. Sin embargo estas dos técnicas son opuestas y poco generales, es decir sólo se aplican a situaciones muy concretas. Por este motivo Galán y Mengshoel introdujeron en [Galán y Mengshoel (2010)] un marco uniforme y general para técnicas de emparejamiento restringido según semejanza. En ese trabajo se analiza también un criterio de emparejamiento según valor de adaptación (y no sólo según semejanza), como una alternativa de menor coste computacional. En esta sección se describe el nuevo método de emparejamiento siguiendo [Galán y Mengshoel (2010)].

El nuevo método de emparejamiento tiene tres características principales: en primer lugar, utiliza un parámetro denominado *índice de emparejamiento* que permite controlar el grado de exploración, haciendo el método general y flexible. En segundo lugar, permite definir las preferencias de emparejamiento en términos de semejanza entre individuos o de valor de adaptación de los individuos. En tercer lugar se presta fácilmente a una implementación autoadaptativa en la que cada individuo de la población tiene su propio *índice de emparejamiento*.

El nuevo método de emparejamiento propuesto en [Galán y Mengshoel (2010)] queda definido por el siguiente algoritmo:

Algoritmo 1 (Fase de emparejamiento):

Entradas:

- P_s Población de padres seleccionados
- γ Tamaño de emparejamiento: Número de padres elegibles para el siguiente emparejamiento
- cr Criterio usado para definir las preferencias de emparejamiento ($cr \in \{semejanza, fitness\}$)
- α Índice de emparejamiento: Entero usado para definir una preferencia de emparejamiento ($2 \leqslant \alpha \leqslant \gamma$)

Salidas:

- P_m Población de padres emparejados

Proceso:

1. Elegir γ padres uniformemente al azar de P_s sin reemplazamiento. Llamamos Ch al conjunto de padres elegidos.

2. Sea p_1 el padre con mejor valor de adaptación en Ch. Eliminar p_1 de P_s y Ch e incluirlo en P_m. El conjunto Ch ahora contiene los candidatos a pareja de p_1

3. Ordenar los $\alpha - 1$ mejores candidatos en Ch según el criterio cr. Cuando $cr = semejanza$, los candidatos quedan ordenados según la semejanza de su fenotipo con el de p_1. Cuando $cr = fitness$, los candidatos con mejor valor de adaptación son colocados los primeros. Sea $B_{\alpha-1}(p_1) = \{b_2, b_3, \ldots, b_{\alpha-1}, b_\alpha\}$ el conjunto de los $\alpha - 1$

mejores candidatos ordenados bajo el criterio cr .

4. Elegir $p_2 = b_\alpha$ como pareja para p_1 , eliminar p_2 de P_s e incluir p_2 en P_m . Hay que resaltar que p_1 y p_2 ocupan posiciones contiguas en P_m .

5. Volver al paso 1 si P_s aún no está vacío.

Figura 2.5: Algoritmo 1. Fase de emparejamiento en algoritmos genéticos: nuevo método de emparejamiento

Básicamente, el algoritmo funciona de la siguiente forma: son elegidos al azar con probabilidad uniforme γ (tamaño de emparejamiento) padres diferentes y el de mejor valor de adaptación es emparejado con otro individuo que queda determinado por cr (criterio de emparejamiento) y α (índice de emparejamiento). El criterio tradicionalmente usado para establecer preferencias de emparejamiento en algoritmos genéticos ha sido el de semejanza en el espacio de fenotipos ([Deb y Goldberg (1989); Eshelman y Schaffer (1991)]). Aquí se ha introducido un nuevo criterio basado en valores de adaptación.

El nuevo método de emparejamiento definido por el algoritmo 1 de la figura 2.5 coincide con el emparejamiento tradicional aleatorio cuando $\gamma = 2$. Por otra parte, los diferentes valores de α corresponden a un determinado balance explotación - exploración , entre $\alpha = 2$ (máxima explotación) y $\alpha = \gamma$ (máxima exploración).

Podemos distinguir tres variantes de interés del algoritmo 1:

a) Emparejamiento *Best-First*.

Cuando tomamos $\alpha = 2$, el mejor padre es emparejado con el primer candidato a pareja según el criterio cr , por lo que el método resultante es llamado *Best-First*.

Se trata de una estrategia que busca la máxima explotación. Si el criterio cr es *fitness*, el padre con mejor valor de adaptación es emparejado con el mejor de los restantes candidatos. Si el criterio es la semejanza, el padre con mejor valor de adaptación es emparejado con el candidato de entre los restantes más semejante a él que, presumiblemente, tendrá también un alto valor de adaptación. Por lo tanto, en ambos casos se emparejan padres con valor de adaptación alto, estrategia claramente explotativa.

En el caso de la estrategia de emparejamiento *Best-First*, el algoritmo 1 (caso general) puede simplificarse, ya que en el paso 3 sólo necesitamos el primer candidato según el criterio cr. Los pasos 3 y 4 del algoritmo 1 de la figura 2.5 se pueden sustituir por el algoritmo 2 que es descrito a continuación:

Algoritmo 2 (elección de pareja Best-First):

Entradas:

p_1	Individuo a ser emparejado
Ch	Conjunto individuos candidatos a pareja de p_1
cr	Criterio usado para definir las preferencias de emparejamiento ($cr \in \{semejanza, fitness\}$)
P_s	Población de padres seleccionados
P_m	Población de padres emparejados

Salidas:

p_2	Pareja de p_1

Proceso:

1. Hacer p_2 = Primer candidato en Ch según el criterio cr.

2. Eliminar p_2 de P_s.

3. Incluir p_2 en P_m.

Figura 2.6: Algoritmo 2. Elección de pareja Best-First (pasos 3 y 4 del Algoritmo 1)

b) Emparejamiento *Best-Last*.

Cuando tomamos $\alpha = \gamma$, el mejor padre es emparejado con el último candidato a pareja según el criterio cr, por lo que el método resultante es denominado *Best-Last*.

Se trata de una estrategia que busca la máxima exploración del espacio de posibles soluciones. Si el criterio cr es *fitness*, el padre con mejor valor de adaptación es emparejado con el peor de los restantes candidatos que normalmente estará alejado de él. Si el criterio es la semejanza, el padre con mejor valor de adaptación es emparejado con el candidato de entre los restantes más distante de él. Por lo tanto, en ambos casos se

emparejan padres distantes dentro del espacio de fenotipos, lo cual se traduce en una estrategia claramente explorativa.

En el caso de la estrategia de emparejamiento *Best-Last*, el algoritmo 1 (caso general) puede simplificarse ya que en el paso 3 sólo necesitamos el último candidato según el criterio cr . Los pasos 3 y 4 del algoritmo 1 (figura 2.5) se pueden sustituir por el algoritmo 3, que indicamos seguidamente:

Algoritmo 3 (elección de pareja Best-Last):

Entradas:

p_1	Individuo a ser emparejado
Ch	Conjunto individuos candidatos a pareja de p_1
cr	Criterio usado para definir las preferencias de emparejamiento ($cr \in \{semejanza, fitness\}$)
P_s	Población de padres seleccionados
P_m	Población de padres emparejados

Salidas:

p_2	Pareja de p_1

Proceso:

1. Hacer p_2 = Último candidato en Ch según el criterio cr .

2. Eliminar p_2 de P_s .

3. Incluir p_2 en P_m .

Figura 2.7: Algoritmo 3. Elección de pareja Best-Last (pasos 3 y 4 del Algoritmo1)

c) Emparejamiento *Autoadaptativo*.

Como veremos en la parte experimental, *Best-first* da mejores resultados en el caso de problemas de optimización que requieran más explotación (problemas unimodales con un óptimo global y sin óptimos locales), mientras que *Best-last* funciona mejor cuando es necesaria más exploración (problemas multimodales que contienen óptimos locales). Buscando un algoritmo robusto para cualquier tipo de problemas se propuso el método *Autoadaptativo*.

El método *Autoadaptativo* consiste en codificar el índice de emparejamiento dentro del cromosoma de cada individuo quedando sometido como el resto de genes a recombinación y mutación. Lo que esperamos es que el valor del índice de emparejamiento se adapte al problema concreto a resolver de forma que obtengamos buenos resultados independientemente de si nos encontramos con problemas unimodales o multimodales. La representación del individuo j, que en el caso no autoadaptativo es $\langle x_{j,1},...,x_{j,n}\rangle$, pasa ahora a ser $\langle x_{j,1},...,x_{j,n},x_{j,n+1}\rangle$ con $x_{j,n+1}=\alpha_j$.

En este caso, α ya no es un input del algoritmo (como en algoritmo 1), porque ahora está codificado en el propio cromosoma. De esta forma, los pasos 3 y 4 del algoritmo 1 de la figura 2.5 se pueden sustituir por el algoritmo 4 tal como se indica a continuación:

Algoritmo 4 (elección de pareja Autoadaptativa):

Entradas:

p_1	Individuo a ser emparejado
Ch	Conjunto individuos candidatos a pareja de p_1
cr	Criterio usado para definir las preferencias de emparejamiento ($cr \in \{semejanza, fitness\}$)
P_s	Población de padres seleccionados
P_m	Población de padres emparejados

Salidas:

p_2	Pareja de p_1

Proceso:

1. Hacer p_2 = Candidato $(\alpha_{p_1}-1)$ en *Ch* según el criterio *cr*.

2. Eliminar p_2 de P_s.

3. Incluir p_2 en P_m.

Figura 2.8: Algoritmo 4. Elección de pareja Autoadaptativa (pasos 3 y 4 del Algoritmo1)

Al iniciar el algoritmo genético, el valor del índice de emparejamiento de cada individuo es inicializado uniformemente al azar con un número entero en el intervalo $[2,\gamma]$. Cuando dos padres de recombinan sus dos hijos heredan los índices de emparejamiento de los

padres. En cuanto a la mutación de índices de emparejamiento, se controla definiendo unos parámetros: $p_=$ es la probabilidad de que el índice de emparejamiento no varíe, p_+ es la probabilidad de que el índice de emparejamiento se incremente en 1, p_- es la probabilidad de que el índice de emparejamiento se decremente en 1 y quedaría una probabilidad $1-p_=-p_+-p_-$ de que el índice de emparejamiento cambie uniformemente al azar. Se ha tomado $p_==0.5$ y $p_+=p_-=0.24$, ya que para los problemas abordados en [Galán y Mengshoel (2010)] son los valores que proporcionan mejores resultados.

El trabajo descrito en [Galán y Mengshoel (2010)] no proporcionó resultados satisfactorios para el emparejamiento autoadaptativo en el 100 % de los problemas estudiados en su sección experimental. Concretamente, tal como se muestra en las figuras 7 (c) y 8 (c) del Anexo I, en problemas multimodales el emparejamiento autoadaptativo tenía un comportamiento no deseado a medida que se incrementaba el tamaño de emparejamiento.

Finalmente, el trabajo detallado en [Galán y Mengshoel (2010)] no fue aceptado para publicación en la revista “Information Sciencies” y surgió la necesidad de mejorarlo de algún modo. Precisamente el objetivo de este trabajo fin de máster ha sido mejorar el trabajo descrito en [Galán y Mengshoel (2010)]. Para ello hemos intentado seguir las recomendaciones de los revisores de la revista “Information Sciencies”: estudiar otras representaciones del cromosoma aparte de la binaria usada en [Galán y Mengshoel (2010)] e investigar nuevos métodos de control del parámetro índice de emparejamiento (aparte del autoadaptativo) que hagan al algoritmo genético más robusto en su comportamiento. Como resultado hemos ampliado el trabajo reseñado en [Galán y Mengshoel (2010)] y lo hemos enviado a la revista “Evolutionary Computation”, donde está en periodo de revisión. Las mejoras propuestas en este trabajo fin de máster al trabajo descrito en [Galán y Mengshoel (2010)] serán descritas en el capítulo 3.

CAPÍTULO 3 PROPUESTAS NOVEDOSAS PARA LA FASE DE EMPAREJAMIENTO DE UN ALGORITMO GENÉTICO

El problema tratado en [Galán y Mengshoel (2010)] consistió en el cálculo del óptimo global de funciones reales de varias variables mediante algoritmos genéticos. Para ello se utilizó una representación binaria del espacio de soluciones. En la representación binaria el genotipo es una cadena de bits. Así pues, en nuestro problema concreto de optimización de funciones de varias variables reales tenemos, para una función de la forma $\Re^n \to \Re$, un fenotipo de la forma $\langle x_1, ..., x_n \rangle$ y un genotipo de la forma $\langle b_1^1, ..., b_m^1, b_{1,}^2 ... , b_m^2, ..., b_{1,}^n ..., b_m^n \rangle$, donde $(b_1^j ... b_m^j)$ son los bits que codifican el valor de la variable x_j en forma binaria.

Al utilizar representación binaria el fenotipo y el genotipo no coinciden. La representación binaria presenta dos graves inconvenientes para el tipo de problema tratado. En primer lugar supone un coste computacional añadido, ya que se necesita obtener el fenotipo para calcular el valor de adaptación. Además, el recubrimiento del espacio de soluciones no es tan completo como en el caso de la representación real (quedan huecos). Para ver esto último consideremos que para los experimentos con la función Sphere se utilizaron 10 bits para representar cada variable real, que tomaba valores en el intervalo $[-10,10]$. Como $2^{10} = 1024$, podemos dividir el intervalo $[-10,10]$ en 1024 trozos con lo cual podemos llegar a representar números reales con una precisión de centésimas. Al ser la función Schwefel altamente multimodal hubo que utilizar un número mucho mayor de bits para representar cada variable. Se utilizaron 100 bits para representar cada variable real que, en este caso, tomaba valores en el intervalo $[-500,500]$. De esta forma se pudo llegar a una precisión del orden de 10^{-27} pero a cambio de aumentar el coste computacional de la conversión entre fenotipo y genotipo.

Como puede verse en las figuras 7 (c) y 8 (c) del Anexo I, la estrategia de emparejamiento autoadaptativo empeora en su comportamiento al aumentar el tamaño de

emparejamiento en el caso de funciones reales multimodales. Estos resultados obtenidos en [Galán y Mengshoel (2010)] animan a pensar que los resultados podrían mejorar si se utilizara representación real en vez de representación binaria. Al tratarse de funciones reales, la representación real permite una correspondencia completa entre fenotipo y genotipo, es decir, cada punto del espacio real (fenotipo) tiene un punto correspondiente exacto en el espacio de la representación (genotipo). Podemos decir que la representación real tiene una precisión mayor que la binaria (a no ser que trabajáramos, en la representación binaria, con un tamaño de cromosoma enorme).

Como se verá en la sección de evaluación experimental, el comportamiento de la estrategia de emparejamiento autoadaptativa mejora cuantitativamente con la nueva representación real respecto al método aleatorio clásico, pero cualitativamente los resultados para tamaños de emparejamiento altos siguen siendo malos (aunque menos malos que en el caso binario). Por ello se investigan en este trabajo fin de máster otros dos métodos de control del índice de emparejamiento a los que llamamos respectivamente "método de control espacial del índice de emparejamiento" y "método de control temporal del índice de emparejamiento".

3.1 Cromosoma con representación real

El problema objeto de nuestro estudio es la optimización de funciones reales de variable real n-dimensionales, formalmente $f:\Re^n \to \Re$. Las funciones pueden ser unimodales, que son las que tienen un solo valor óptimo y multimodales, que tienen muchos valores óptimos locales y un único valor óptimo global que es el que tratamos de encontrar.

Compararemos a continuación como se diseña el algoritmo genético que va a intentar resolver este problema, tanto usando representación real como usando representación binaria. En ambos casos tenemos una población que evoluciona generación tras generación. La población está formada por individuos.

a) Representación. En la representación binaria un individuo tiene un fenotipo que es un punto en el espacio de soluciones, o sea $\vec{x}=\langle x_1,...,x_n\rangle$. A cada fenotipo se le asocia de forma biunívoca un genotipo que es una cadena de bits de la forma $\vec{b}=\langle b_1^1,...,b_m^1,b_{1,}^2...,b_m^2,...,b_{1,}^n...,b_m^n\rangle$, donde $(b_1^j...b_m^j)$ son los bits que codifican el valor de la variable x_j en forma binaria. Necesitamos tener un par de métodos que realizan la

conversión entre fenotipo y genotipo, tal como $\vec{b}=fenotipoAGenotipo(\vec{x})$ y $\vec{x}=genotipoAFenotipo(\vec{b})$.

En la representación real cada gen del genotipo es un número real que representa el punto correspondiente del fenotipo (el punto en el espacio de soluciones). Por lo tanto fenotipo y genotipo coinciden y no es necesario realizar ninguna conversión entre fenotipo y genotipo.

b) Inicialización. En ambas representaciones cada individuo de la población se inicializa uniformemente al azar. En la representación binaria se obtiene una cadena binaria al azar de longitud igual al tamaño del cromosoma. En la representación real se obtiene para cada variable, que constituye un gen del cromosoma, un valor al azar dentro del rango de posibles valores para esa dimensión en el espacio de búsqueda.

c) Evaluación. La evalución consiste en obtener el valor de adaptación de cada individuo de la población. En la representación binaria, para cada individuo (caracterizado por su genotipo $\langle b_1^1,\ldots,b_m^1,b_{1,}^2\ldots,b_m^2,\ldots,b_{1,}^n\ldots,b_m^n\rangle$) hay que obtener en primer lugar el fenotipo correspondiente, aplicando el método de conversión:

$$\langle x_{1,}x_{2,}\ldots,x_n\rangle=genotipoAFenotipo(\langle b_1^1,\ldots,b_m^1,b_{1,}^2\ldots,b_m^2,\ldots,b_{1,}^n\ldots,b_m^n\rangle).$$

Una vez tenemos el fenotipo $\langle x_{1,}x_{2,}\ldots,x_n\rangle$ el valor de adaptación es el valor de la función a optimizar para ese punto, tomando el valor opuesto si se trata de minimizar el valor, o sea $\pm f(\langle x_{1,}x_{2,}\ldots,x_n\rangle)$.

En el caso de la representación real el proceso es más sencillo pues, como genotipo y fenotipo coinciden, basta con calcular el valor de la función en ese punto: $\pm f(\langle x_{1,}x_{2,}\ldots,x_n\rangle)$.

d) Selección de padres. En ambos casos la selección de padres se realiza por torneo. Se elige al azar de entre la población un número de individuos (normalmente dos individuos) de los cuales es seleccionado para pasar a la lista de posibles padres el de mejor valor de adaptación.

e) Emparejamiento. Esta es la fase en que se centra el presente trabajo. Tanto en el caso de la representación binaria como en al caso de la representación real se aplica lo dicho en la sección 2.4.

f) Recombinación. Para la representación binaria se usa cruce por un punto (*one-point crossover*) con probabilidad de cruce igual a uno. Esto significa que, para cada pareja de

padres, se escoge un número al azar dentro de intervalo $[1, l-1]$, siendo l la longitud del cromosoma. Entonces se obtienen los hijos intercambiando las colas de los cromosomas de los padres a partir de ese punto. Si el número obtenido al azar es r y los padres tienen los genotipos:

$$p^1 = \langle b^1_{1,} \dots, b^1_{r-1}, b^1_r, b^1_{r+1}, \dots, b^1_l \rangle$$

$$p^2 = \langle b^2_{1,} \dots, b^2_{r-1}, b^2_r, b^2_{r+1}, \dots, b^2_l \rangle$$

obtendremos dos hijos con los genotipos:

$$h^1 = \langle b^1_{1,} \dots, b^1_{r-1}, b^2_r, b^2_{r+1}, \dots, b^2_l \rangle$$

$$h^2 = \langle b^2_{1,} \dots, b^2_{r-1}, b^1_r, b^1_{r+1}, \dots, b^1_l \rangle .$$

En el caso de la representación real hemos utilizado recombinación discreta, que es el método de recombinación para representación real que más se asemeja al cruce por un punto de la representación binaria. Para ello se obtiene un número r al azar, tal que $1 \leqslant r \leqslant N$, siendo N el número de dimensiones que coincide con la longitud del cromosoma. Si los padres tienen los genotipos:

$$p^1 = \langle x^1_{1,} \dots, x^1_{r-1}, x^1_r, x^1_{r+1}, \dots, x^1_N \rangle$$

$$p^2 = \langle x^2_{1,} \dots, x^2_{r-1}, x^2_r, x^2_{r+1}, \dots, x^2_N \rangle$$

obtendremos dos hijos con los genotipos:

$$h^1 = \langle x^1_{1,} \dots, x^1_{r-1}, x^2_r, x^2_{r+1}, \dots, x^2_N \rangle$$

$$h^2 = \langle x^2_{1,} \dots, x^2_{r-1}, x^1_r, x^1_{r+1}, \dots, x^1_N \rangle .$$

g) Mutación. Para la representación binaria se utiliza el cambio de bit $(bit - flip)$, con probabilidad de mutación igual a la inversa de la longitud del cromosoma, para cada bit (gen) del cromosoma. Esto significa que, por término medio, debe cambiar un bit por cromosoma. Si tenemos un cromosoma de longitud l, sea $\langle b_{1,} \dots, b_{r-1}, b_r, b_{r+1}, \dots, b_l \rangle$, cada bit tiene una probabilidad $1/l$ de invertir su valor (pasar de 1 a 0 ó de 0 a 1).

No hay un equivalente claro de la mutación por cambio de bit para el caso de la representación real aunque se ha intentado buscar un operador que tuviera cierta similitud, al que hemos llamado “mutación gaussiana modificada”. En primer lugar si tenemos un cromosoma de longitud n (el número de dimensiones del espacio de búsqueda), sea $\langle x_1, \dots, x_n \rangle$, cada gen (variable) muta con una probabilidad $1/n$. Por lo tanto, por término medio, debe cambiar un gen por cromosoma. Si un gen ha de mutar, la mutación se produce añadiendo al valor del gen un valor obtenido al azar a partir de una distribución gaussiana con media igual a 0 y una desviación standard de 0.5

h) Selección de supervivientes. En ambos casos se ha utilizado reemplazo generacional con elitismo. La población original es reemplazada totalmente por la nueva generación, excepto el individuo con mejor valor de adaptación, que no es reemplazado sino que se incorpora a la nueva población sustituyendo a uno de los hijos.

La siguiente tabla resume las características del algoritmo genético para ambas representaciones

	Algoritmo genético con representación binaria	**Algoritmo genético con representación real**
Representación	Binaria	Real
Inicialización	Uniformemente al azar	Uniformemente al azar
Evaluación	1.Obtener el fenotipo a partir del genotipo 2. Calcular el valor de la función para el fenotipo	Calcular el valor de la función para el fenotipo (que coincide con el genotipo)
Selección de padres	Torneo	Torneo
Emparejamiento	Nuevo método de emparejamiento descrito en la sección 2.4	Nuevo método de emparejamiento descrito en la sección 2.4
Recombinación	Cruce por un punto (*one-point crossover*)	Recombinación discreta
Mutación	Cambio de bit (*bit-flip*)	Gaussiana modificada
Selección de supervivientes	Reemplazo generacional con elitismo	Reemplazo generacional con elitismo

Tabla 3.1: características del algoritmo genético utilizado en las representaciones binaria y real

3.2 El método espacial

La característica del que denominamos método de emparejamiento espacial consiste en considerar el espacio que ocupan los individuos de la población dividido en sectores, de forma que cada sector tiene asignado un valor de índice de emparejamiento diferente que quedará asociado a todos los individuos que ocupen ese sector. Vamos a proponer dos métodos de emparejamiento espacial.

3.2.1 Método espacial-1

Para realizar este tipo de emparejamiento distribuimos la población por sectores. Cada sector tiene asignado un índice de emparejamiento diferente. Por ejemplo, el primer sector tiene un índice de emparejamiento $\alpha = 2$ y a este sector pertenecen los j primeros

individuos dentro de la lista ordenada de individuos de la población. Al segundo sector, con índice de emparejamiento $\alpha=3$, pertenecen los individuos comprendidos entre el que ocupa la posición $j+1$ y el que ocupa la posición $2j$ y así sucesivamente para el resto de sectores. La distribución de sectores se realiza de forma lineal, tal como se describe a continuación (en el punto b). Para realizar el emparejamiento se utiliza el índice de emparejamiento asignado al sector al que pertenece el padre primario (el primer padre elegido para el emparejamiento, al que nos referimos en los algoritmos de la sección 2.4 como p_1), quedando el hijo resultante en el mismo sector en qué estaba el padre primario. Para poder implementar este método de emparejamiento hay que realizar algunos cambios con respecto al algoritmo original, quedando finalmente como se describe a continuación.

a) No realizamos selección de padres. Esto es para garantizar que se mantiene la distribución original de índices de emparejamiento, es decir, que a lo largo de todas las generaciones, se mantienen los mismos sectores, cada uno de ellos con su índice de emparejamiento y el mismo número de individuos. Además los individuos en cada sector son descendientes de los individuos que en generaciones anteriores han ocupado ese mismo sector.

b) Cada individuo de la población tiene asociado un valor de índice de emparejamiento (α). Este valor le corresponde a ese individuo por ocupar una determinada posición i en la lista de individuos de la población o, dicho de otra forma, por pertenecer a un determinado sector. Si tenemos una población de μ individuos y un tamaño de emparejamiento γ los diferentes valores de índice de emparejamiento (α) se asignan de la siguiente forma: hemos de distribuir linealmente valores de α, de tal forma que $2\leqslant\alpha\leqslant\gamma$, entre los μ individuos, por lo tanto $\alpha(i)=2+mi$, siendo m la pendiente $m=\frac{\gamma-1}{\mu}$.

c) La fase de emparejamiento quedaría reformulada de la siguiente forma:

1. Se va eligiendo sucesivamente cada individuo de la población. Llamaremos p_1 al individuo actualmente elegido para actuar como padre “primario”. El padre primario tiene la característica de que aporta el valor de α asociado a su posición y que el hijo generado ocupará esa misma posición (y por lo tanto mantendrá ese mismo valor de α).

2. Elegimos al azar $\gamma - 1$ individuos de la población total, excluido p_1. Estos individuos forman el conjunto Ch y son los individuos candidatos a emparejarse con p_1.

3. La pareja de p_1 se extrae de Ch con el mismo algoritmo descrito en la sección 2.4, es decir: se ordenan los $\alpha - 1$ mejores candidatos en Ch según el criterio cr. Llamamos a esta lista ordenada $B_\alpha(p_1) = [b_2, \ldots, b_\alpha]$. La pareja para p_1 es $p_2 = b_\alpha$ (el último de la lista).

4. La lista de parejas Pm será una lista de 2μ individuos: cada par de individuos sucesivos son el padre primario y su correspondiente pareja.

d) En la fase de recombinación, con cada par de elementos sucesivos de Pm (que son las parejas p_1 y p_2 que han ido entrando en la lista en la fase de emparejamiento) generaremos un hijo, que ocupará en la nueva lista de individuos de la población la misma posición que ocupaba el padre "primario" (p_1), de la siguiente forma: Con p_1 y p_2 se generan dos hijos de los que se descarta el de peor valor de adaptación. El de mejor valor de adaptación sustituye al padre "primario" en su misma posición (con lo que mantendrá el mismo valor de α) únicamente si su valor de adaptación es mejor o igual que el del padre (en caso contrario se mantiene el padre en esta posición).

e) La fase de mutación se realiza como en la programación original.

Hemos de quitar la selección de padres para garantizar que cada índice de emparejamiento inicial se mantiene a lo largo de todas las generaciones, heredándolo sucesivamente los hijos descendientes del padre inicial siguiendo una línea ininterrunpida de padres primarios. Al quitar la selección de padres estamos preservando las minorias, lo cual podría dar buenos resultados a largo plazo, pero hay que hacer notar que se trata del único algoritmo, de los estudiados, que no realiza selección de padres.

3.2.2 El método espacial-2

En la versión anterior del método espacial, como ya mencionamos, se ha tenido que eliminar la selección de padres. Ahora vamos a proponer otro método, al que también llamamos espacial, ya que mantiene a lo largo de las generaciones unos valores del índice de emparejamiento asociados a la posición que ocupa cada individuo en la población, pero

con la misma estructura algorítmica de los otros métodos que hemos estudiado (y, por lo tanto, con selección de padres).

A cada posición de un individuo dentro de la población se le asigna un valor de índice de emparejamiento. Este valor se podría asignar uniformemente al azar, pero se ha preferido asignarlo mediante una distribución lineal porque la distribución será similar y permite poder realizar, en un futuro, variaciones cambiando la pendiente o el tipo de distribución. De esta forma el individuo que ocupa la posición $\mathrm{i}-esima$ en la lista de individuos de la población tiene automáticamente asignado el índice de emparejamiento $\alpha(\mathrm{i})=2+m\mathrm{i}$, siendo $m=\frac{\gamma-1}{\mu}$. Durante el proceso de emparejamiento el valor de índice de emparejamiento utilizado por el algoritmo es el correspondiente a la posición que ocupa el padre primario (p_1 en la figura 2.5). Los hijos obtenidos tendrán el índice de emparejamiento que les corresponda por la posición ordinal que ocupen dentro de la población.

3.3 El método temporal

El método de emparejamiento temporal consiste en hacer que el índice de emparejamiento (único para toda la población) vaya variando con el tiempo, es decir con el paso de las generaciones. Pretendemos que haya más exploración en las primeras generaciones, con un índice de emparejamiento alto, y que éste vaya disminuyendo a lo largo de las generaciones, de forma que irá aumentando la explotación y disminuyendo la exploración.

Para ello, el índice de emparejamiento de la generación $\mathrm{i}-ésima$ se obtiene multiplicando por una constante de decaimiento el índice de emparejamiento de la generación anterior, o sea $\alpha_i=K\cdot\alpha_{i-1}$, siendo K la constante de decaimiento, $0\leqslant K\leqslant 1$. De esta forma el índice de emparejamiento disminuye exponencialmente. Para la primera generación hacemos el índice de emparejamiento igual al tamaño de emparejamiento, $\alpha_1=\gamma$, que corresponde a la máxima exploración. Como los índices de emparejamiento son números enteros la implementación se ha realizado de la siguiente forma. Hemos usado una variable real auxiliar, α', tal que $\alpha'_1=\gamma$ y $\alpha'_{\mathrm{i}}=K\cdot\alpha'_{\mathrm{i}-1}$ y obtenemos el índice de emparejamiento de cada generación quedándonos con la parte entera redondeada de la variable auxiliar para la misma generación: $\alpha_{\mathrm{i}}=\mathrm{int}(round(\alpha'_{\mathrm{i}}))$. Como

$\forall i, 2 \leqslant \alpha_i \leqslant \gamma$, tomaremos $\alpha_i = 2$ para cualquier resultado de $\alpha_i = \text{int}(round(\alpha'_i))$ que sea igual o menor que 2.

Si $K = 0$ estamos en el caso que hemos denominado *Best – First*. Si $K = 1$ estamos en el caso que hemos denominado *Best – Last*.

CAPÍTULO 4 SOFTWARE DESARROLLADO

Para la realización de la parte experimental del presente trabajo de fin de máster se ha desarrollado un programa informático que vamos a describir en este capítulo. El programa se ha desarrollado *ex profeso* para la realización del presente trabajo sin utilizar, de forma total o parcial, otros programas o librerias. El programa se ha desarrollado en el lenguaje de programación Python. El lenguaje de programación Python se caracteriza por producir programas compactos y fácilmente legibles, es sencillo de usar, dispone de tipos de datos de alto nivel, como listas y diccionarios, muy fáciles de manejar. Pero, sobre todo, tiene dos características que lo hacen especialmente interesante para implementar algoritmos evolutivos. Por una parte permite desarrollar una programación orientada a objetos, de forma que podemos asociar cada fase del algoritmo a una clase de Python, resultando un código limpio y claro. Por otra parte, es un lenguaje interpretado que no necesita una compilación previa ni siquiera a un código intermedio (lo genera el propio intérprete en tiempo de ejecución). Esto nos ha permitido que los ficheros de parámetros sean módulos Python, lo que hace que la información sea más fácilmente accesible por el programa y que el ciclo cambio – ejecución sea muy ágil, facilitando el trabajo experimental.

Aparte de los ficheros que contienen las clases, existen dos ficheros de parámetros que permiten fijar las condiciones concretas de cada experimento. El fichero parametros_funciones.py permite definir las características de las funciones a optimizar, básicamente la definición de la propia función, el número de dimensiones del dominio y el rango de valores permitidos para cada una de las dimensiones del dominio. El fichero parametros.py permite definir los parámetros que se pueden ajustar para la ejecución del algoritmo genético. Los parámetros serán descritos cuando se necesite durante este capítulo.

En la sección 4.1 describimos cada una de las clases: Test, Algoritmo, Funcion, Individuo, Poblacion, Mating, Recombinacion y Mutacion. En la sección 4.2 explicamos el funcionamiento en ejecución del software desarrollado, cuando es utilizado para realizar un conjunto de experimentos.

4.1 Descripción de las clases

Clase Test

Esta clase se encarga de hacer que el algoritmo genético se ejecute un número determinado de veces. Este número de veces está definido en el fichero de parámetros (`NUM_RUNS`). Guarda información relevante en cada ejecución, con la que calcula los valores medios del mejor valor de adaptación en cada generación, una vez terminadas todas las ejecuciones. Finalmente, genera un fichero en disco con la información obtenida en el experimento y los parámetros bajo los que se ha realizado el experimento.

En las tablas 4.1 y 4.2 se describen respectivamente las propiedades y métodos de esta clase.

Propiedad	**Tipo de datos**	**Descripción**
fitness	Lista de números reales	Media, obtenida sobre todas las ejecuciones del algoritmo, del mejor valor de adaptación en cada generación. En problemas de maximización el valor de y coincide con el valor de adaptación, $fitness = y$. En problemas de minimización hemos tomado como valor de adaptación el valor de la función cambiado de signo, $fitness = -y$
y	Lista de números reales	Media, obtenida sobre todas las ejecuciones del algoritmo, del mejor valor de la función en cada generación.

Tabla 4.1: Propiedades de la clase Test

Método	**Entradas**	**Retorno**	**Descripción**
exe			Repite la ejecución del algoritmo definido en la clase Algoritmo el número de veces que se indica en `NUM_RUNS` (fichero de parámetros). Al terminar guarda la información en un fichero en disco
calculaMedias	medias nuevos N	Nuevo valor para medias	*medias* es la lista de valores medios hasta la ejecución anterior (*N*-1) y *nuevos* la lista de nuevos valores obtenidos en la ejecución *N*. Devuelve la lista de valores medios actualizada con los nuevos valores

Tabla 4.2: Métodos de la clase Test

Clase Algoritmo

Esta clase lleva a cabo la ejecución del algoritmo genético, tal como se describió en la figura 2.4. Básicamente hace lo siguiente: en primer lugar, una inicialización de la población y también del módulo generador de números aleatorios, para garantizar que cada ejecución del algoritmo es aleatoriamente independiente de las demás. A continuación, repite las diferentes fases hasta que se cumple la condición de terminación definida en el fichero de parámetros (`TERMINATION_CONDITION_TYPE`). Las fases que se repiten son: selección de padres, emparejamiento, recombinación, mutación y selección de supervivientes.

En las tablas 4.3 y 4.4 se describen respectivamente las propiedades y métodos de esta clase.

Propiedad	**Tipo de datos**	**Descripción**
fitness	Lista de números reales	Para almacenar el mejor valor de adaptación en cada generación para la ejecución actual del algoritmo
y	Lista de números reales	Para guardar el mejor valor de la función en cada generación para a ejecución actual del algoritmo

Tabla 4.3: Propiedades de la clase Algoritmo

Método	**Entradas**	**Retorno**	**Descripción**
exe			Obtiene objetos que necesita para acceder a las clases Funcion, Poblacion, Recombinacion, Mutacion y Mating, que se describen en esta misma sección, y lleva a cabo, de forma iterativa, la ejecución del algoritmo genético, hasta que se cumpla la condición de terminación.
guarda	pobl		Guarda los mejores valores de adaptación y de la función, para la generación actual, de la población que se le paso mediante el objeto pobl. Los valores se guardan en las propiedades *fitness* e *y* respectivamente

Tabla 4.4: Métodos de la clase Algoritmo

Clase Funcion

La clase Funcion encapsula las características particulares de las funciones reales de varias variables objeto de nuestro estudio.

En el fichero parametros_funciones.py se define cada función mediante una clase. Hay, por lo tanto, una clase Sphere, una clase Schwefel y una clase Rastrigin. Estas clases tiene la propiedad `RANGO_XI`, que es una lista de tuplas, con los valores mínimo y máximo permitidos para cada una de las dimensiones del dominio de la función. El número de elementos de la lista coincide con el número de dimensiones del dominio. También tienen un método, en el que se define la propia función, es decir, se le pasa como entrada un vector que es un punto del dominio de la función y el método devuelve el valor de la función para ese punto.

En el fichero de parámetros se especifica, mediante el parámetro `FITNESS_TYPE`, cuál es la función que se va a optimizar. La clase Funcion es una clase derivada (subclase) de la clase Sphere, Schwefel o Rastrigin, según el valor del parámetro `FITNESS_TYPE`.

En las tablas 4.5 y 4.6 se describen las propiedades y métodos de esta clase.

Propiedad	**Tipo de datos**	**Descripción**
N	Número entero	Aquí se guarda el número de dimensiones del dominio de la función. N es igual al número de elementos de la lista RANGO_XI
TIPO_OPT	Número entero	Indica si se trata de una función que hay que minimizar o maximizar.
RANGO_XI	Lista de tuplas de números reales	Cada tupla contiene los valores mínimo y máximo permitidos para cada dimensión del dominio de la función.

Tabla 4.5: Propiedades de la clase Funcion

Método	Entradas	Retorno	Descripción
fun	$(x_1, \ldots, x_N)$	y	Este método lo hereda la clase Funcion de la superclase que se ha definido en el parámetro `FITNESS_TYPE`. Al método se le pasa como entrada un vector de números reales y devuelve el valor de la función para ese punto
fitness	$(x_1, \ldots, x_N)$	(y,fitness)	Al método se le pasa como entrada un vector, $(x_1, \ldots, x_N)$, que es un punto en el espacio de soluciones. El método devuelve una tupla formada por el valor de la función correspondiente a ese punto y el valor de adaptación correspondiente al mismo punto. El valor de la función se obtiene mediante el método *fun*. El valor de adaptación es igual al valor de la función para problemas de maximización. Para problemas de minimización el valor de adaptación es igual al valor de la función cambiado de signo
calculaFitness	individuos		A este método se le pasa como entrada una lista de elementos de la clase *Individuo* y el método, llamando a su vez al método fitness, actualiza los elementos de la lista con el valor de la función y el valor de adaptación para cada elemento de la lista

Tabla 4.6: Métodos de la clase Funcion

Clase Individuo

Esta clase contiene la información que caracteriza a cada individuo de la población. Cada individuo es una instancia de esta clase. En las tablas 4.7 y 4.8 se describen respectivamente las propiedades y métodos de esta clase.

Propiedad	Tipo de datos	Descripción
geno	Lista de números reales	Es el genotipo del individuo. En la representación real coincide con el fenotipo, por lo que es también un punto en el espacio de soluciones
fit	Objeto de la clase Funcion	Es una referencia al objeto Funcion que representa a la función que estamos optimizando. Se necesita para obtener el valor de adaptación del individuo
y	Número real	Es el valor que toma la función que estamos optimizando en el punto del espacio de soluciones correspondiente al individuo
fitness	Número real	Es el valor de adaptación del individuo. Numéricamente coincide con y o con $-y$, según se trate de un problema de maximización o minimización

Tabla 4.7: Propiedades de la clase Individuo

Método	Entradas	Retorno	Descripción
genAlAzar	rango	Número real	Al método se le pasa como entrada una tupla con los valores mínimo y máximo de la variable real correspondiente y devuelve un número real al azar entre esos valores
inicializa			Este método inicializa cada gen del genotipo del individuo (propiedad geno) utilizando el método genAlAzar

Tabla 4.8: Métodos de la clase Individuo

Clase Poblacion

La clase Poblacion es la clase central del programa. Por una parte, contiene la población de individuos que evoluciona y el mejor individuo (el individuo con mayor valor de adaptación) de la población representa la solución buscada. Por otra parte, todos los procesos del algoritmo genético (selección, recombinación, mutación, etc) parten de métodos de esta clase, que utilizan objetos de otras clases (que son las que realmente implementan cada proceso) para ejecutarlos.

En las tablas 4.9 y 4.10 se describen las propiedades y métodos de esta clase.

Propiedad	Tipo de datos	Descripción
fit	Objeto de la clase Funcion	Es una referencia al objeto Funcion que representa la función que estamos optimizando. Se necesita para obtener el valor de adaptación del individuo
individuos	Lista de objetos de la clase Individuo	Contiene todos los individuos de la población
padres	Lista de objetos de la clase Individuo	Es una lista con los individuosseleccionados para ser los padres de la siguiente generación
hijos	Lista de objetos de la clase Individuo	Es una lista que contiene los hijos recién obtenidos
mejorindi	Objeto de tipo Individuo	Contiene una copia del mejor individuo de la población
numgeneraciones	Número entero	Contador de generaciones

Tabla 4.9: Propiedades de la clase Poblacion

Método	Entradas	Retorno	Descripción
selecPadres			De entre los individuos de la población (lista *individuos*), lleva a cabo la selección de los que pueden ser padres de la siguiente generación y guarda los individuos seleccionados en la lista *padres.* La forma de realizar la selección queda determinada por algunos parámetros que se especifican en el fichero de parámetros: `PARENT_SELECTION_TYPE`, `GENERATIONAL_GAP`, `TOURNAMENT_SIZE`
matea	Objeto de la clase Mating		Accediendo al método *mate* de la clase *Mating* lleva a cabo, partiendo de la lista *padres*, el emparejamiento de padres, dejando, en la misma lista *padres*, los padres ya emparejados (el primer padre se recombinará con el segundo, el tercero con el cuarto, etc)
recombina	Objeto de la clase Recombinacion		Obtiene dos nuevos objetos de la clase Individuo que son dos nuevos hijos. Después obtiene los genotipos para cada uno de los dos nuevos hijos por recombinación de dos padres. Los padres se van obteniendo de la lista de padres de dos en dos (están emparejados por el proceso anterior de emparejamiento,

			método *mate*). Para llevar a cabo la recombinación accede al método recombina de la clase Recombinación. El proceso se repite hasta que tengamos el número de hijos especificado en el parámetro `GENERATIONAL_GAP` (fichero de parámetros)
muta	Objeto de la clase Mutacion		Para cada individuo contenido en la lista hijos se lleva a cabo la mutación del genotipo. Para ello se accede al método muta de la clase Mutacion
reemplaza			Este método realiza un reemplazo generacional con elitismo. Para ello, en primer lugar, calcula el valor de adaptación de los nuevos hijos. Toda la población actual (lista *individuos*) es sustituida por la nueva (lista *hijos*), excepto el individuo con mejor valor de adaptación que permanece en la población (elitismo)
finProceso			Este método se usa para determinar cuándo hay que terminar el proceso. Las condiciones de terminación se especifican en el fichero de parámetros (`TERMINATION_CONDITION_TYPE` y `GENERATIONS_NUMBER`)

Tabla 4.10: Métodos de la clase Poblacion

Clase Mating

La clase Mating es la que implementa la fase de emparejamiento, en la que se centra el presente trabajo de fin de máster. De ello se encarga el método mate. Los demás métodos de la clase son métodos que utiliza el método mate. En la tabla 4.11 se describen los métodos de esta clase. La clase Mating no tiene propiedades.

Método	Entradas	Retorno	Descripción
mate	Lista de objetos de la clase Individuo	Lista de objetos de la clase Individuo	Este método recibe como entrada una lista de individuos, concretamente es la lista de individuos seleccionados para ser padres en la fase de selección de padres (es el conjunto que en las figuras 2.5 a 2.8 se identifica como P_s). El método implementa el algoritmo de emparejamiento descrito en las figuras 2.5 a 2.8. En el fichero de parámetros están los parámetros que permiten controlar el proceso: `MATING_SIZE`, `MATING_INDEX` y `MATING_CR`. El método retorna una lista de individuos (padres) emparejados, de forma que, en la posterior fase de recombinación, se recombinarán el primero con el segundo, el tercero con el cuarto, y así sucesivamente
ordenaPorFitness	Lista de objetos de la clase Individuo	Lista de objetos de la clase Individuo	Ordena los individuos de la lista de entrada según su valor de adaptación y devuelve la lista ordenada
mejorFitness	Lista de objetos de la clase Individuo	Objeto de la clase Individuo	Obtiene, de entre los individuos de la lista de entrada, el que tiene mejor valor de adaptación y lo devuelve.
iesimoCr	- Lista de objetos de la clase Individuo - Número entero que indica el ránking del individuo buscado (iésimo mejor o peor) - Booleano que indica si buscamos el iésimo mejor o peor - (Opcional) Objeto de la clase Individuo, necesario si el criterio de ordenación es semejanza	Objeto de la clase Individuo	Este método obtiene el iésimo mejor o peor candidato de la lista de individuos, según el criterio de preferencias de emparejamiento (parámetro `MATING_CR` del fichero de parámetros)

Tabla 4.11: Métodos de la clase Mating

Clase Recombinacion

Esta clase implementa la fase de recombinación. Se han programado dos tipos de recombinación, que se pueden seleccionar en el fichero de parámetros, mediante el parámetro CROSSOVER_TYPE. Si partimos de dos genotipos $g_1=\langle x^1_{1,}\ldots,x^1_N\rangle$ y $g_2=\langle x^2_{1,}\ldots,x^2_N\rangle$, en el primer tipo de recombinación, denominado *discrete recombination*, obtenemos un número entero al azar $i\in\{1,\ldots,N\}$ y los nuevos genotipos serían $g'_1=\langle x^1_{1,},\ldots x^1_i,x^2_{i+1},\ldots,x^2_N\rangle$ y $g'_2=\langle x^2_{1,},\ldots x^2_i,x^1_{i+1},\ldots,x^1_N\rangle$. El segundo tipo de recombinación que se ha programado se denomina *whole arithmetic recombination* [Eiben y Smith (2003)], en el que cada gen del nuevo genotipo mezcla los genes correspondientes de los padres. En el fichero de parámetros hay un parámetro, llamado ALFA, que representa la proporción de la mezcla. Partiendo de los mismos genotipos $g_1=\langle x^1_{1,}\ldots,x^1_N\rangle$ y $g_2=\langle x^2_{1,}\ldots,x^2_N\rangle$, los nuevos genotipos $g'_1=\langle x'^1_{1,}\ldots,x'^1_N\rangle$ y $g'_2=\langle x'^2_{1,}\ldots,x'^2_N\rangle$ se obtienen de la siguiente forma: $x'^1_i=\alpha\cdot x^1_i+(1-\alpha)\cdot x^2_i$ y $x'^2_i=\alpha\cdot x^2_i+(1-\alpha)\cdot x^1_i$. Se realizaron unos experimentos preliminares para determinar qué tipo de recombinación utilizar. Como resultado de estas pruebas decidimos realizar los experimentos con *discrete recombination*.

En la tabla 4.12 se describen los métodos de esta clase. La clase Recombinacion no tiene propiedades.

Método	**Entradas**	**Retorno**	**Descripción**
recombina	Dos objetos de la clase individuo	Tupla formada por dos genotipos (un genotipo es una lista de números reales)	Este método realiza la recombinación de los genotipos de los dos individuos que recibe como entrada y obtiene, por recombinación, dos nuevos genotipos

Tabla 4.12: Métodos de la clase Recombinacion

Clase Mutacion

Esta clase implementa la fase de mutación. Se han programado tres tipos de mutación, que se pueden seleccionar en el fichero de parámetros, mediante el parámetro MUTATION_TYPE. A continuación describimos estos tres tipos de mutación, partiendo de un individuo, cuyo genotipo es $g=\langle x_{1,}\ldots,x_N\rangle$.

En el primer tipo, denominado *mutación uniforme* [Eiben y Smith (2003)], cada gen tiene una probabilidad $1/N$ de mutar. De media tendremos, por lo tanto, un gen mutado por cada individuo. Si un gen es seleccionado para mutar la mutación se produce eligiendo un nuevo valor para ese gen, al azar (un número real aleatoriamente al azar, dentro del intervalo de números reales permitido para ese gen).

El segundo tipo es la *mutación gaussiana* [Eiben y Smith (2003)]. En este caso cada gen, x_i, es sustituido por $x_i + \Delta x_i$, siendo Δx_i un valor obtenido al azar para una distribución gaussiana con media igual a 0 y cuya desviación estándar se indica en el fichero de parámetros, en el parámetro `SIGMA`. En caso de que el resultado de la expresión $x_i + \Delta x_i$ esté fuera del rango de valores permitidos para el gen (parámetro `RANGO_XI` del fichero parametros_funciones), el gen se queda sin mutar.

Al tercer tipo de mutación lo hemos denominado *mutación gaussiana modificada.* En este caso, como en la mutación uniforme, hay una probabilidad $1/N$ de mutar cada gen, de forma que es muy probable que el gen no mute. Pero, en caso de que el gen deba mutar, muta con mutación gaussiana, con otra pequeña diferencia: en caso de que el resultado de la expresión $x_i + \Delta x_i$ esté fuera del rango de valores permitidos para el gen (parámetro `RANGO_XI` del fichero parametros_funciones), si $x_i + \Delta x_i$ es mayor que el valor máximo permitido, el gen toma como valor el valor máximo permitido; y si $x_i + \Delta x_i$ es menor que el valor mínimo permitido, el gen toma como valor el valor mínimo permitido.

Como en el caso de la recombinación, realizamos experimentos preliminares para decidir qué tipo de mutación utilizar. Como resultado de esas pruebas decidimos realizar los experimentos con *la mutación gaussiana modificada,* con desviación estándar igual a 0.5, por ser la que obtenía mejores resultados.

En la tabla 4.13 se describen los métodos de esta clase. La clase Mutacion no tiene propiedades.

Método	**Entradas**	**Retorno**	**Descripción**
muta	Objetos de la clase individuo	Lista de números reales (el genotipo mutado)	El método realiza la mutación del genotipo del individuo que recibe como entrada y devuelve el genotipo mutado.

Tabla 4.13: Métodos de la clase Mutacion

4.2 Descripción de la ejecución del algoritmo genético

El proceso de ejecución del algoritmo genético se inicia obteniendo una instancia de la clase Test e invocando el método exe, tal como se indica en la figura 4.1.

```
t = Test()
t.exe()
```

Figura 4.1: Inicio del proceso, invocando el método exe de la clase Test

De esta forma, el método exe de la clase Test se encarga de ejecutar el algoritmo un número determinado de veces, obteniendo las medias de los resultados de cada ejecución, tal como se indica en la figura 4.2.

```
REPITE HASTA (i > parametros.NUM_RUNS)
      ag = Algoritmo()
      ag.exe()
      CALCULAR valores medios;
      i = i + 1
FIN DE REPITE
GUARDAR valores medios;
```

Figura 4.2: El método exe de la clase Test ejecuta repetidamente el algoritmo y calcula los valores medios de los resultados obtenidos

El método exe de la clase Algoritmo es el que realiza la ejecución del algoritmo genético. Para ello obtiene las instancias que necesita de todas las clases y repite las fases del algoritmo hasta que se cumple la condición de terminación. En la figura 4.3 se muestra el esquema del método exe de la clase Algoritmo.

```
INICIALIZAR el generador de números aleatorios
fit = Funcion()
recom = Recombinación()
muta = Mutacion()
pobl = Poblacion(fit)
mate = Mating()
REPITE HASTA (pobl.finProceso())
      guarda(pobl)
      pobl.selecPadres()
      pobl.matea(mate)
      pobl.recombina(recom)
      pobl.muta(muta)
      pobl.reemplaza()
FIN DE REPITE
```

Figura 4.3: El método exe de la clase Algoritmo. Inicializa el generador de números aleatorios y obtiene instancias de las clases necesarias. Después repite las fases del algoritmo genético hasta que se cumple la condición de terminación. La clase Poblacion contiene listas de instancias de la clase Individuo. Hay que observar que a la clase Poblacion hay que pasarle una instancia de la clase Funcion para que se puedan calcular los valores de adaptación de los individuos

La clase población contiene la lista de individuos (miembros de la clase Individuo) que forman la población, así como otras listas auxiliares de individuos (padres, hijos…), necesarias en diferentes fases del algoritmo genético.

Al terminar el proceso, el programa genera un fichero que contiene los resultados del experimento. Estos son propiedades de la clase Test: la media, obtenida sobre todas las ejecuciones del algoritmo, del mejor valor de la función en cada generación (propiedad *y*) y la media, obtenida sobre todas las ejecuciones del algoritmo, del mejor valor de adaptación en cada generación (propiedad *fitness*). También se guarda en el mismo fichero una copia del fichero de parámetros, para que quede constancia de las condiciones en las que se realizó el experimento.

CAPÍTULO 5 EVALUACIÓN EXPERIMENTAL

El problema objeto de estudio es la optimización de funciones reales de varias variables. Formalmente, dada una función $f:\Re^n \to \Re$, la optimización consiste en encontrar un $x^* \in \Re^n$ tal que $f(x^*) \geq f(x) \forall x \neq x^*$ (caso de maximización) o bien encontrar un $x^* \in \Re^n$ tal que $f(x^*) \leq f(x) \forall x \neq x^*$ (caso de minimización). Ese valor óptimo sería el óptimo global. La funciones multimodales (como las funciones Schwefel y Rastrigin) se caracterizan porque existe un óptimo global pero muchos óptimos locales. Esto introduce una dificultad añadida al problema de optimización, porque el algoritmo de búsqueda puede converger a uno de esos óptimos locales sin encontrar el óptimo global. En el caso de las funciones unimodales (como la función Sphere) no hay óptimos locales.

Lo que se pretende estudiar es la influencia de distintos métodos de emparejamienro de individuos en la eficacia del algoritmo genético a la hora de encontrar el óptimo global.

Para llevar a cabo los experimentos de optimización se ha utilizado un algoritmo genético en el que se ha tenido especial cuidado en la implementación de diferentes métodos de emparejamiento. Se ha utilizado representación real, selección de padres por torneo con tamaño de torneo igual a dos, recombinación discreta, mutación gaussiana con una probabilidad $1/n$ de mutar un gen (siendo n el número de genes) y desviación standard igual a 0.5, así como selección de supervivientes generacional con elitismo. Para cada ejecución del algoritmo se ha inicializado el generador de números aleatorios de forma independiente.

5.1 Experimentos variando el valor del tamaño de emparejamiento

En esta sección se comparan experimentalmente los métodos *Best-First* ($\alpha = 2$), *Best-Last* ($\alpha = \gamma$), *Autoadaptativo* y *Random*. En principio es de esperar una mayor explotación

con *Best-First* y una mayor exploración con *Best-Last*. Para el caso *Autoadaptativo* cabe esperar una combinación de exploración y explotación adaptada al problema concreto de optimización planteado.

Los experimentos realizados nos permitirán comparar:

- Funciones unimodales y multimodales. Para ello se realizan experimentos con la función Sphere (unimodal) y la función Schwefel (multimodal).

- Preferencia para el emparejamiento basada en el valor de adaptación y basada en semejanza. Para ello se repiten los experimentos con los dos criterios.

- Métodos de emparejamiento tradicional (*Random*) y avanzados (*Best-First*, *Best-Last* y *Autoadaptativo*).

- Varios valores de tamaño de emparejamiento. Se han realizado experimentos para $\gamma \in \{3,5,10,20,30\}$.

5.1.1 Experimentos con Best-First, Best-Last y Autoadaptativo para la función Sphere

La función Sphere se define de la siguiente forma:

$$f_{\text{Sphere}}(x) = \sum_{i=1}^{n} x_i^2 \ .$$

Esta función tiene un mínimo en el punto $x = (0,\ldots,0)$ cuyo valor es 0. Los experimentos se han realizado con $n = 20$, $-10 \le x_i \le 10 \ \forall i \in \{1,\ldots,20\}$. Además se usó un tamaño de población de 100 individuos.

Las figuras 5.1 (a, b y c) representan la evolución, generación a generación, de la media del mejor (mínimo) valor de la función, utilizando varias estrategias de emparejamiento, con $cr = fitness$. Para cada experimento se han realizado cien ejecuciones del algoritmo genético. En las tres figuras también se ha presentado el resultado para emparejamiento al azar (*Random*), como referencia. Junto al caso *Random*, se representan los casos *Best-First* (figura 5.1 (a)), *Best-Last* (figura 5.1 (b)) y

Autoadaptativo (figura 5.1 (c)). En los tres casos se realizaron experimentos para valores de tamaño de emparejamiento $\gamma = \{3,5,10,20,30\}$.

En figura 5.1 (a) se observa que *Best-First* obtiene mejores resultados que *Random* para cualquier valor de tamaño de emparejamiento, aumentando la mejora al aumentar el valor de tamaño de emparejamiento. Este comportamiento también se observa para el método *Autoadaptativo* (figura 5.1 (c)) pero en este caso la mejora sobre el caso *Random* se manifiesta a partir de $\gamma = 10$. En la figura 5.1 (b) se pone de manifiesto que los resultados para *Best-Last* son peores que para *Random*, empeorando al aumentar el valor del tamaño de emparejamiento.

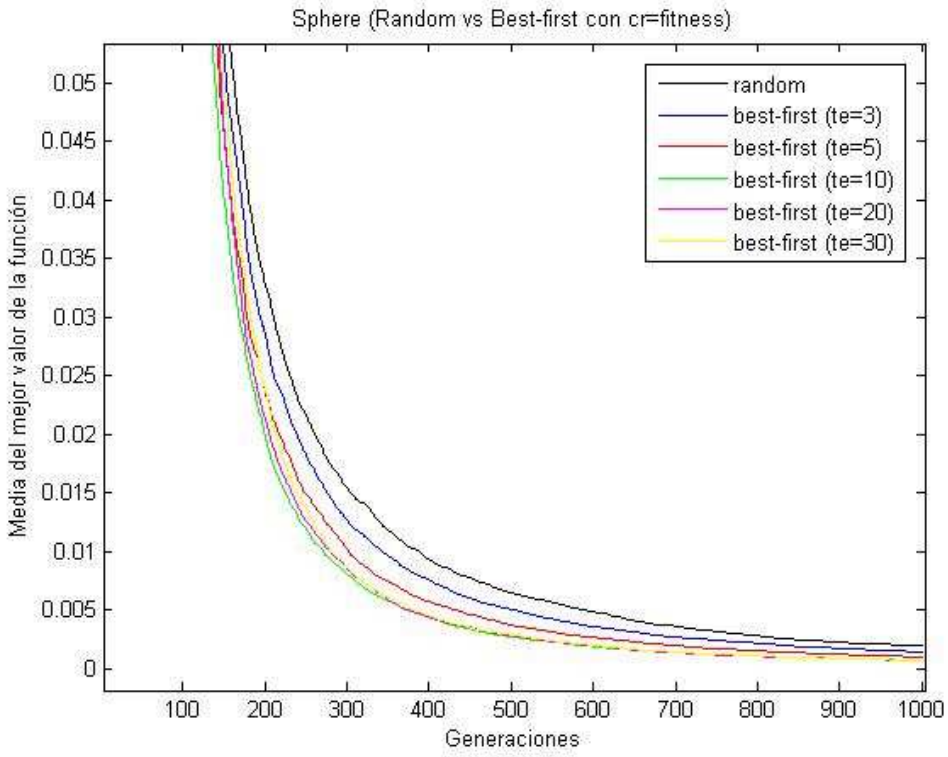

Figura 5.1 (a): Media del mejor valor de la función para la minimización de la función Sphere para el método de emparejamiento Best-First y criterio *cr = fitness* (te = tamaño de emparejamiento)

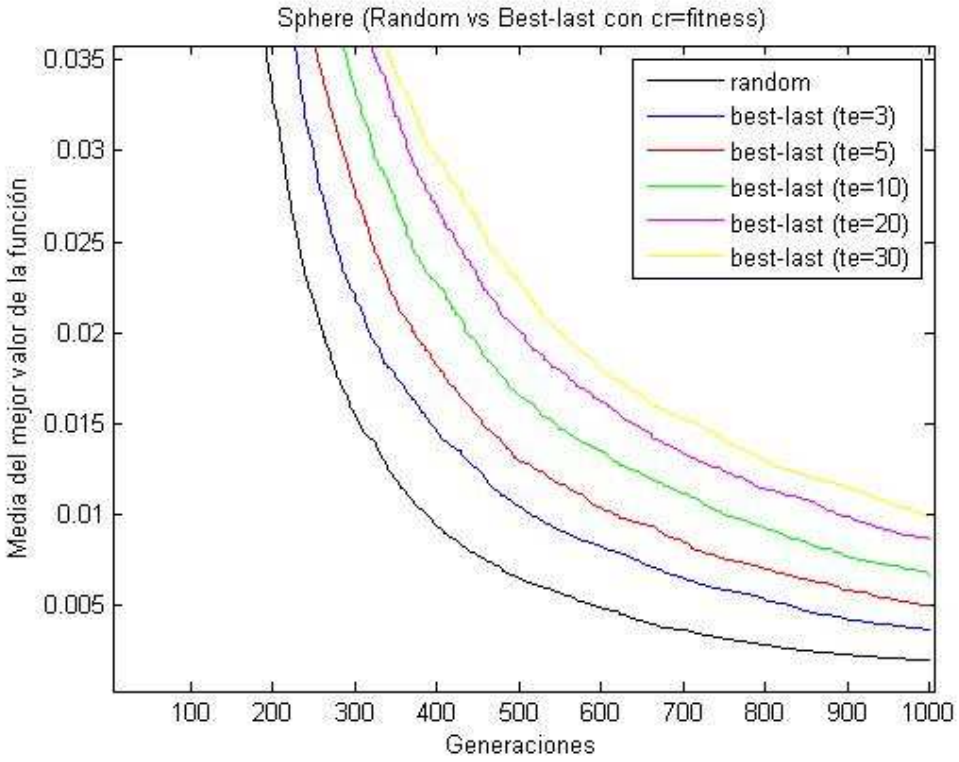

Figura 5.1 (b): Media del mejor valor de la función para la minimización de la función Sphere para el método de emparejemiento Best-Last y criterio $cr = fitness$ (te = tamaño de emparejamiento)

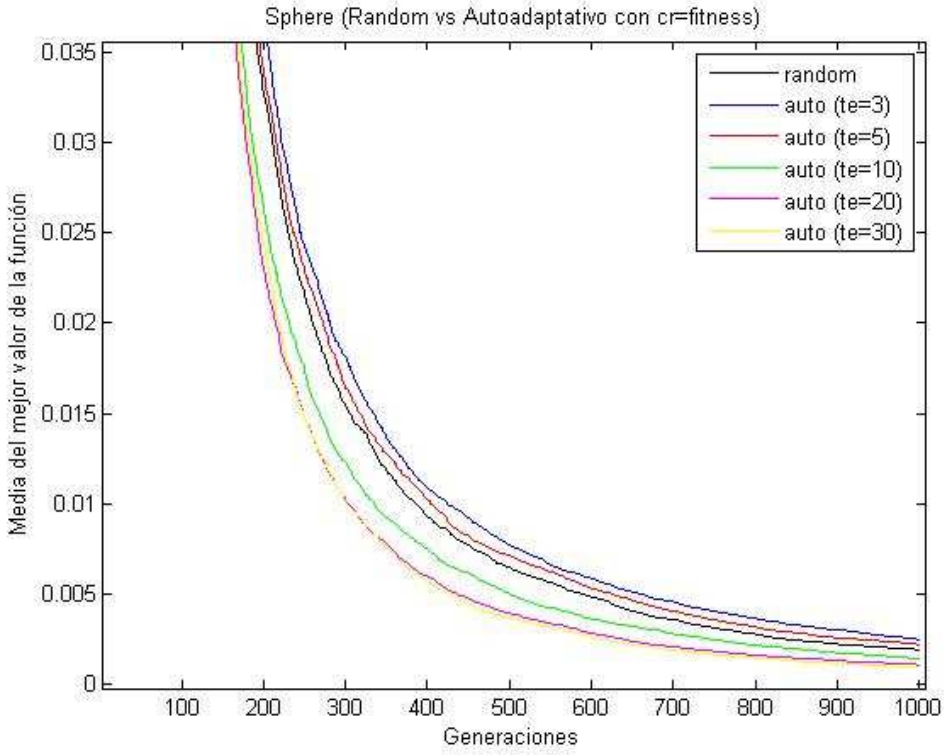

Figura 5.1 (c): Media del mejor valor de la función para la minimización de la función Sphere para el método de emparejamiento Autoadaptativo y criterio $cr = fitness$ (te = tamaño de emparejamiento)

Las figuras 5.2 (a, b y c) representan la evolución, generación a generación, de la media del mejor (mínimo) valor de la función, utilizando varias estrategias de emparejamiento, con $cr = semejanza$. Como puede observarse sigue un patrón similar al que se observa en las figuras 5.1 para $cr = fitness$.

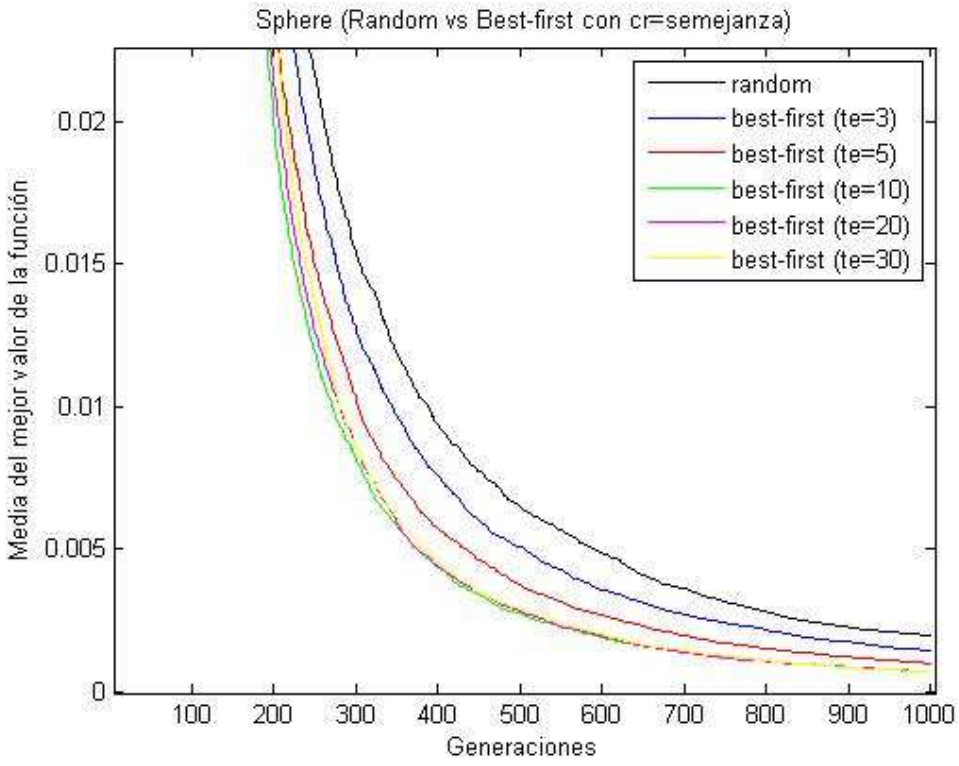

Figura 5.2 (a): Media del mejor valor de la función para la minimización de la función Sphere para el método de emparejamiento Best-First y criterio $cr = semejanza$ (te = tamaño de emparejamiento)

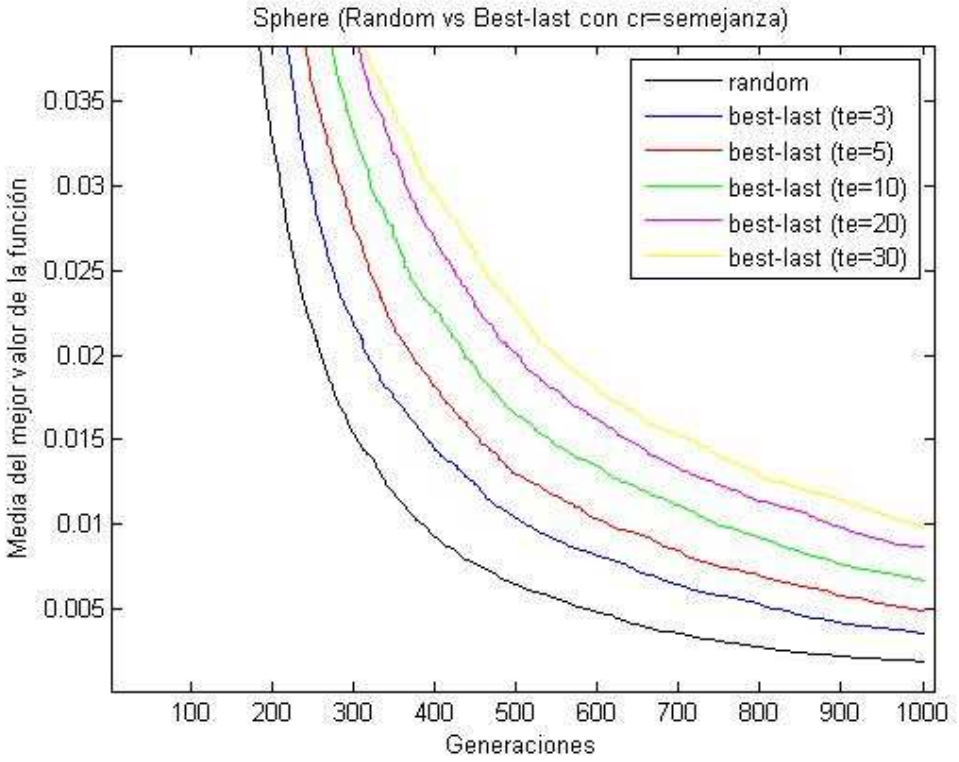

Figura 5.2 (b): Media del mejor valor de la función para la minimización de la función Sphere para el método de emparejamiento Best-Last y criterio *cr = semejanza* (te = tamaño de emparejamiento)

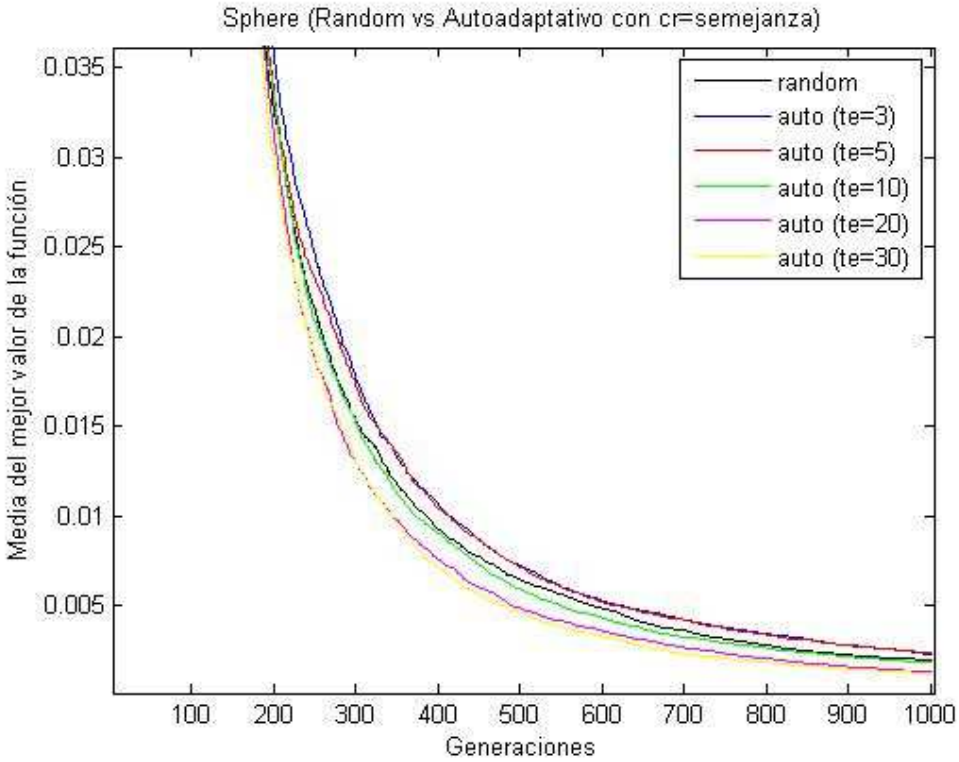

Figura 5.2 (c): Media del mejor valor de la función para la minimización de la función Sphere para el método de emparejamiento Autoadaptativo y criterio *cr = semejanza* (te = tamaño de emparejamiento)

5.1.2 Experimentos con Best-First, Best-Last y Autoadaptativo para la función Schwefel

La función Schwefel [Schwefel (1981)] se define de la siguiente forma:

$$f_{\text{Schwefel}}(x)=\sum_{i=1}^{n} x_i \cdot \sin(\sqrt{|x_i|}) \ .$$

Esta función tiene un máximo en el punto $x=(420.9687,\ldots,420.9687)$ cuyo valor es $n\cdot 418.9829$. Los experimentos se han realizado con $n=10$, $-500\leq x_i\leq 500\ \forall i\in\{1,\ldots,10\}$. Se usó un tamaño de población de 100 individuos.

Las figuras 5.3 (a, b y c) representan la evolución, generación a generación, de la media del mejor (máximo) valor de la función, utilizando varias estrategias de emparejamiento, con $cr=fitness$. En este caso para cada experimento se han realizado quinientas ejecuciones del algoritmo. En las tres figuras también se ha presentado el resultado para emparejamiento al azar (*Random*), como referencia. Junto al caso *Random,* se representan los casos *Best-First* (figura 5.3 (a)), *Best-Last* (figura 5.3 (b)) y *Autoadaptativo* (figura 5.3 (c)). En los tres casos se realizaron experimentos para valores de tamaño de emparejamiento $\gamma=\{3,5,10,20,30\}$.

En este caso, los resultados son opuestos a los obtenidos para Sphere. En la figura 5.3 (b) se observa que *Best-Last* obtiene mejores resultados que *Random* para la función Schwefel. La mejora obtenida por *Best-Last* aumenta al aumentar el valor del tamaño de emparejamiento. En cambio, según puede verse en la figura 5.3 (a), *Best-First* obtiene peores resultados que Random. Los resultados empeoran al aumentar el valor del tamaño de emparejamiento. El método *Autoadaptativo* (figura 5.3 (c)) mejora al disminuir el valor del tamaño de emparejamiento, de forma que obtiene mejores resultados que Random para $\gamma=5$ y $\gamma=3$, obteniendo peores resultados para γ igual o superior a 10.

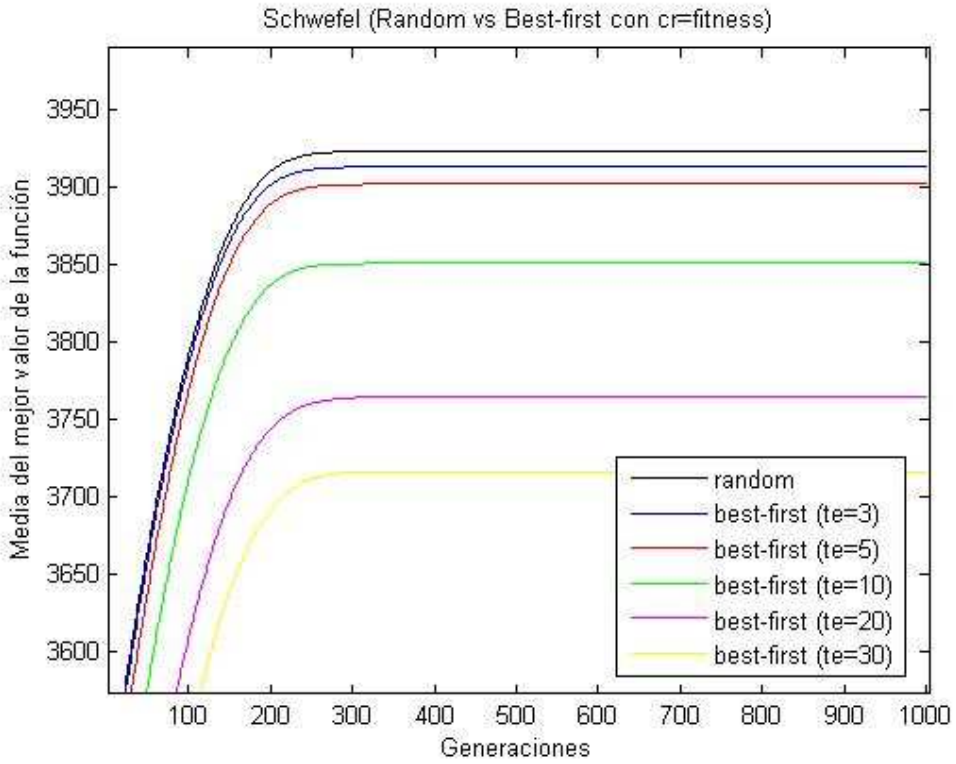

Figura 5.3 (a): Media del mejor valor de la función para la maximización de la función Schwefel para el método de emparejamiento Best-First y criterio *cr = fitness* (te = tamaño de emparejamiento)

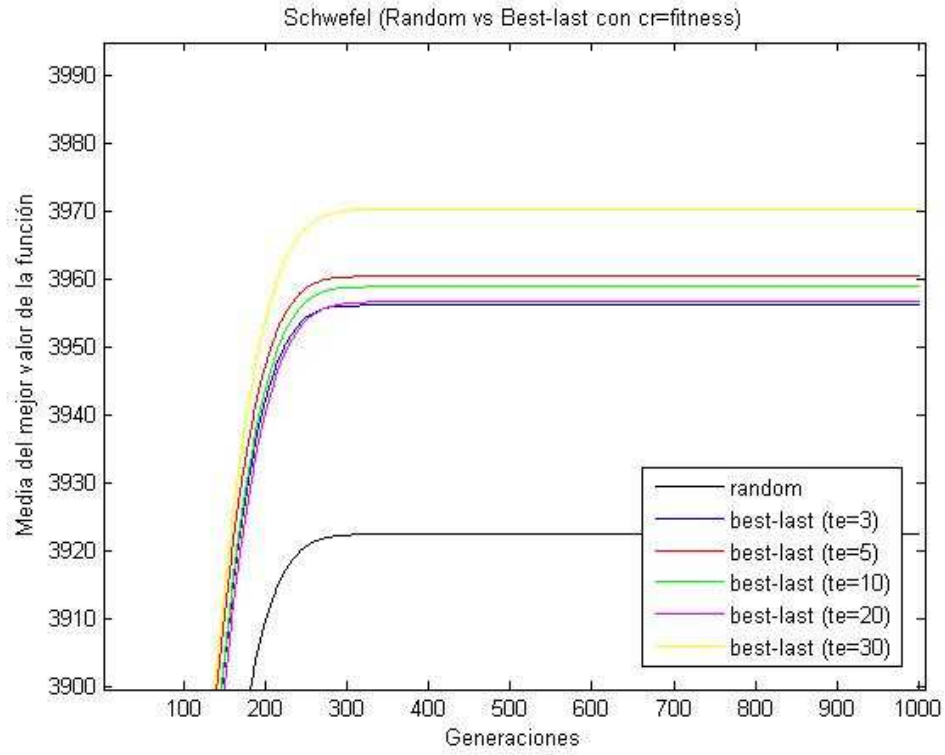

Figura 5.3 (b): Media del mejor valor de la función para la maximización de la función Schwefel para el método de emparejamiento Best-Last y criterio *cr = fitness* (te = tamaño de emparejamiento)

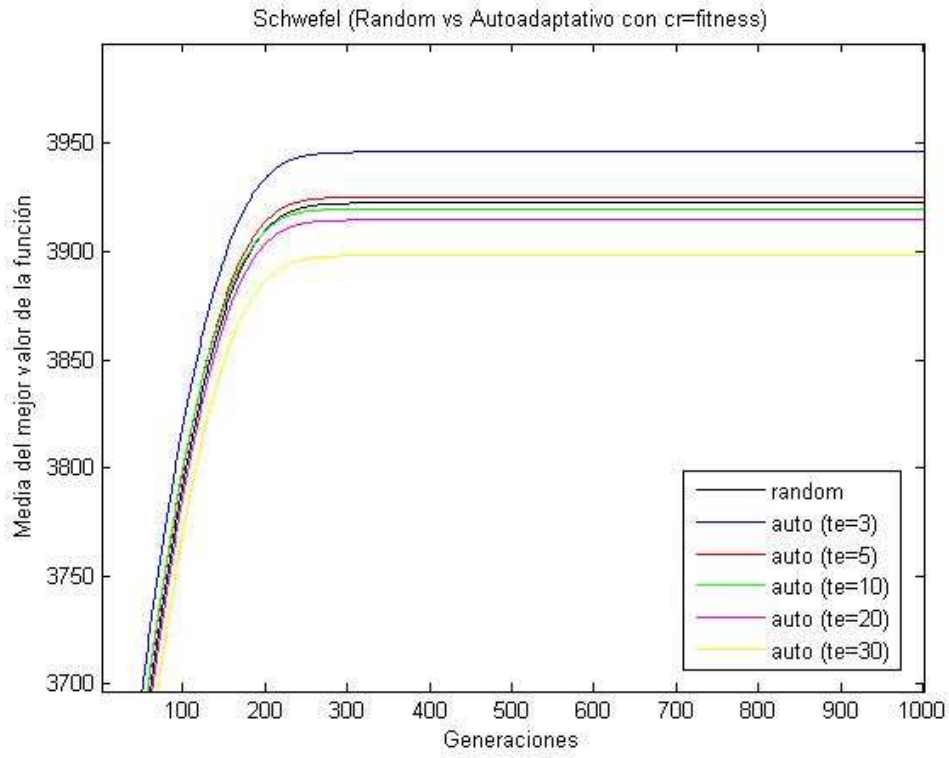

Figura 5.3 (c): Media del mejor valor de la función para la maximización de la función Schwefel para el método de emparejamiento Autoadaptativo y criterio $cr = fitness$ (te = tamaño de emparejamiento)

Las figuras 5.4 (a, b y c) representan la evolución, generación a generación, de la media del mejor (máximo) valor de la función, utilizando varias estrategias de emparejamiento, con $cr = semejanza$. Como puede observarse sigue un patrón similar al que se observa en las figuras 5.3 para $cr = fitness$. Sin embargo, en este caso las mejoras obtenidas sobre el caso *Random* por el método *Best-Last* y el método *Autoadaptativo,* son algo mayores que las obtenidas para $cr = fitness$. En este caso, el método *Autoadaptativo* mejora a *Random* para $\gamma = 3$, $\gamma = 5$ y $\gamma = 10$.

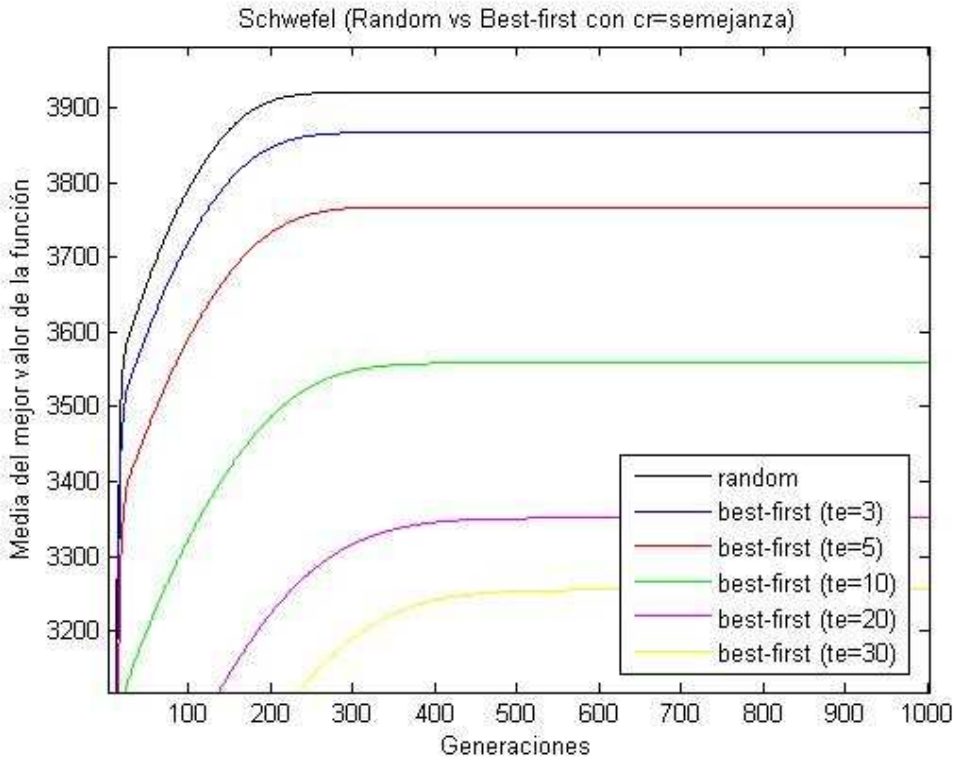

Figura 5.4 (a): Media del mejor valor de la función para la maximización de la función Schwefel para el método de emparejamiento Best-First y criterio *cr = semejanza* (te = tamaño de emparejamiento)

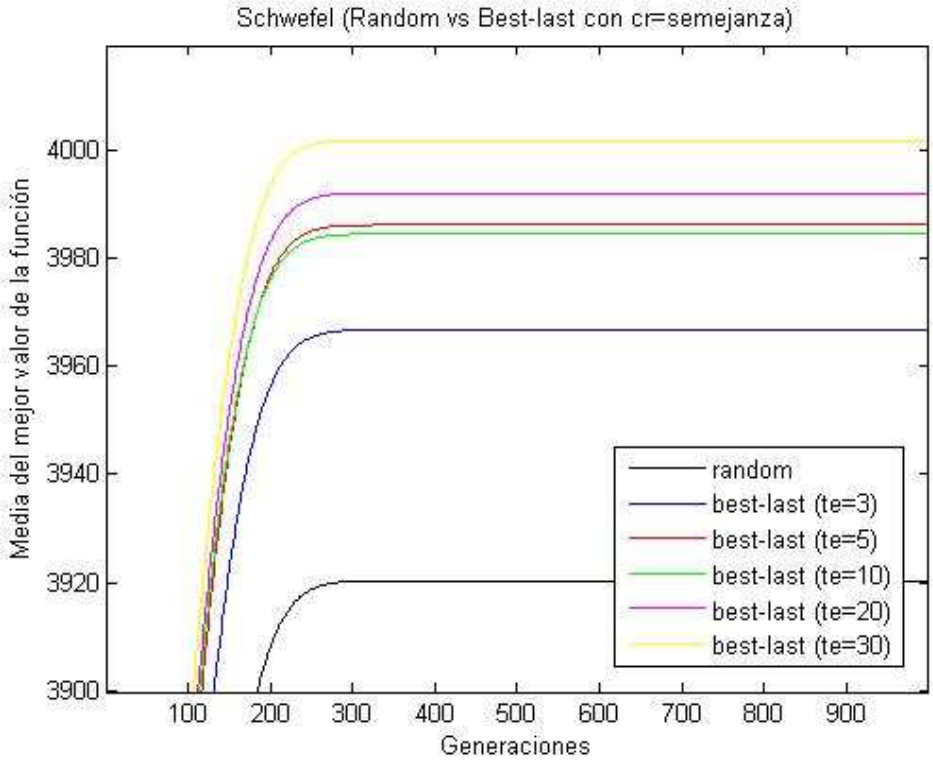

Figura 5.4 (b): Media del mejor valor de la función para la maximización de la función Schwefel para el método de emparejamiento Best-Last y criterio *cr = semejanza* (te = tamaño de emparejamiento)

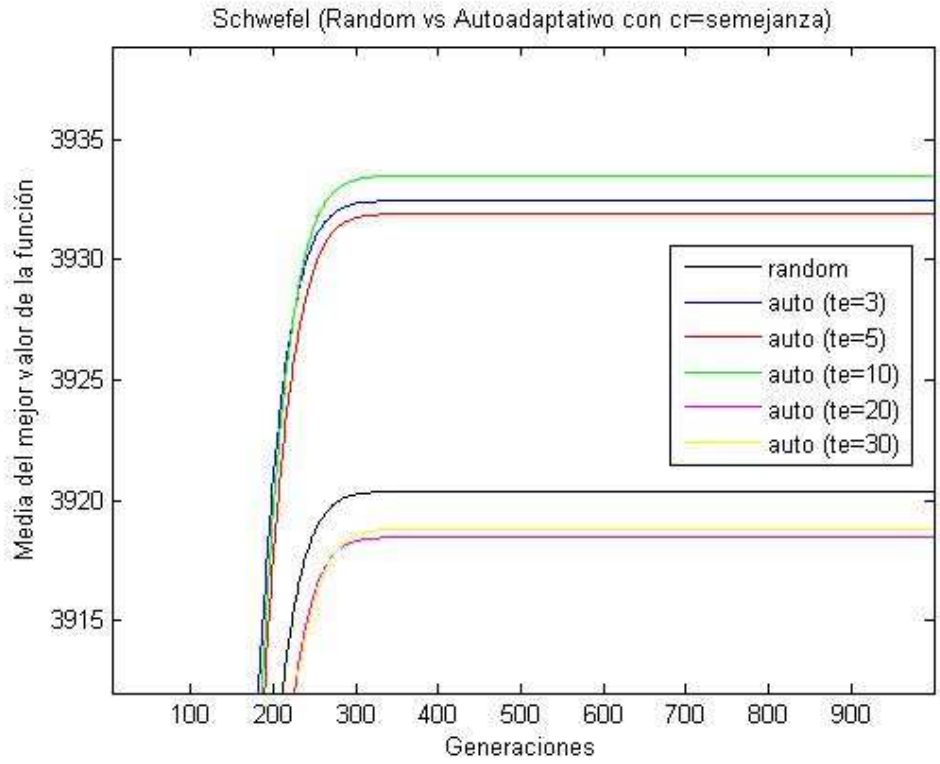

Figura 5.4 (c): Media del mejor valor de la función para la maximización de la función Schwefel para el método de emparejamiento Autoadaptativo y criterio $cr = semejanza$ (te = tamaño de emparejamiento)

5.2 Experimentos variando el valor del índice de emparejamiento

En esta sección se comparan experimentalmente los métodos *Random* y $Best - \alpha$. En principio es de esperar que que el grado de exploración será mayor cuanto mayor sea el valor de α.

Los experimentos realizados nos permitirán comparar:

- Funciones unimodales y multimodales. Para ello se realizan experimentos con la función Sphere (unimodal), la función Schwefel (multimodal) y la función Rastrigin, también multimodal, que podemos considerar intermedia entre Sphere y Schwefel.

- Preferencia para el emparejamiento basada en valor de adaptación y basada en semejanza. Para ello se repiten los experimentos para los dos criterios.

- Métodos de emparejamiento tradicional (*Random*) y avanzado ($Best - \alpha$).

- Varios valores de índice de emparejamiento. Se han realizado experimentos para $\alpha \in \{2,5,10,15,20\}$ y un valor de tamaño de emparejamiento $\gamma = 20$.

La función Rastrigin [Mühlenbein, Schomisch y Born (1991)] se define de la siguiente forma:

$$f_{\text{Rastrigin}}(x) = 10 \cdot n + \sum_{i=1}^{n} (x_i^2 - 10 \cdot \cos(2 \cdot \pi \cdot x_i)) \ .$$

Esta función tiene un mínimo en el punto $x = (0,\ldots,0)$ cuyo valor es 0. Los experimentos se han realizado con $-5.12 \leq x_i \leq 5.12 \ \forall i \in \{1,\ldots,n\}$. La función Rastrigin es una función multimodal. Podemos considerarla, desde el punto de vista de la dificultad de la optimización, como un caso intermedio entre la función unimodal Sphere y la función Schwefel, altamente multimodal.

5.2.1 Experimentos usando como criterio de emparejamiento $cr = semejanza$

La siguiente tabla (tabla 5.1) resume algunos de los parámetros utilizados en los experimentos de esta sección.

Parámetro/**Función**	**Sphere**	**Rastrigin**	**Schwefel**
Dimensiones	10	15	5
Nº ejecuciones	100	250	250
Generaciones	400	400	250
Individuos	50	50	50

Tabla 5.1: Parámetros usados en los experimentos de la sección 5.2.1

Las figuras 5.5 (a, b y c) representan la evolución, generación a generación, de la media del mejor valor de la función, utilizando como método de emparejamiento *Random* y $Best-\alpha$, para varios valores del índice de emparejamiento α, con tamaño de emparejamiento igual a 20 y $cr = semejanza$, para las funciones Sphere, Rastrigin y Schwefel.

A partir de las figuras 5.5 (a y c) vemos que los mejores resultados para Sphere se obtienen para valores bajos de α, mientras que los mejores resultados para Schwefel se obtienen para valores altos de α, como era de esperar. En cambio para Rastrigin (figura

5.5 (b)) los mejores resultados se obtienen para un valor intermedio ($\alpha=15$). Estos resultados coinciden con los obtenidos en [Galán y Mengshoel (2010)] para el caso de representación binaria del cromosoma (vease, por ejemplo, las figuras 9 (a, b y c) del anexo I que corresponden a las figuras 5.5 (a, b y c) de este trabajo).

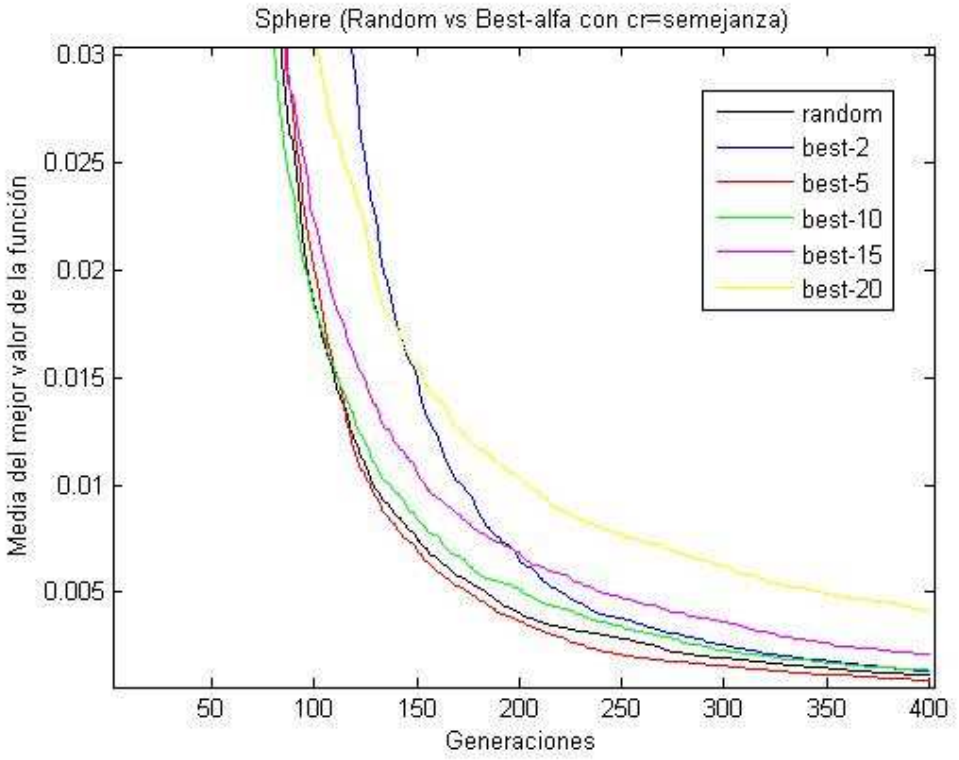

Figura 5.5 (a): Media del mejor valor de la función para la minimización de la función Sphere para el método de emparejamiento *Best* $-\alpha$ y criterio $cr=semejanza$.

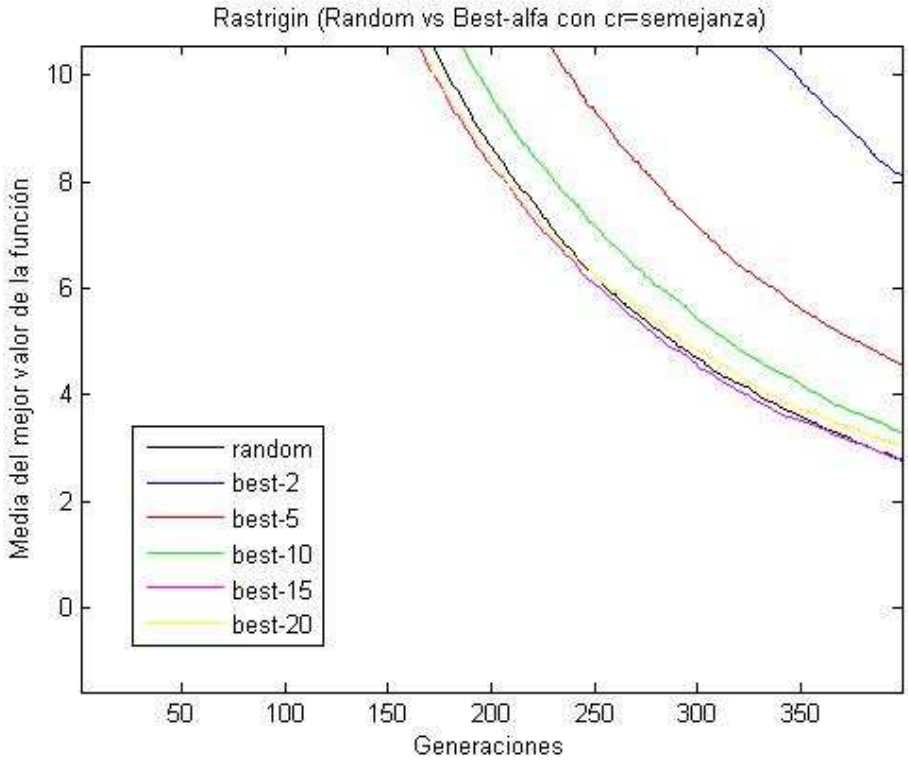

Figura 5.5 (b): Media del mejor valor de la función para la minimización de la función Rastrigin para el método de emparejamiento *Best* $-\alpha$ y criterio $cr = semejanza$

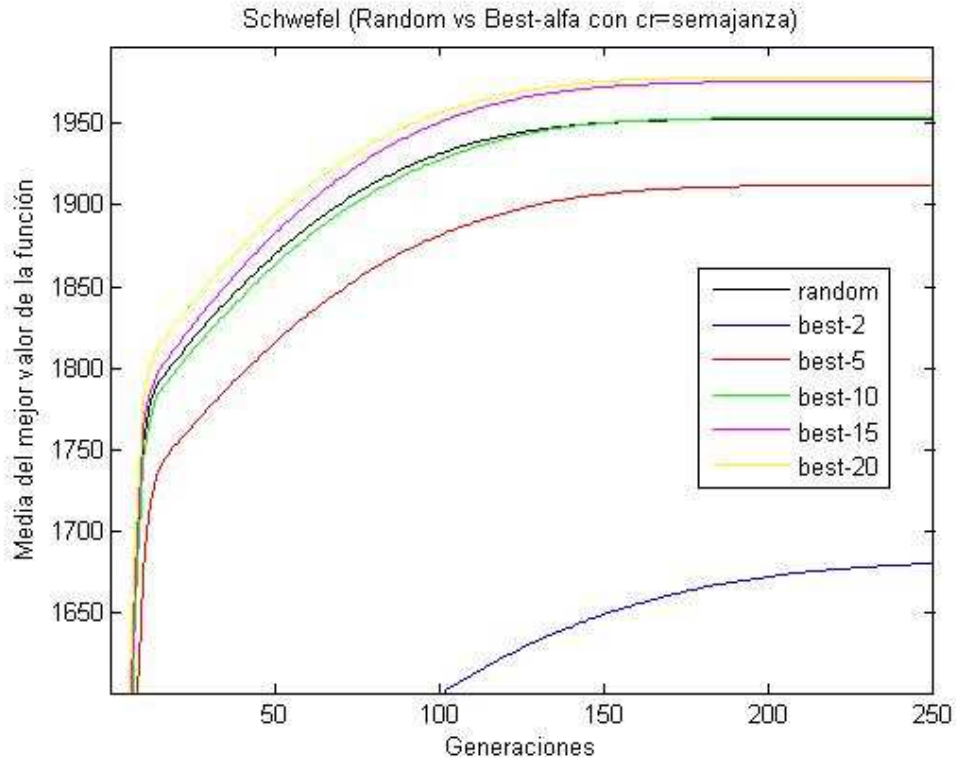

Figura 5.5 (c): Media del mejor valor de la función para la maximización de la función Schwefel para el método de emparejamiento *Best* $-\alpha$ y criterio $cr = semejanza$.

5.2.2 Experimentos usando como criterio de emparejamiento $cr = fitness$

La siguiente tabla (tabla 5.2) resume algunos de los parámetros utilizados en los experimentos de esta sección.

Parámetro/**Función**	**Sphere**	**Rastrigin**	**Schwefel**
Dimensiones	10	15	5
Nº ejecuciones	100	250	250
Generaciones	400	400	250
Individuos	50	50	50

Tabla 5.2: Parámetros usados en los experimentos de la sección 5.2.2

Los parámetros son los mismos con los que se realizaron los experimentos de la sección 5.2.1. Las figuras 5.6 (a, b y c) representan la evolución, generación a generación, de la media del mejor valor de la función, utilizando como método de emparejamiento *Random* y $Best - \alpha$, para varios valores del índice de emparejamiento α, con tamaño de emparejamiento igual a 20 y $cr = fitness$, para las funciones Sphere, Rastrigin y Schwefel.

Como puede observarse los resultados son similares a los obtenidos en 5.2.1. también en este caso los resultados coinciden con los obtenidos en [Galán y Mengshoel (2010)] para el caso de representación binaria del cromosoma (vease, por ejemplo, las figuras 9 (d, e y f) del anexo I que corresponden a las figuras 5.6 (a, b y c) de este trabajo).

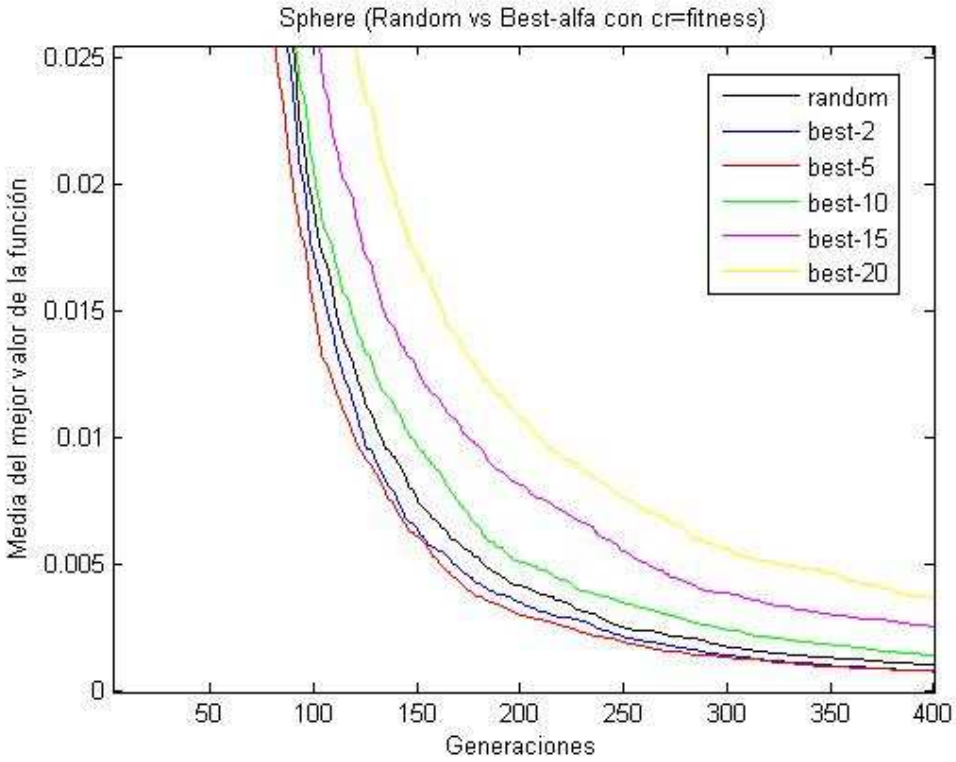

Figura 5.6 (a): Media del mejor valor de la función para la minimización de la función Sphere para el método de emparejamiento $Best-\alpha$ y criterio $cr=fitness$

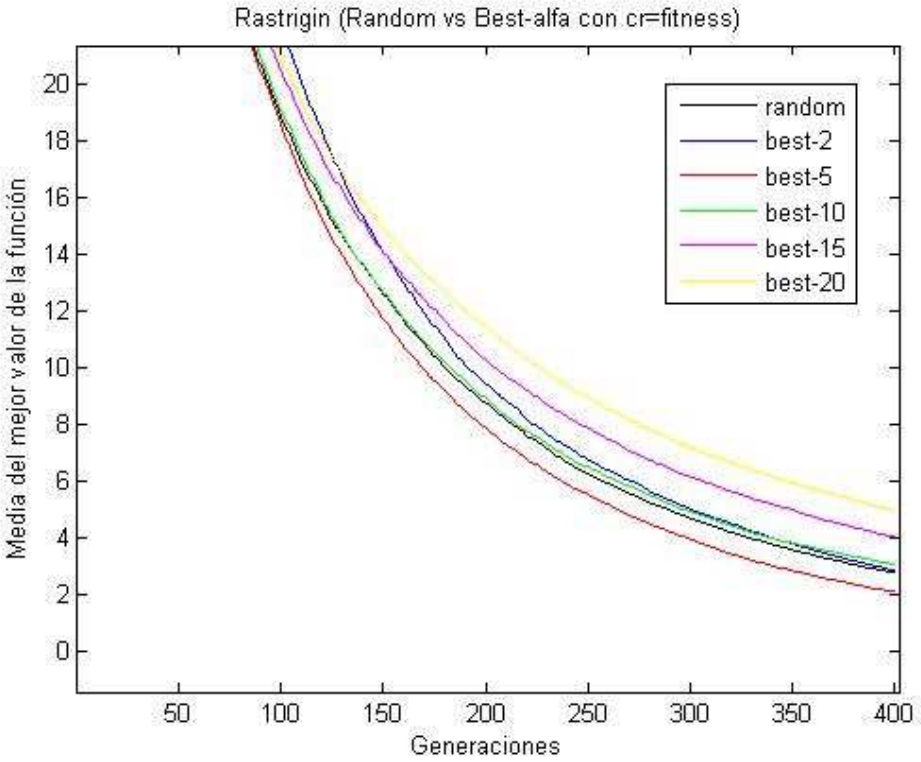

Figura 5.6 (b): Media del mejor valor de la función para la minimización de la función Rastrigin para el método de emparejamiento $Best-\alpha$ y criterio $cr=fitness$

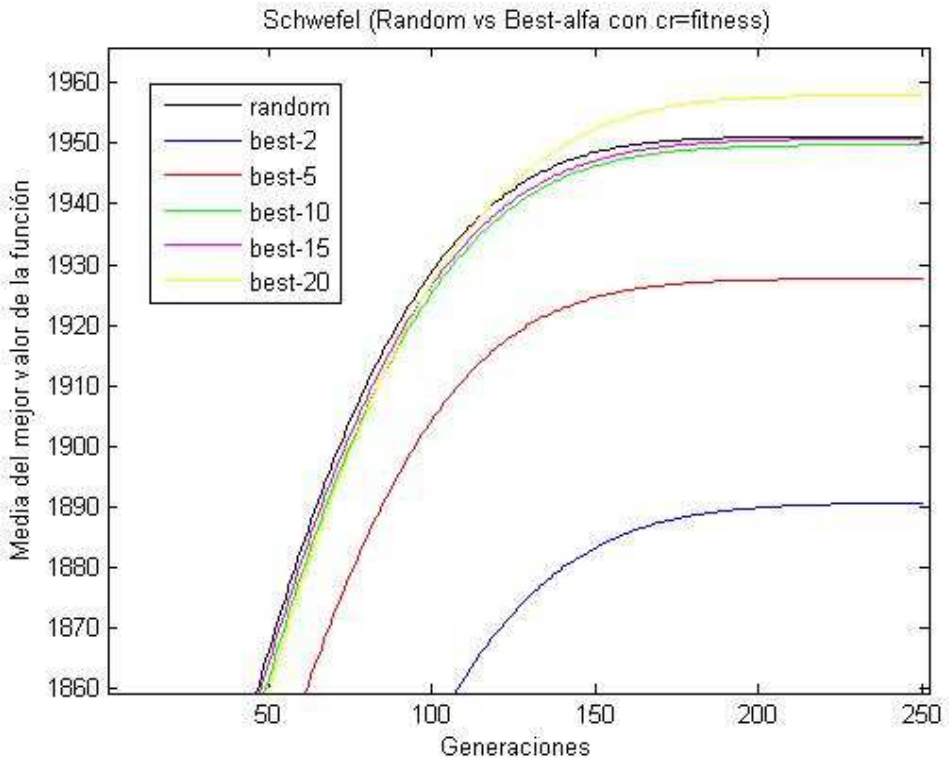

Figura 5.6 (c): Media del mejor valor de la función para la maximización de la función Schwefel para el método de emparejamiento $Best-\alpha$ y criterio $cr=fitness$

5.3 Discusión sobre los experimentos de las secciones 5.1 y 5.2

Los experimentos descritos en las secciones 5.1 y 5.2 confirman, para la representación real, los resultados obtenidos por [Galán y Mengshoel (2010)] para la representación binaria. El emparejamiento *Best-First* es el que obtiene los mejores resultados para problemas unimodales. El comportamiento es tanto mejor cuanto mayor sea el tamaño de emparejamiento. El emparejamiento *Best-Last* es el que obtiene los mejores resultados para problemas multimodales. El resultado es tanto mejor cuanto mayor sea el tamaño de emparejamiento. El método *Autoadaptativo* se comporta de forma diferente según el tipo de problema. Para problemas unimodales mejora el caso *Random* para valores altos del tamaño de emparejamiento y para problemas multimodales mejora el caso *Random* para valores bajos del tamaño de emparejamiento.

También queda confirmada la validez del valor de adaptación como criterio para definir las preferencias de emparejamiento.

Tal como ocurre para la representación binaria, también se observa para la representación real que el método *Autoadaptativo* no ofrece un comportamiento lo suficientemente robusto como para poder aplicarlo con confianza a cualquier tipo de problema, ya que se comporta bien para valores bajos o altos del tamaño de emparejamiento según se trate de problemas multimodales o unimodales. No obstante, los resultados obtenidos para el método *Autoadaptativo* usando la representación real son mejores que los obtenidos con la representación binaria (esto se pone de manifiesto especialmente al optimizar funciones más complejas, como Schwefel: puede compararse la figura 5.4 (c) con la figura 8 (c) del anexo I).

La representación real también tiene la ventaja de un menor coste computacional que se manifiesta en un tiempo de ejecución menor. Ello es debido a que, en la representación binaria, hay que realizar continuamente conversiones entre genotipo y fenotipo, para calcular el valor de adaptación, mientras que en la representación real no es necesario realizar ninguna conversión, ya que fenotipo y genotipo coinciden.

Con los métodos espacial y temporal intentaremos explorar métodos que ofrezcan un comportamiento robusto independientemente del tipo de problema.

5.4 Experimentos para el método espacial

Para estos experimentos hemos usado como criterio de emparejamiento únicamente $cr = semejanza$, ya que pensamos que ha quedado suficientemente constatada la equivalencia de ambos criterios de emparejamiento, lo que hace innecesario la repetición de los experimentos para ambos casos.

Para los experimentos realizados para los métodos espaciales y temporal (secciones 5.4 y 5.5) se han utilizado los mismos parámetros de los experimentos de la sección 5.1 que resumimos en la tabla 5.3.

Parámetro/**Función**	**Sphere**	**Schwefel**
Dimensiones	20	10
Nº ejecuciones	100	500
Generaciones	1000	1000
Individuos	100	100

Tabla 5.3: Parámetros usados en los experimentos de las secciones 5.4 y 5.5

5.4.1 Experimentos para el método espacial-1

La figura 5.7 representa la evolución, generación a generación, de la media del mejor valor de la función, utilizando el método de emparejamiento *Espacial-1,* para la función Sphere, para varios valores del tamaño de emparejamiento.

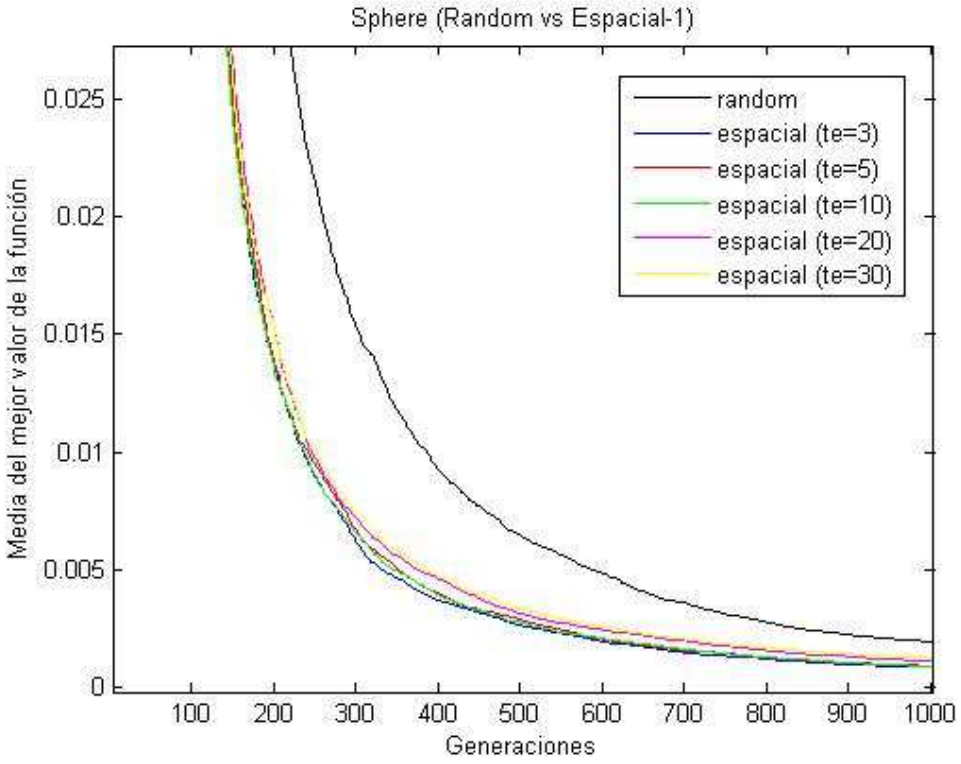

Figura 5.7: Media del mejor valor de la función para la minimización de la función Sphere para el método de emparejamiento *Espacial-1* y criterio *cr = semejanza* (te = tamaño de emparejamiento)

La figura 5.8 representa la evolución, generación a generación, de la media del mejor valor de la función, utilizando el método de emparejamiento *Espacial-1,* para la función Schwefel, para varios valores del tamaño de emparejamiento.

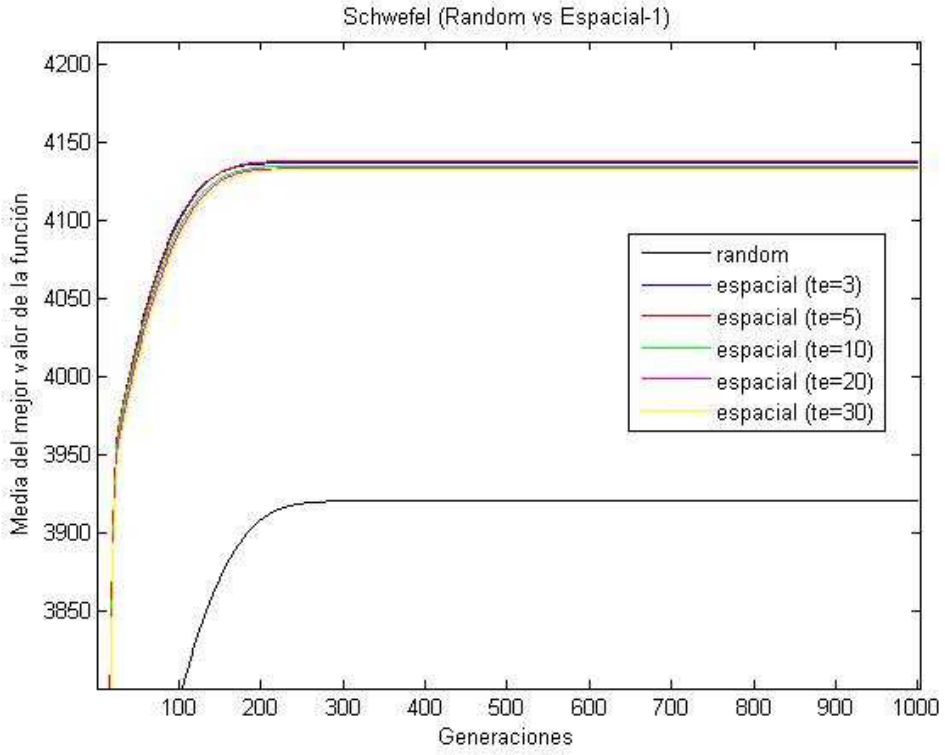

Figura 5.8: Media del mejor valor de la función para la maximización de la función Schwefel para el método de emparejamiento *Espacial-1* y criterio $cr = semejanza$ (te = tamaño de emparejamiento)

Como se ve el método *Espacial-1* mejora claramente los resultados de *Random*. Sin embargo la comparación se hace difícil al no tener *Espacial-1* selección de padres. Por ello, sobre el mismo algoritmo del método *Espacial-1*, por lo tanto sin selección de padres, realizamos otros experimentos con emparejamiento al azar (*Random*, para lo que hacemos $\gamma = \alpha = 2$), emparejamiento Best-First (para ello hacemos $\alpha = 2$, $\gamma = 30$) y emparejamiento Best-Last (para lo que hacemos $\alpha = \gamma = 30$). En las figuras 5.9 y 5.10 se presentan los resultados junto con los de *Espacial-1* para $\gamma = 30$, para la optimización de las funciones Sphere y Schwefel, respectivamente. En estas últimas figuras, por lo tanto, en ninguno de los métodos comparados se ha realizado selección de padres.

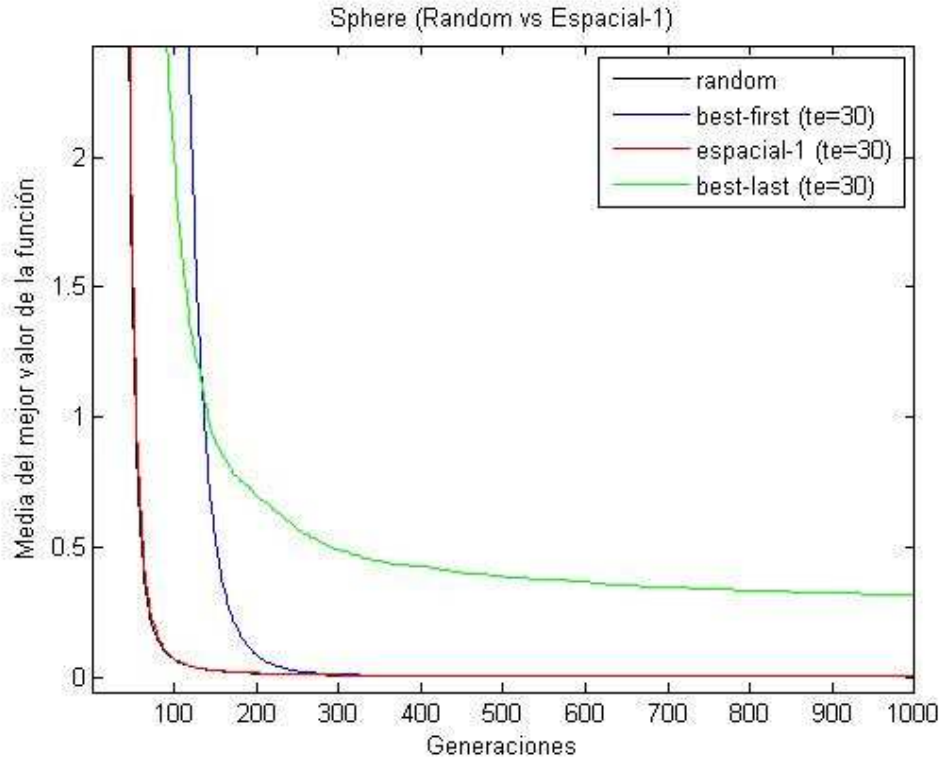

Figura 5.9: Media del mejor valor de la función para la maximización de la función Sphere para los métodos de emparejamiento *Espacial-1, Best-First* y *Best-Last* y criterio *cr = semejanza* . En todos los casos se ha quitado la selección de padres (te = tamaño de emparejamiento)

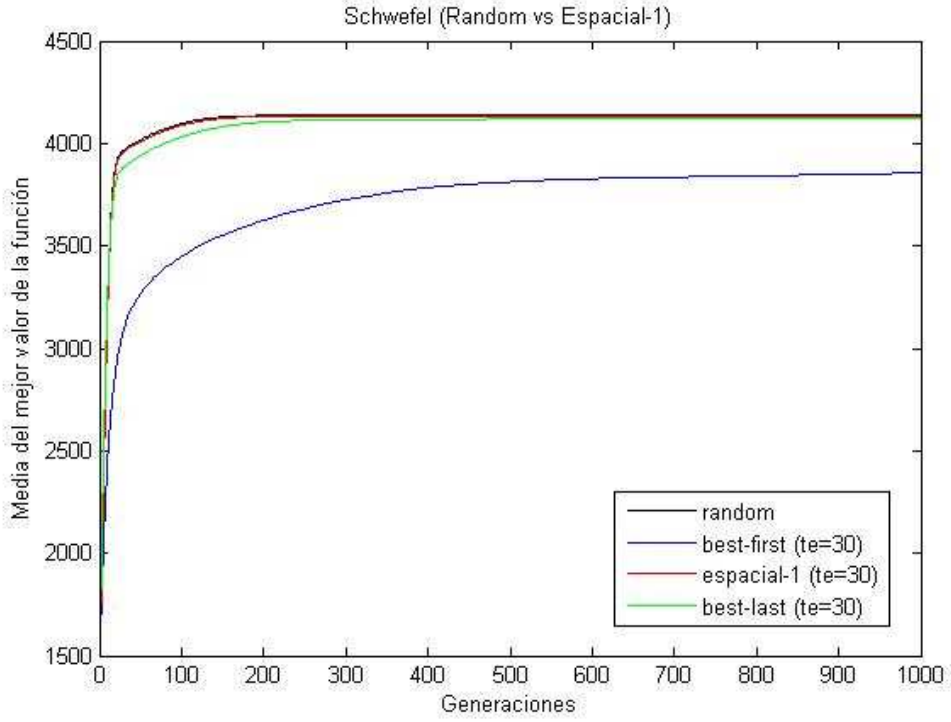

Figura 5.10: Media del mejor valor de la función para la maximización de la función Schwefel para los métodos de emparejamiento *Espacial-1, Best-First* y *Best-Last* y criterio *cr = semejanza* . En todos los casos se ha quitado la selección de padres (te = tamaño de emparejamiento)

5.4.2 Experimentos para el método espacial-2

La figura 5.11 representa la evolución, generación a generación, de la media del mejor valor de la función, utilizando el método de emparejamiento *Espacial-2,* para la función Sphere, para varios valores del tamaño de emparejamiento.

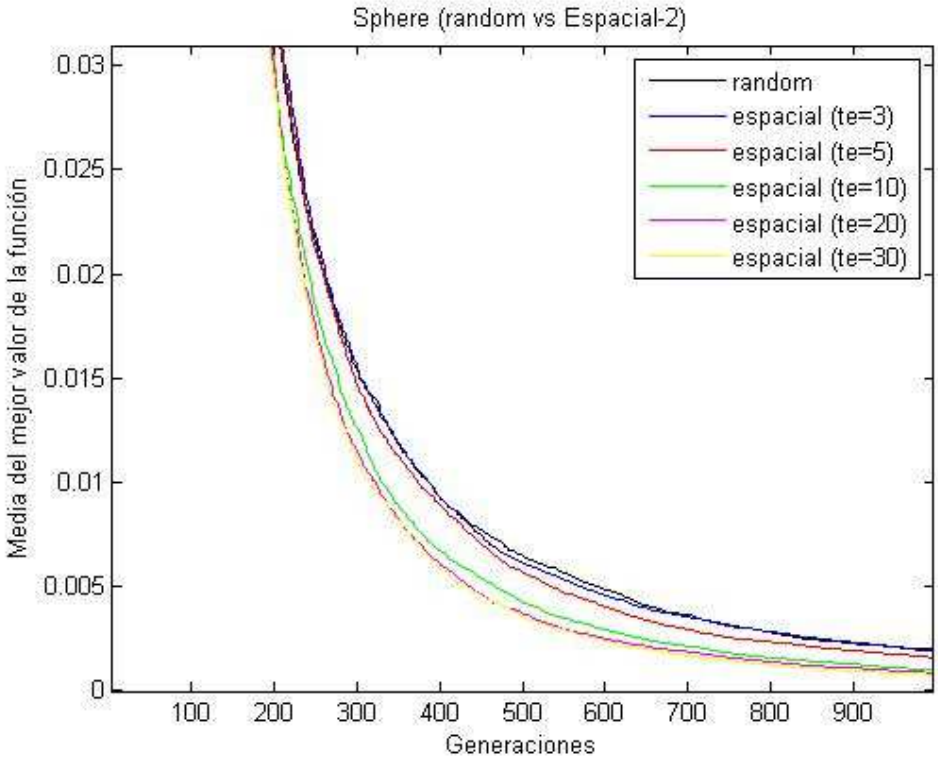

Figura 5.11: Media del mejor valor de la función para la minimización de la función Sphere para el método de emparejamiento *Espacial-2* y criterio *cr = semejanza* (te = tamaño de emparejamiento)

La figura 5.12 representa la evolución, generación a generación, de la media del mejor valor de la función, utilizando el método de emparejamiento *Espacial-2,* para la función Schwefel, para varios valores del tamaño de emparejamiento.

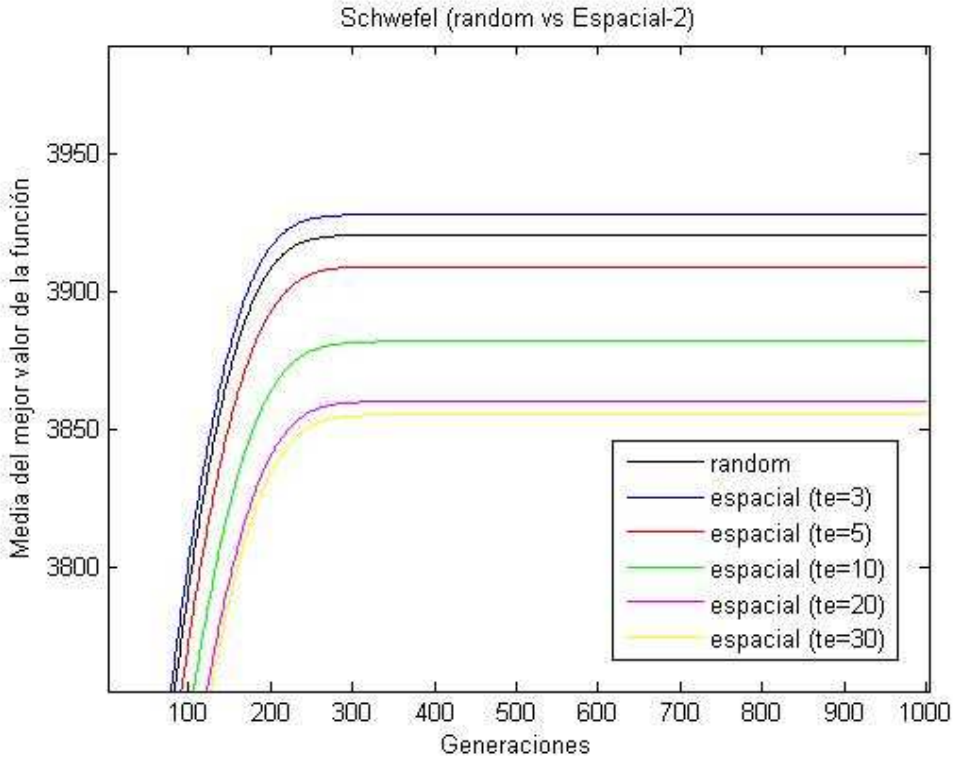

Figura 5.12: Media del mejor valor de la función para la maximización de la función Schwefel para el método de emparejamiento *Espacial-2* y criterio $cr = semejanza$ (te = tamaño de emparejamiento)

5.5 Experimentos para el método temporal

Para el método temporal, como está claro que se ha de comportar como $Best - First$ para valores bajos de la constante de decaimiento y como $Best - Last$ para valores altos de dicha constante, se han realizado experimentos para intentar determinar si existe un rango de valores de la constante de decaimiento tal que el algoritmo tenga un comportamiento robusto, independientemente del tipo de problema. Para ello se han realizado experimentos variando el valor de la constante de decaimiento (K), para un tamaño de emparejamiento $\gamma = 30$.

La figura 5.13 representa la evolución, generación a generación, de la media del mejor valor de la función, utilizando el método de emparejamiento *Temporal,* para la función Sphere, para varios valores de la constante de decaimiento.

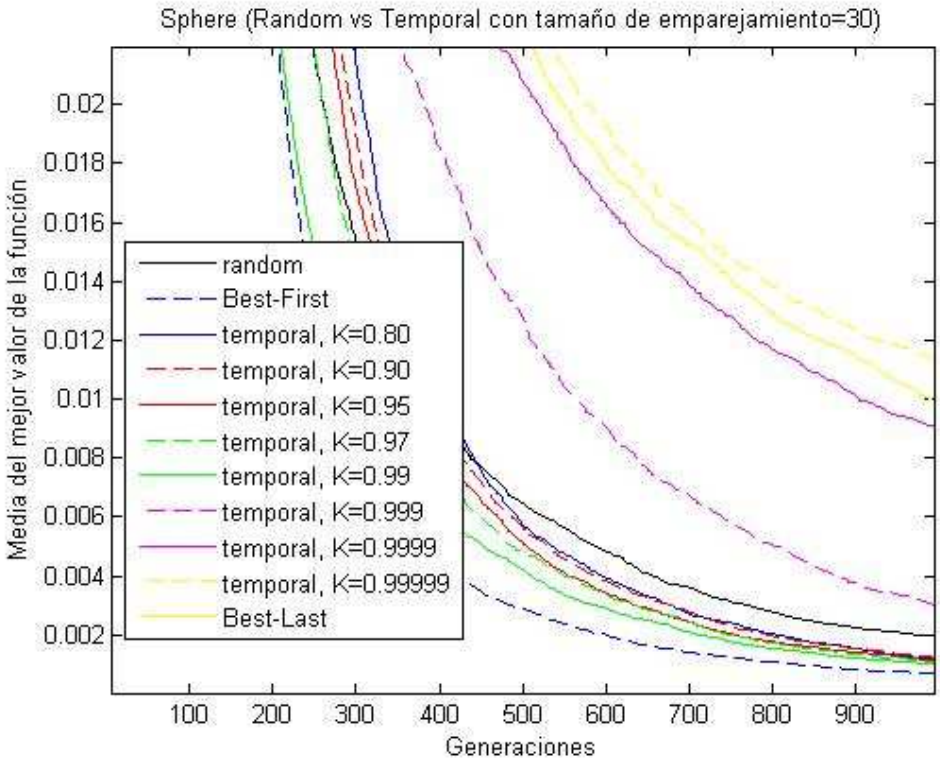

Figura 5.13: Media del mejor valor de la función para la minimización de la función Sphere para el método de emparejamiento *Temporal* y criterio *cr = semejanza*

La figura 5.14 representa la evolución, generación a generación, de la media del mejor valor de la función, utilizando el método de emparejamiento *Temporal,* para la función Schwefel, para varios valores de la constante de decaimiento.

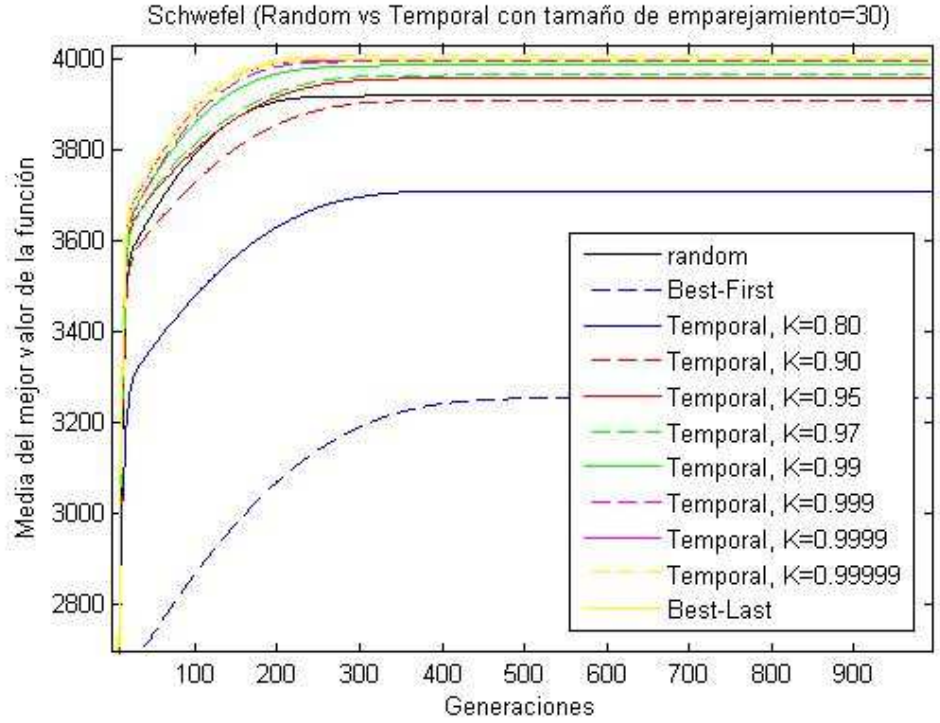

Figura 5.14: Media del mejor valor de la función para la maximización de la función Schwefel para el método de emparejamiento *Temporal* y criterio *cr = semejanza*

5.6 Discusión sobre los experimentos de las secciones 5.4 y 5.5

Comparando la gráfica 5.2(c) con la gráfica 5.7 y la gráfica 5.4(c) con la gráfica 5.8, se observa que el método *Espacial-1* mejora ampliamente los resultados del método *Autoadaptativo*. Sin embargo, el hecho de que en el método *Espacial-1* no aparezcan diferencias entre los diferentes valores del tamaño de emparejamiento, nos hace pensar que la causa de la mejora radica en que, al haber eliminado la selección de padres, hemos aumentado la diversidad permitiendo la reproducción de individuos con peor valor de adaptación. Esta interpretación la damos con reservas ya que, si bien parece lógico que el aumento de exploración, debido a la inexistencia de la selección de padres, mejore los resultados al optimizar una función altamente multimodal como Schwefel (figura 5.8), debería empeorar los resultados al tratar de optimizar la función Sphere, ¡y también los mejora! (figura 5.7). No consideramos, por lo tanto, este método comparable con el *Autoadaptativo* (que sí tiene selección de padres) aunque sí parece ser un método robusto, que mejora los resultados de *Random* para problemas unimodales y multimodales. Esta interpretación parece justificada por los resultados representados en las figuras 5.9 y 5.10: al quitar la selección de padres se ven diferencias significativas entre *Best-First* y *Best-Last* pero no entre *Random* y el mejor de los otros dos (*Best-First* en el caso de Sphere y *Best-Last* en el caso de Schwefel). Al ser *Random* igual al mejor posible (*Best-First* o *Best-Last*) no hay espacio para que el método *Espacial-1* pueda mejorar a *Random*: *Espacial-1* iguala a *Random* y al mejor de los otros dos (*Best-First* o *Best-Last*) quedando los tres métodos iguales. Al quitar la selección de padres el efecto del emparejamiento no se detecta. De estos resultados podremos extraer algunas conclusiones en el siguiente capítulo.

En cuando al método *Espacial-2*, no es más robusto que el método *Autoadaptativo* ya que, aunque mejora los resultados de *Random* en el caso de la función Sphere, no consigue mejorarlos en el caso de la función Schwefel (en los experimentos realizados sólo pudo mejorar los resultados de *Random* para un tamaño de emparejamiento igual a 2).

Estos resultados tan diferentes entre ambos métodos espaciales se ponen de manifiesto si observamos la evolución del índice de emparejamiento del individuo con mejor valor de adaptación.

Las figuras 5.15 y 5.16 representan la evolución del índice de emparejamiento del individuo con mejor valor de adaptación, utilizando el método de emparejamiento *Espacial-1*, para la optimización de las funciones Sphere y Schwefel, respectivamente.

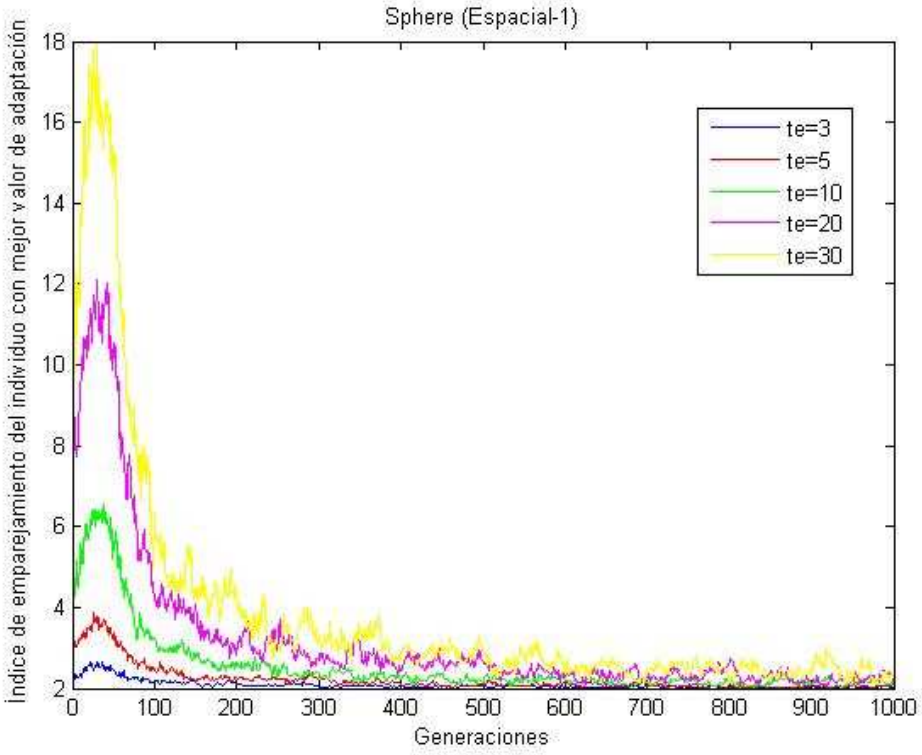

Figura 5.15: Índice de emparejamiento del individuo con mejor valor de adaptación para la minimización de la función Sphere con el método de emparejamiento *Espacial-1* y criterio *cr* = *semejanza* (te = tamaño de emparejamiento)

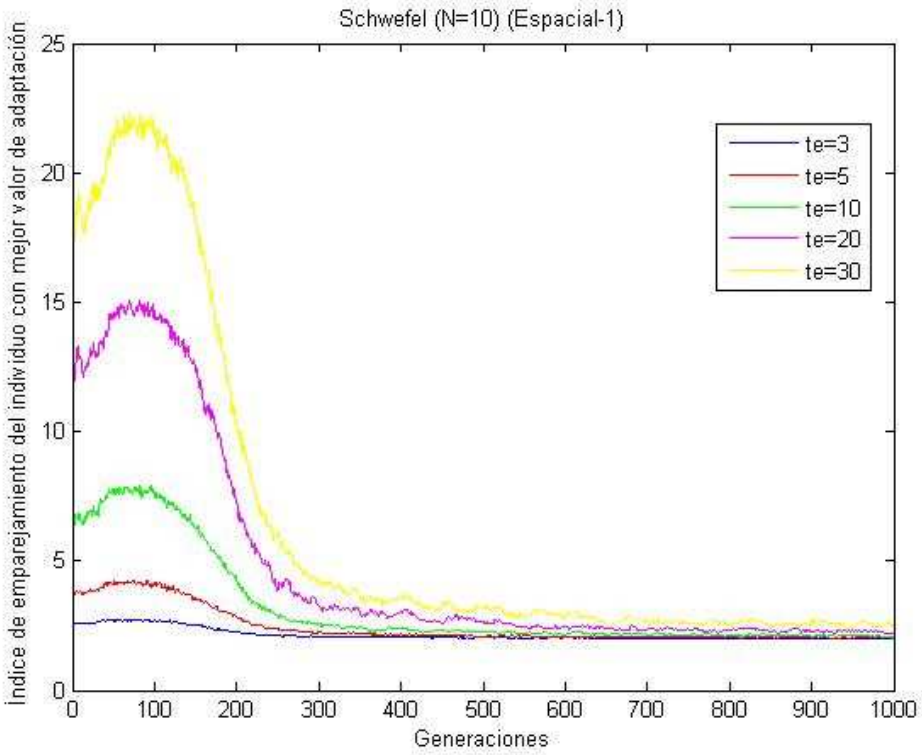

Figura 5.16: Índice de emparejamiento del individuo con mejor valor de adaptación para la maximización de la función Schwefel con el método de emparejamiento *Espacial-1* y criterio *cr* = *semejanza* (te = tamaño de emparejamiento)

Las figuras 5.17 y 5.18 representan la evolución del índice de emparejamiento del individuo con mejor valor de adaptación, utilizando el método de emparejamiento *Espacial-2*, para la optimización de las funciones Sphere y Schwefel, respectivamente.

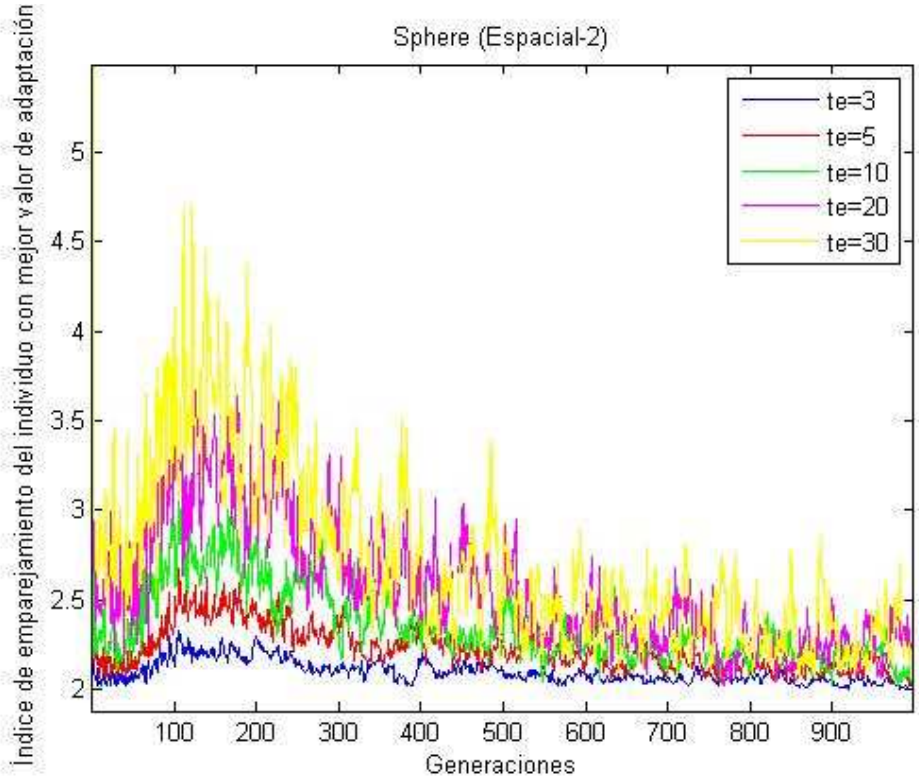

Figura 5.17: Índice de emparejamiento del individuo con mejor valor de adaptación para la minimización de la función Sphere con el método de emparejamiento *Espacial-2* y criterio $cr = semejanza$ (te = tamaño de emparejamiento)

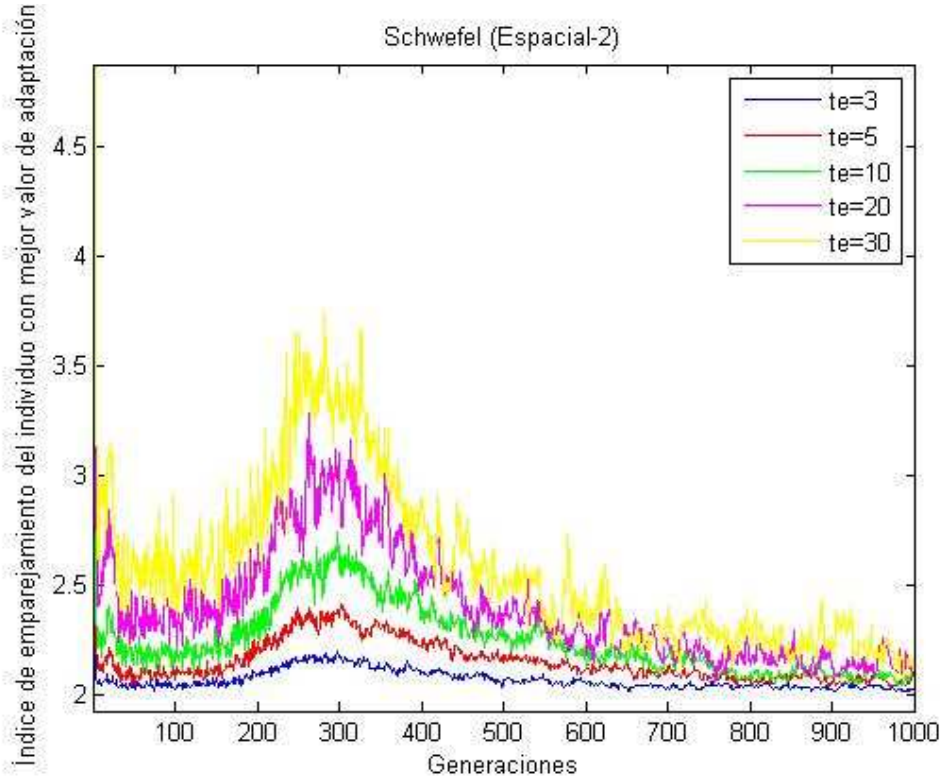

Figura 5.18: Índice de emparejamiento del individuo con mejor valor de adaptación para la maximización de la función Schwefel con el método de emparejamiento *Espacial-2* y criterio *cr = semejanza* (te = tamaño de emparejamiento)

Como puede observarse, en todos los casos hay una fase explorativa seguida de una fase explotativa. En ambos métodos la fase explorativa parece alargarse más en el tiempo en el caso de la función Schwefel: de alguna forma el método espacial se adapta al problema más complejo alargando la fase explorativa. En el método *Espacial-1*, el que obtiene los mejores resultados, el índice de emparejamiento del individuo con mejor valor de adaptación evoluciona de una forma más estable y alcanza valores mucho más altos que en el caso del método *Espacial-2*, que obtiene peores resultados.

En cuanto al método *Temporal*, se observa que, en el caso de Sphere, mejora a *Random* para $K < 0.999$ y, en el caso de Schwefel, mejora a *Random* para $K > 0.90$. Por lo tanto hay un intervalo de valores de K para los que el método temporal se comporta como un método robusto, que mejora los resultados de *Random* para cualquier tipo de problemas. En el caso estudiado (tamaño de emparejamiento $\gamma = 30$) el método se comporta de forma robusta para $0.90 < K < 0.999$, especialmente para $K \approx 0.99$.

En realidad el caso es muy similar al del método general donde también podemos variar el valor del índice de emparejamiento según el tipo de problema. Sin embargo, el método *Temporal* tiene la ventaja de que, si queremos ajustar el algoritmo para obtener los mejores resultados para un problema concreto de optimización, es más fácil y más preciso

ajustar un valor real como la constante de decaimiento (K) que un valor entero como el índice de emparejamiento. Además el método temporal, para algunos valores de la constante de decaimiento, se comporta como un método robusto que funciona bien en problemas unimodales y multimodales.

Si comparamos el método *Temporal* con el *Autoadaptativo*, vemos que el método temporal ofrece un comportamiento más robusto, pero el *Autoadaptativo*, cuyos resultados también son aceptables con la representación real, no necesita que ajustemos ningún parámetro.

CAPÍTULO 6 CONCLUSIONES Y TRABAJO FUTURO

En este capítulo enumeramos las conclusiones más importantes que se derivan del presente trabajo y proponemos una serie de ideas que pueden resultar de interés para investigaciones futuras.

El presente trabajo de fin de máster confirma, para la representación real, los resultados obtenidos por Galán y Mengshoel [Galán y Mengshoel (2010)] para la representación binaria. Se han repetido todos los experimentos (secciones 5.1 y 5.2) de [Galán y Mengshoel (2010)], en este caso usando representación real y, en todos los casos, los resultados confirman los allí obtenidos, mejorando algo los resultados en el caso del método *Autoadaptativo*. Esta mejora es debida al hecho de haber empleado una representación más cercana al problema a resolver.

Se ha confirmado que las restricciones de emparejamiento basadas en el valor de adaptación (criterio $cr = fitness$) ofrecen resultados equivalentes a las restricciones basadas en la semejanza (criterio $cr = semejanza$), con menor coste computacional.

El índice de emparejamiento permite ajustar el algoritmo de emparejamiento al problema concreto a resolver. Así, el emparejamiento *Best-First* es el que obtiene mejores resultados para funciones unimodales, mientras que el emparejamiento *Best-Last* es el que obtiene mejores resultados para problemas multimodales. Cuando el grado de dificultad del problema es desconocido el método *Autoadaptativo* puede ser una buena opción ya que mejora claramente los resultados de *Random* en los problemas unimodales y ofrece unos resultados aceptables en los problemas multimodales. En el caso del método *Autoadaptativo* los resultados mejoran con la representación real, confirmando la

validez del método pero también que adolece de cierta falta de robustez, ya que, para problemas complejos y valores altos del tamaño de emparejamiento, los resultados quedan, claramente, por debajo de *Random* (figura 5.4(c)).

Al intentar un método más robusto que el método *Autoadaptativo*, se han propuesto los métodos que hemos denominado *Espaciales* y *Temporal*. El método *Espacial-2* ofrece unos pobres resultados mientras que los buenos resultados obtenidos por el método *Espacial-1* quedan enmascarados por el hecho de que en este método no hay selección de padres. Sin embargo, los resultados obtenidos para el método *Espacial-1* parecen indicar que sería ventajoso aumentar la diversidad de la población. Para esto proponemos varias estrategias. Una sería modificar ligeramente el algoritmo de recombinación, de forma que, a partir de dos padres, se generen más de dos hijos, de los que nos quedaríamos finalmente con los dos mejores. Otra sería obtener varios candidatos en la fase de mutación, usando distintos valores de la desviación estándar, para finalmente quedarnos con el mejor. Una tercera posibilidad sería incorporar, de forma continuada o cada cierto número de generaciones, nuevos individuos, obtenidos al azar, a la población. Por último, se podría sustituir el reemplazo generacional utilizado en el algoritmo por un reemplazo (μ,λ) o $(\mu+\lambda)$. Se ha realizado algún experimento utilizando reemplazo $(\mu+\lambda)$, habiendo obtenido mejoras discretas en el caso de representación binaria y ningún tipo de mejora en el caso de representación real, pero es una vía que habría que explorar con más detenimiento en el futuro.

Proponemos, para un futuro trabajo, la implementación de alguno de los mecanismos del párrafo anterior y la repetición de todos los experimentos.

En cuanto al método Temporal, es más robusto que el método Autoadaptativo. Esto es debido a que el algoritmo genético mejora al aplicar una mayor exploración inicial y una mayor explotación final. En efecto, ajustando la constante de decaimiento a $K \approx 0.99$, permite optimizar tanto funciones unimodales como multimodales, mejorando los resultados de *Random* para ambos tipos de problemas. Por otra parte, si conocemos el tipo de problema que queremos optimizar, permite que ajustemos el algoritmo al problema, variando la constante de decaimiento, de forma más precisa que variando el índice de emparejamiento.

BIBLIOGRAFÍA

[Bäck (1996)] T. Bäck, *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, Oxford, UK, 1996.

[Banzhaf, Nordin y Keller (1999)] W. Banzhaf, P. Nordin, R. E. Keller, F. D. Francone. *Genetic Programming: An Introduction*. Morgan Kaufmann, San Francisco, 1999.

[De Jong (1975)] K. A. De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, 1975.

[Deb y Goldberg (1989)] K. Deb, D. E. Goldberg. *An investigation of niche and species formation in genetic function optimization*. En Proceedings of the 3rd International Conference on Genetic Algorithms (ICGA'89), pp 42-50, George Mason University, Fairfax, VA. Morgan Kaufmann, San Francisco, CA. 1989.

[Eiben y Smith (2003)] A. E. Eiben, J. E. Smith. *Introduction to Evolutionary Computing*. Springer. Berlin. Heidelberg. 2003.

[Eshelman y Schaffer (1991)] L. J. Eshelman, J. D. Schaffer, *Preventing premature convergence in genetic algorithms by preventing incest*. En Proceedings of the 4th International Conference on Genetic Algorithms (ICGA'91), pp 115-122, San Diego, CA. Morgan Kaufmann, San Francisco, CA. 1991.

[Fernandes (2001)] C. Fernandes, R. Tavares, C. Munteanu, A. Rosa. *Using assortative mating in genetic algorithms for vector quantization problems*. En: Proceedings of the 2001 ACM Symposium on Applied Computing (ACM SAC'2001), pp 361-365, Las Vegas, NV. ACM, New York. 2001.

[Fogel, Owens y Walsh (1965)] L. J. Fogel, A. J. Owens, M. J. Walsh. Artificial intelligence through a simulation of evolution. En: A. Callahan, M. Maxfield, L. J. Fogel, Eds., *Biophysics and Cybernetic Systems*. Spartan, Washington, DC, 1965.

[Fogel, Owens y Walsh (1966)] L. J. Fogel, A. J. Owens, M. J. Walsh. *Artificial Intelligence Through Simulated Evolution*. Wiley. Chichester, UK, 1966.

[Galán y Mengshoel (2010)] S. F. Galán, O. J. Mengshoel. *A Novel Mating Approach for Genetic Algorithms*. Versión preliminar enviada a "Information Sciencies Journal" (finalmente no fué aceptada para publicación), 2010

[Goldberg (1989)] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[Holland (1973)] J. H. Holland. Genetic algorithms and the optimal allocation of trials. *SIAM J. of Computing*. 2 pp. 88-105, 1973.

[Holland (1992)] J. H. Holland. *Adaption in Natural and Artificial Systems*. MIT Press, Cambridge, MA, 1992. Primera edición: 1975, The University of Michigan Press, Ann Arbor.

[Huang (2001)] C. F. Huang. *An analysis of mate selection in genetic algorithms*. Technical Report CSCS-2001-2002, Center for the Study of Complex Systems, University of Michigan, 2001.

[Koza (1994)] J. R. Koza. *Genetic Programming II*. MIT Press, Cambridge, MA, 1994.

[Matsui (1999)] K. Matsui. *New selection method to improve the population diversity in genetic algorithms*. En: Proceedings of the 1999 IEEE International Conference on Systems, Man, and Cybernetics (IEEE SMC'99), vol. 1, pp. 625-630, Tokyo, Japón, 1999.

[Michalewicz (1996)] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, Berlin, Heidelberg, New York, 3rd edn., 1996.

[Mühlenbein, Schomisch y Born (1991)] H. Mühlenbein, D. Schomisch, J. Born. *The parallel genetic algorithm as function optimizer*. Parallel Computing, 17:619-632. 1991.

[Ochoa (2005)] G. Ochoa, C. Mädler-Kron, R. Rodríguez, K. Jaffe. *Assortative mating in genetic algorithms for dynamic problems*. En: F. Rothlauf, J. Branke, S. Cagnoni, D. W.

Corne, R. Drechsler, Y. Jin, P. Machado, E. Marchiori, J. Romero, G. D. Smith, G. Squillero, editores, Applications of Evolutionary Computing, vol. 3449, pp. 617-622. Springer, 2005.

[Rechenberg (1973)] I. Rechenberg. *Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution*. Fromman-Holzboog, Stuttgart, 1973.

[Ronald (1995)] E. Ronald. *When selection meets seduction*. En: Proceedings of the 6th International Conference on Genetic Algorithms (ICGA'95), pp. 167-173, Pittsburgh, PA. Morgan Kaufmann, San Francisco, CA, 1995

[Schwefel (1981)] H. P. Schwefel. *Numerical Optimization of Computer Models*, volume 26. Wiley, Chichester, UK, 1981

[Schwefel (1995)] H. P. Schwefel. *Evolution and Optimum Seeking*. Wiley, New York, 1995.

[Smith (2000)] R. E. Smith, C. Bonacina, P. Kearney, W. Merlat. *Embodiment of evolutionary computation in general agents*. Evolutionary Computation, 8(4): 475-493, 2000.

[Smith y Bonacina (2003)] R. E. Smith, C. Bonacina. *Mating restriction and niching pressure: results from agents and implications for general evolutionary computation*. En: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2003), pp. 1382-1393, Chicago, IL. Springer, 2003.

[Ting (2003)] C. K. Ting, S. T. Li, C. Lee. *On the harmonious mating strategy through tabu search*. Information Sciencies: an International Journal, 156(3-4): 189-214, 2003.

[Unemi y Nagayoshi (1997)] T. Unemi, M. Nagayoshi. *Evolution of learning robot team via local mating strategy*. En: Fourth European Conference on Artificial Life, Brighton, UK. Poster Session. 1997.

ANEXO I

S. F. Galán, O. J. Mengshoel. *A Novel Mating Approach for Genetic Algorithms*. Versión preliminar enviada a “Information Sciencies Journal” (finalmente no fué aceptada para publicación), 2010

A Novel Mating Approach for Genetic Algorithms

Severino F. Galán
Artificial Intelligence Dept.
National Distance University of Spain (UNED)
C/ Juan del Rosal, 16
28040 Madrid, Spain
seve@dia.uned.es

Ole J. Mengshoel
Carnegie Mellon University
NASA-Ames Research Center
Mail Stop 269-3, P.O. Box 1
Moffett Field, CA 94035, USA
ole.mengshoel@sv.cmu.edu

Abstract

Genetic algorithms typically use crossover, which relies on mating a set of selected parents. As part of crossover, random mating is often carried out. A novel approach to parent mating is presented in this work. The novel approach can be applied in combination with a traditional similarity-based criterion to measure distance between individuals or with a fitness-based criterion. We introduce a parameter called *mating index* that allows different mating strategies to be developed within a uniform framework: from an exploitative strategy called Best-First to an explorative one called Best-Last. Self-Adaptive mating is a novel mating strategy defined in the context of the novel algorithm, whose goal is to achieve a balance between exploitation and exploration in a domain-independent manner. The present work consists of formally defining the novel mating approach, analyzing its behavior, and developing an extensive experimental study to quantitatively determine its benefits. In the domain of real function optimization, the experiments show that, as the degree of multimodality of the function at hand grows, it is convenient to increase the mating index in order to obtain good performance. In the case of Self-Adaptive mating strategy, the experiments give good results for a significant set of the studied cases.

Keywords: Genetic algorithms, premature convergence, mating strategies, mating index, self-adaptive mating.

1 Introduction

Genetic algorithms (GAs) [10, 9] use stochastic search methods based on natural evolution in order to solve adaptation problems in fields like optimization, design, learning, or scheduling, among others. In a GA, a set of candidate solutions is created each generation. The quality of a solution, its fitness, determines its chance to survive and reproduce. Two processes form the basis of genetic algorithms: variation (recombination and mutation) and selection. While the former facilitates diversity and novelty, the latter favours quality. Ideally, at the end of a running GA, a solution with optimal or near-optimal fitness is found.

Premature convergence to local optima is one of the most frequent difficulties that arise when applying GAs to complex problems. It occurs when genetic operators can no longer generate offspring that are fitter than their suboptimal parents. Premature convergence is associated with the loss of diversity in the population. However, too much population diversity can lead to a dramatic deterioration of GA efficiency. Therefore, an important issue in the design and application of GAs is the trade-off between exploitation of the best individuals and exploration of alternative regions of the search space.

By focusing on the mating phase of GAs, the present work deals with achieving a proper balance between exploitation and exploration. Traditionally, mating takes place after parent selection and prior to recombination. Normally, parents are mated in pairs so that each pair can subsequently be recombined. A key question is how mating should be carried out in order to give strong GA performance. The traditional mating approach consists of selecting a parent's mate uniformly at random from the set of remaining parents. In addition to the traditional random mating approach, other approaches exist that apply mating restriction techniques based on similarity relations between parents [5, 7]. Although these methods have been shown to benefit GA performance, they are costly in computational terms. This disadvantage is due to the fact that similarity comparisons between two parents' chromosomes normally take place at a gene level. Furthermore, these methods were designed for rather specific contexts like *fitness sharing* [5] and *incest prevention* [7] and, therefore, their impact has been quite limited.

The goal of this work is to develop, analyze, and evaluate a novel and general approach to mating in GAs. The novel approach uses a parameter called *mating index* which allows the degree of exploration to be controlled depending on the hardness of the problem being solved. In this way, we hope our approach can easily be applied to a wide variety of problems. In addition, by using fitness-based comparisons between parents, rather than just similarity-based comparisons, the computational complexity of the applied mating algorithm can be reduced. Furthermore, the novel approach lends itself to a self-adaptive algorithm which gives rise to a useful mating strategy.

The domain of real function optimization is used in this work to experimentally study the benefits of the new approach. We anticipate here that:

- The main parameters of the new approach, *mating size* and *mating in-*

dex, have a strong influence on GA performance. This influence heavily depends on the shape of the function being optimized.

- The degree of multimodality for the function at hand determines which specific mating index produces best performance. In general, the higher the degree of multimodality is, the higher the mating index should be.
- From a qualitative point of view, fitness-based mating produces analogous results to those of similarity-based mating. However, fitness-based mating needs less computation time; for example, in the experiments summarized in Table 3 the mean computation time for fitness-based mating is on average 25 times shorter than the mean computation time for similarity-based mating.
- Experiments with SELF-ADAPTIVE mating strategy give good results for a significant set of the studied cases.
- ANOVA tests have been performed in order to determine the statistical significance of the experimental results obtained in this work.

The rest of this paper is structured as follows. Section 2 reviews previous work on restricted mating in GAs. Section 3 introduces our novel approach to mating, along with the different mating strategies derived from it. Section 4 analyzes the mating approach introduced in this work. Section 5 includes an extensive empirical evaluation of the novel mating strategies. A discussion of our approach to mating in GAs is made in Section 6. Finally, Section 7 contains the main conclusions derived from the present work.

2 Restricted Mating in Genetic Algorithms

The usual way of mating parents in GAs consists of taking a parent from the mating pool and selecting its mate by choosing uniformly at random one of the remaining parents. The mated parents are then removed from the mating pool, and the same process is repeated until all the individuals have been mated. Restricted mating techniques, which do not select a mate uniformly at random, have been successfully developed for specific contexts such as *fitness sharing* and *incest prevention*. Other approaches that incorporate mating preferences into evolutionary systems are: *assortative mating genetic algorithms* [8, 11, 16], *correlative tournament selection* [14], *seduction* [17], *tabu genetic algorithm* [22], and *evolving agents* [20, 19, 23].

Fitness sharing [5] is a method that forces the population to maintain different niches. In multimodal optimization problems, where a number of high-fitness individuals corresponding to various local optima are identified, niches are search space regions around local optima, and high-fitness niches are of particular interest. Fitness sharing adjusts fitnesses of individuals prior to parent selection, so that individuals are allocated to niches in proportion to the niches fitness. In order to improve the efficiency of fitness sharing, Deb and Goldberg

[5] used a restricted mating approach whose goal was to avoid the creation of lethal (low fitness) individuals. Once niches are formed in the population, the recombination of two parents from different niches is likely to form lethal offspring. Therefore, restricted mating among individuals of the same niche is promoted. This is achieved by following the same scheme as random mating but, given a parent, a candidate mate is accepted only if the phenotype/genotype distance between them is smaller than a given threshold. Otherwise, another candidate is sought. If no candidate is accepted, one is chosen uniformly at random as in random mating. In the case of fitness sharing for real functions optimization, the phenotype space corresponds to the real values of the variables, while the genotype space uses a binary representation for them. If similarity is measured within the phenotypic space, Euclidean distance is used. Hamming distance is employed when similarity between individuals is measured within the genotypic space. Other GA approaches applying restricted mating in the specific context of multimodal optimization problems are: *island models* [3, 13], *diffusion models* [12, 24], and *automatic speciation models* [1, 21].

In contrast to fitness sharing, incest prevention [7] was defined in the context of global optimization rather than niching. Incest prevention promotes restricted mating between dissimilar enough individuals. In general, when two similar individuals are mated, their offspring may not introduce significant new information about the search space, which provokes a reduction in the performance of the GA. Incest prevention follows a dual scheme to that used in fitness sharing: A candidate mate is accepted only if its phenotype/genotype distance to the current parent is greater than a given threshold. Usually, this threshold is reduced when better offspring is not obtained during the search process.

In comparison to random mating, similarity-based restricted mating was shown to produce a more effective exploration of the search space both in fitness sharing [5] and in incest prevention [7]. However, these similarity-based approaches are in some sense the polar opposites, and in this work we develop a uniform framework for the similarity-based approach, thus improving the understanding of it. At the same time, the time cost associated with measuring the distances between individuals is an important disadvantage of the similarity-based approach, and may be prohibitive. This work explores fitness-based mating as an alternative for establishing mating preferences with a lower computational cost. Although fitness-based restricted mating was addressed in [4, 2], this technique has not been sufficiently investigated in the past, due to the widespread use of similarity in the definition of mating approaches. One of the goals of this work is to thoroughly compare fitness-based mating strategies with their similarity-based counterparts.

The present work aims at formalizing a general mating approach which allows a wide range of mating strategies to be defined and effectively applied to the task of global optimization in GAs. Finally, a self-adaptive mating method is developed.

3 The New Mating Approach

This section presents a novel approach to mating in GAs. Our novel approach has three main characteristics. Firstly, a parameter named *mating index* allows the degree of exploration to be controlled in a simple way, which makes our approach flexible and general. Secondly, it allows mating preferences to be defined either in terms of similarity between individuals or in terms of fitness of individuals, in contrast to most of the mating strategies reviewed in Section 2, which are typically based on similarity between individuals. Thirdly, the novel approach lends itself to a self-adaptive implementation in which each individual in the population has its own mating preference; in this way, different mating strategies can be applied depending on the hardness of the fitness function and the current state of the search process.

The novel approach is defined by the following algorithm, which constitutes a GA mating phase, taking place between parent selection and parent recombination.

Algorithm 1 *(Mating step). Novel mating approach for GAs:*

Input:

P_s	*Population of selected parents*
γ	*Mating Size: number of eligible parents for next mating*
cr	*Criterion used for defining mating preferences* ($cr \in \{\mathsf{similarity}, \mathsf{fitness}\}$)
α	*Mating Index: integer used for defining a mating preference* ($2 \leq \alpha \leq \gamma$)

Output:

P_m	*Population of mated parents*

1. *Choose γ parents uniformly at random from P_s without replacement. The set of chosen parents is denoted by Ch.*

2. *Let p_1 be the parent in Ch with highest fitness. Remove p_1 from P_s and Ch, and include it in P_m. Set Ch now includes the candidate mates for p_1.*

3. *Order the $\alpha - 1$ best candidates in Ch under criterion cr. When $cr =$ similarity, candidates are ranked according to phenotype similarity with p_1. When $cr =$ fitness, candidates with higher fitness are ranked first. Let $B_{\alpha-1}(p_1) = \{b_2, b_3, \ldots, b_{\alpha-1}, b_\alpha\}$ be the set of $\alpha-1$ ordered best candidates obtained under criterion cr.*

4. *Choose $p_2 = b_\alpha$ as mate for p_1, remove p_2 from P_s, and include p_2 in P_m. It should be noted that p_1 and p_2 are placed in contiguous positions in P_m.*

5. *Go back to step 1 if P_s is not empty yet.*

In Algorithm 1, γ (mating size) different parents are randomly chosen for the next round of mating, and the fittest of them is mated with another individual as determined by *cr* (mating criterion) and α (mating index). Similarity in the phenotype space is the traditional criterion used to establish mating preferences in GAs [5, 7]. This is why similarity has been included in the domain of *cr* in Algorithm 1. Due to the computational complexity of similarity comparisons, a new fitness-based criterion for establishing mating preferences is also introduced. While determining the similarity of two individuals requires examining their chromosomes gene by gene, comparing their fitnesses involves examining only two numbers.

Note that the novel mating approach defined by Algorithm 1 becomes the traditional mating approach when $\gamma = 2$, since parents are then mated uniformly at random. Therefore, the novel mating approach is a generalization of the traditional approach. The values of mating index α induce different mating strategies corresponding to a wide range of degrees of exploitation versus exploration.

In Algorithm 1, a specific α value defines a mating strategy that we name BEST-$(\alpha - 1)$TH, since the best parent is mated with the $(\alpha - 1)$-th mating candidate. Several variants of the general scheme in Algorithm 1 are possible. This work focuses on three of them:

1. When $\alpha = 2$, the best parent is mated with the *first* mating candidate under criterion *cr*. Thus, the resulting scheme is called BEST-FIRST mating strategy.

2. When $\alpha = \gamma$, the best parent is mated with the *last* mating candidate under criterion *cr*. This strategy is called BEST-LAST mating.

3. When the parameter α is made local to each individual, encoded into the chromosome, and subjected to recombination and mutation, a SELF-ADAPTIVE mating strategy results.

In the rest of this section, these variants are discussed in more detail.

3.1 Best-First Mating

Exploitation of the best solutions in the current population can be achieved by setting $\alpha = 2$ in Algorithm 1. In this way, the fittest of the chosen parents, p_1, is mated with the first of the candidates under criterion *cr*. If a fitness-based criterion is used, p_1's mating preference is clearly an exploitative strategy, since fitter candidates are preferred over the rest. If a similarity-based criterion is used, p_1's mating preference is exploitative as well, since it is implicitly assumed that fitter candidates are more similar to p_1 than the rest.

In step 3 of Algorithm 1, an ordering of the first $\alpha - 1$ candidates in *Ch* under criterion *cr* is in fact not necessary in BEST-FIRST mating. Just the first of the candidates under criterion *cr* is sought. Thus, only a variable storing the currently first candidate is needed to implement BEST-FIRST. Steps 3 and 4 of Algorithm 1 can be designed as follows:

Algorithm 2 *(*Best-First *mate election).*

Input:
- p_1 *Individual to be mated*
- Ch *Candidate mates for* p_1
- cr *Criterion used for defining mating preferences* ($cr \in \{$*similarity with respect to* p_1, *fitness*$\}$)
- P_s *Population of selected parents*
- P_m *Population of mated parents*

Output:
- p_2 *Mate for* p_1

1. $p_2 \leftarrow$ *First candidate in Ch under criterion cr*
2. *Remove* p_2 *from* P_s
3. *Include* p_2 *in* P_m

Best-First mating with a similarity-based criterion is inspired by the mating strategy used by Deb and Goldberg [5] in the context of fitness sharing. Whereas Deb and Goldberg used a similarity threshold to guide the mating process within niches, Best-First mating employs a mating size parameter in order to obtain a certain degree of exploitation. Similarity-based Best-First mating is also similar to *positive assortative mating* [8, 11, 16], which chooses the most similar candidate as mate for the current individual. At the same time, Best-First mating with a fitness-based criterion has common characteristics with some of the mating methods developed in [4, 2].

3.2 Best-Last Mating

Exploration of alternative solutions to the best ones in the current population can be performed by setting $\alpha = \gamma$ in Algorithm 1. By doing that, the fittest of the chosen parents, p_1, is mated with the last of the candidates under criterion cr. If a fitness-based criterion is used, p_1's mating preference is clearly an explorative strategy, since the fittest parent prefers less fit candidates over the rest. If a similarity-based criterion is used, p_1's mating preference is explorative as well, since the most distant candidate in the phenotype space is chosen for mating.

In step 3 of Algorithm 1, an ordering of the first $\alpha - 1$ candidates in Ch under criterion cr is not necessary in Best-Last mating. Only the last of the candidates under such a criterion is sought. Therefore, a unique variable storing the currently last candidate is needed to implement Best-Last. Steps 3 and 4 of Algorithm 1 can be implemented as follows:

Algorithm 3 *(*Best-Last *mate election).*

Input:
- p_1 *Individual to be mated*

Ch *Candidate mates for* p_1
cr *Criterion used for defining mating preferences*
($cr \in \{$*similarity with respect to* p_1, *fitness*$\}$)
P_s *Population of selected parents*
P_m *Population of mated parents*
Output:
p_2 *Mate for* p_1

1. $p_2 \leftarrow$ *Last candidate in Ch under criterion cr*
2. *Remove* p_2 *from* P_s
3. *Include* p_2 *in* P_m

Best-Last mating with a similarity-based criterion is inspired by the mating strategy used by Eshelman and Schaffer [7] for incest prevention. While Eshelman and Schaffer used a similarity threshold to prevent incest, Best-Last mating achieves a particular degree of exploration by setting the mating size value. Similarity-based Best-Last mating is also similar to *negative assortative mating* [8, 11, 16], which chooses the most dissimilar candidate as mate for the current individual. On the other hand, to the best of our knowledge Best-Last mating with a fitness-based criterion has not yet been investigated in the literature.

3.3 Self-Adaptive Mating

A GA's parameters can either be manually tuned in advance or automatically controlled during execution. Compared to manual parameter tuning, advantages of automatic parameter control are that (i) it is less taxing on the GA's user and (ii) parameters can be adapted to the state of the search process. A classification of parameter setting techniques for evolutionary algorithms can be found in [6, Chapter 8]. This section deals with self-adaptive control of mating parameters. Self-adaptive parameter control consists of encoding the parameters into the chromosomes and performing recombination and mutation on them. In this way, the values of the parameters leading to better individuals will have a greater chance to survive.

If an individual j is represented as $\langle x_{j,1}, \ldots, x_{j,n} \rangle$, its extended representation under Self-Adaptive mating would be $\langle x_{j,1}, \ldots, x_{j,n}, x_{j,n+1} \rangle$, where $x_{j,n+1} = \alpha_j$ is the mating index for individual j. In other words, the mating index is now a local parameter, and each individual has an independent mating preference. The algorithm performing Self-Adaptive mating can easily be obtained from Algorithm 1 by removing α from the input and substituting $\alpha - 1$ with $x_{p_1,n+1} - 1$ in step 3. In this case, steps 3 and 4 of Algorithm 1 can be designed as follows:

Algorithm 4 *(*Self-Adaptive *mate election).*

Input:

p_1	*Individual to be mated*
Ch	*Candidate mates for* p_1
cr	*Criterion used for defining mating preferences* ($cr \in \{$*similarity with respect to* p_1, *fitness*$\}$)
P_s	*Population of selected parents*
P_m	*Population of mated parents*

Output:

p_2	*Mate for* p_1

1. $p_2 \leftarrow (\alpha_{p_1} - 1)$*-th candidate in Ch under criterion cr*
2. *Remove* p_2 *from* P_s
3. *Include* p_2 *in* P_m

It remains to consider how mating indexes are initialized, recombined, and mutated. As far as initialization is concerned, each mating index is assigned an integer generated uniformly at random from range $[2, \gamma]$. Recombination of the mating indexes of two parents can be carried out in several ways: by assigning to the two children the mean of the parents' mating indexes, or by letting the two children inherit the parents' mating indexes, among other possibilities. This work uses the latter method, since we have found it to produce better experimental results. Mutation of mating indexes is implemented by setting a probability $p_=$ that mating index is unchanged, a probability p_+ that mating index is incremented by one, a probability p_- that mating index is decremented by one, and a probability $1-p_=-p_+-p_-$ that mating index is changed uniformly at random. Values $p_= = 0.5$ and $p_+ = p_- \lesssim 0.25$ were employed, since they led to better performance in the experiments.

3.4 An Example of the Novel Mating Strategies

We now discuss an example illustrating Best-First mating, Best-Last mating, and Self-Adaptive mating. Consider in Algorithm 1 a population of six selected parents, $P_s = \{A, B, C, D, E, F\}$, resulting after parent selection in a GA. The six parents have to be mated before recombination. Figure 1 depicts the parents according to their phenotype (x-axis) and their fitness (y-axis), where it is assumed that there is a bijection between phenotypes of individuals and a certain discretized interval of real numbers.

The random mating strategy (Random) mates parents by choosing a mate uniformly at random among the remaining parents. A possible mating resulting from this strategy is $\{CF, EA, BD\}$. It is important to note that fitness or similarity information is not used at any step of Random mating.

If a mating size $\gamma = 6$ is assumed for simplicity, the Best-First, Best-Last, and Self-Adaptive mating strategies create $Ch = \{A, B, C, D, E, F\}$. The first mate, p_1, is the parent with highest fitness in Ch; $p_1 = F$ in this case. In Best-First, the second mate, p_2, is the first of the candidates in

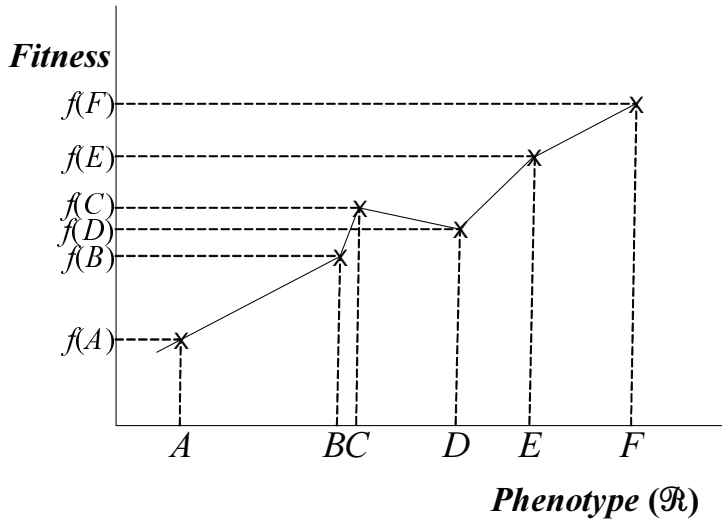

Figure 1: Population of six parents $\{A, B, C, D, E, F\}$ to be mated. These parents belong to a real interval $I \subset \Re$.

$\{A, B, C, D, E\}$ under criterion *cr*. First, we discuss the similarity-based criterion, under which $p_2 = E$. As a result, F and E are mated, and the same process continues until all of the parents have been mated. A new set *Ch* would be formed prior to each pairing between p_1 and p_2. Ultimately, the mating resulting from this strategy is $\{FE, CB, DA\}$. Second, we discuss the fitness-based criterion, under which $p_2 = E$ for $p_1 = F$ like under similarity-based criterion. Next, C is paired with D, instead of with B as in the similarity case. The final pairing is $\{FE, CD, BA\}$.

The Best-Last mating strategy works analogously to Best-First, but now the last of the candidates under criterion *cr* is assigned to p_2 at each iteration of the mating algorithm. In this way, under similarity-based mating, F and A are first mated. In the end, the mating resulting from this strategy is $\{FA, EB, CD\}$. For fitness-based mating, the final pairing is $\{FA, EB, CD\}$ as well.

The Self-Adaptive mating is now considered. The following mating indexes will be assumed: $\{\alpha_A = 5, \alpha_B = 2, \alpha_C = 3, \alpha_D = 3, \alpha_E = 4, \alpha_F = 2\}$. The first mate, p_1, is again the parent with highest fitness in *Ch*; $p_1 = F$ in this case. The second mate, p_2, is the $(\alpha_F - 1)$-th candidate in $\{A, B, C, D, E\}$ under criterion *cr*. First, we discuss the similarity-based criterion, under which $p_2 = E$. As a result, F and E are mated, and the same process continues until all of the parents have been mated. A new set *Ch* is formed prior to each pairing between p_1 and p_2. When the number of candidates for p_1 in *Ch* is smaller than α_{p_1}, the last element in *Ch* under criterion *cr* is selected as mate for p_1. The mating resulting from this strategy is $\{FE, CD, BA\}$. Second, we discuss the fitness-based criterion, under which $p_2 = E$ like under similarity-based crite-

rion. Then, C is paired with B, instead of with D as in similarity-based mating. The final pairing is $\{FE, CB, DA\}$.

It should be noted that BEST-FIRST produces the best potential mating for the simple fitness function in Figure 1, since mating parents with high fitness will favor the creation of children with high fitness with higher probability. However, in more realistic and interesting fitness functions, it is clear that BEST-FIRST is not always optimal, as Section 5 shows.

4 Analysis

The present section analyzes the novel mating approach introduced in this work by making use of an idealized model. The analysis is developed for the two types of domains that can be encountered in function optimization: on the one hand, unimodal fitness functions with a unique optimum and, on the other hand, multimodal fitness functions with many local optima. In both cases, the influence of mating on the effectiveness to reach the global optimum is studied. For simplicity, this section only deals with similarity-based mating. As in Section 3.4, it will be assumed that there is a bijection between phenotypes of individuals and a certain discretized interval of real numbers.

4.1 Analysis for Unimodal Fitness Functions

Consider the maximization problem of the linear fitness function depicted in Figure 2, $f(x) = x$, where x represents individuals' phenotype defined in the range $[x_1, x_2]$ with $x_1 < x_2$. For any other unimodal fitness function, an analogous argument to the following one could be made.

Let $x' \in (x_1, x_2)$ (see Figure 2) denote the fittest individual within set Ch of Algorithm 1; in other words, x' is the best individual resulting from selecting (uniformly at random) γ individuals from the current population of parents. Therefore, x' is the next individual to be assigned a mate x''. Candidate mates for x' can only belong to range $[x_1, x']$, since $f(x') \geq f(x'')$ from Algorithm 1.

Assume that the recombination of two parents, $x_{p_1} \in [x_1, x_2]$ and $x_{p_2} \in [x_1, x_2]$ with $x_{p_1} < x_{p_2}$, is performed in the phenotype space and produces children, $x_{\text{child}(x_{p_1}, x_{p_2})}$, uniformly distributed over $[x_{p_1}, x_{p_2}]$. Consequently, the expected fitness of a child of x_{p_1} and x_{p_2} can be calculated as:

$$\overline{f}(x_{\text{child}(x_{p_1}, x_{p_2})}) = \frac{f(x_{p_1}) + f(x_{p_2})}{2}, \tag{1}$$

since the probability that $x_{\text{child}(x_{p_1}, x_{p_2})} \notin [x_{p_1}, x_{p_2}]$ is equal to zero and f is linear with respect to x. Given that $f(x) = x$, Equation 1 turns into:

$$\overline{f}(x_{\text{child}(x_{p_1}, x_{p_2})}) = \frac{x_{p_1} + x_{p_2}}{2}. \tag{2}$$

From Equation 2, the expected fitness of a child resulting from the recombination

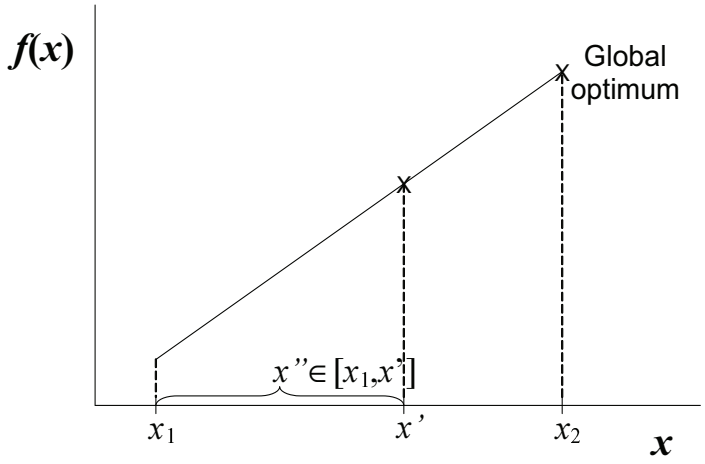

Figure 2: A linear unimodal fitness function with no local optima except for the global optimum. Individual $x' \in [x_1, x_2]$ is mated with individual $x'' \in [x_1, x']$.

of x' and x'', with $x'' \in [x_1, x']$, is:

$$\overline{f}(x_{\text{child}(x',x'')}) = \frac{x' + x''}{2}. \tag{3}$$

It is important to note that $\overline{f}(x_{\text{child}(x',x'')})$ in Equation 3 increases as x'' approaches x' and reaches a maximum when $x'' = x'$.

It can be concluded from the previous result that, in the case of unimodal fitness functions, those mating strategies under the novel approach favoring recombination with similar individuals produce a more effective search for the global optimum.

4.2 Analysis for Multimodal Fitness Functions

Consider the maximization problem of a multimodal fitness function, $g(x)$, where x represents individuals' phenotype defined in the range $[x_1, x_2]$ with $x_1 < x_2$, as depicted in Figure 3. Assume that function $g(x)$ has a global optimum at x_{op}, whose basin of attraction lies in the range $[x_{\text{op}} - b, x_{\text{op}} + b]$. The basin of attraction for a maximum is the set of points in the fitness landscape from which a steepest-ascent hill-climbing search would finish reaching that maximum.

Without loss of generality, consider that individual $x' = x_1$ is to be assigned a mate x''. Assume that function $g(x)$ reaches several local optima between x' and $x_{\text{op}} - b$. Contrarily to Section 4.1, individual x' has been chosen outside the basin of attraction for the global optimum, which is the usual situation for the population individuals when a multimodal fitness function is optimized.

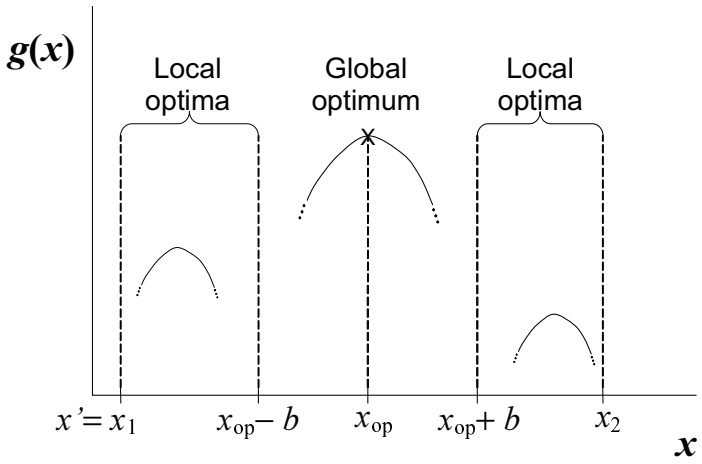

Figure 3: A multimodal fitness function.

The argument in Section 4.1 can be applied to individuals only in the basin of attraction for the global optimum.

As in Section 4.1, assume that the recombination of two parents, $x_{p_1} \in [x_1, x_2]$ and $x_{p_2} \in [x_1, x_2]$ with $x_{p_1} < x_{p_2}$, is performed in the phenotype space and produces children, $x_{\text{child}(x_{p_1}, x_{p_2})}$, uniformly distributed over $[x_{p_1}, x_{p_2}]$. Since the problem of global optimization is to be solved, it is convenient to calculate the probability of reaching the basin of attraction for the global optimum by recombining x' with a candidate mate $x'' \in [x_1, x_2]$. (We are not interested in reaching the regions of the fitness landscape located outside the basin of attraction for the global optimum, since that would lead to premature convergence.) Such a probability is zero for $x'' \in [x_1, x_{\text{op}} - b)$, since $[x', x''] \cap [x_{\text{op}} - b, x_{\text{op}} + b] = \emptyset$. For $x'' \in [x_{\text{op}} - b, x_{\text{op}} + b]$, the probability that the child reaches $[x_{\text{op}} - b, x_{\text{op}} + b]$ is equal to $\frac{x'' - (x_{\text{op}} - b)}{x'' - x'}$. Finally, for $x'' \in (x_{\text{op}} + b, x_2]$, the probability is equal to $\frac{2b}{x'' - x'}$, since $[x', x''] \cap [x_{\text{op}} - b, x_{\text{op}} + b] = [x_{\text{op}} - b, x_{\text{op}} + b]$. In summary, as shown in Figure 4:

$$Pr_{\text{basin}}(x = x'') = \begin{cases} 0 & \text{if} \quad x'' \in [x_1, x_{\text{op}} - b) \\ \frac{x'' - (x_{\text{op}} - b)}{x'' - x'} & \text{if} \quad x'' \in [x_{\text{op}} - b, x_{\text{op}} + b] \\ \frac{2b}{x'' - x'} & \text{if} \quad x'' \in (x_{\text{op}} + b, x_2] \end{cases} .$$

It can be concluded from Figure 4 that, in the case of multimodal fitness functions and individuals outside the basin of attraction for the global optimum, those mating strategies under the novel approach favoring recombination with dissimilar individuals produce a more effective search for the global optimum. While candidate mates in $[x_1, x_{\text{op}} - b)$ lead to local optima, there is always a probability greater than zero that the basin of attraction for the global optimum is reached if candidate mates are taken from $[x_{\text{op}} - b, x_2]$.

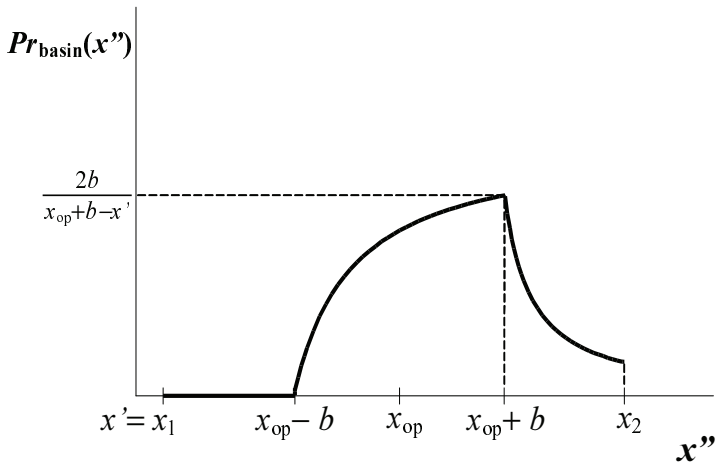

Figure 4: Probability that a child of x' and x'' reaches the basin of attraction for the global optimum $[x_{\text{op}} - b, x_{\text{op}} + b]$.

5 Experiments

The domain studied in the experiments is optimization of real functions. Given an n-dimensional function, f: $\Re^n \longrightarrow \Re$, global optimization consists of determining $x^* \in \Re^n$ such that $f(x^*) \geq f(x)$ $\forall x \neq x^*$ with $x \in \Re^n$. This definition corresponds to a maximization problem. In the case of minimization, the inequality to be considered is $f(x^*) \leq f(x)$. In this section, a discretized real interval is considered for each dimension of the function domain. Each interval point is encoded as a binary string by using a Gray code.

The experiments were performed by means of a simple GA using tournament parent selection with tournament size equal to two, one-point crossover with crossover probability equal to one, bit-flip mutation with mutation probability equal to the inverse of the chromosome length, generational survivor selection, and elitism for the best individual. Different seeds for the random number generator were used for each run of the simple GA. All of the experiments were carried out on a Pentium M processor 760 (2 GHz) running Windows.

5.1 Mating Size Experiments

This section contains a comparative evaluation of the following mating strategies: Random mating, Best-First mating, Best-Last mating, and Self-Adaptive mating. In general, Best-First mating strategy produces exploitation of the best solutions in the current population, Best-Last mating produces exploration of alternative solutions to the best ones in the current population, and Self-Adaptive mating produces a combination of exploration and ex-

ploitation that depends on the shape of the fitness function and the state of the search process.

The rest of this section is structured so that the following comparisons are progressively made:

(a) *unimodal* vs. *multimodal* fitness functions: Two different types of functions were tested, namely Sphere function in Section 5.1.1 and Schwefel function in Section 5.1.2. While Sphere is a unimodal function with just one local optimum (the global optimum), Schwefel is a multimodal function that contains a high number of local optima.

(b) *fitness-based* vs. *similarity-based* mating preferences: Both cases are explored in Section 5.1.1 and Section 5.1.2.

(c) *traditional* (RANDOM) vs. *advanced* (BEST-FIRST, BEST-LAST, and SELF-ADAPTIVE) mating strategies

(d) varying *mating sizes*: The range of explored values is $\gamma \in \{3, 5, 10, 20, 30\}$.

5.1.1 Experiments for Sphere Function

The Sphere function is defined as follows:

$$f_1(x) = \sum_{i=1}^{n} x_i^2.$$

This function has a minimum at $(x_1 = 0, \ldots, x_n = 0)$ whose value is 0. The experiments were designed for $n = 20$, $-10 \leq x_i \leq 10$ $\forall i \in \{1, \ldots, 20\}$, and 10 bits were used to represent each variable; as a result, chromosomes with 200 genes were employed. The population size selected was 100 individuals.

Figure 5 represents the evolution, generation by generation, of the mean best fitness for Sphere function when fitness-based mating strategies are used. One hundred runs were carried out for each experiment. Whereas the RANDOM mating strategy is depicted in the three graphs of Figure 5 for illustrative purposes, the rest of the mating strategies (BEST-FIRST, BEST-LAST, and SELF-ADAPTIVE) are depicted in just one graph. For these three advanced strategies, experiments were performed for different mating size values: $\gamma \in \{3, 5, 10, 20, 30\}$.

The BEST-FIRST mating strategy performs better than the RANDOM strategy, as shown in Figure 5(a). In general, the performance improvement obtained by BEST-FIRST increases with the mating size. This behavior can also be observed in Figure 5(c) for the SELF-ADAPTIVE strategy, although it takes mating size $\gamma = 10$ to begin to see an improvement over the RANDOM strategy. From Figure 5(b), it is clear that the BEST-LAST mating strategy performs worse than the traditional RANDOM strategy in the case of Sphere function. This behavior gets worse as mating size increases.

Figure 6 shows the evolution of the mean best fitness for the Sphere function when similarity-based mating strategies are utilized. In general, the results for

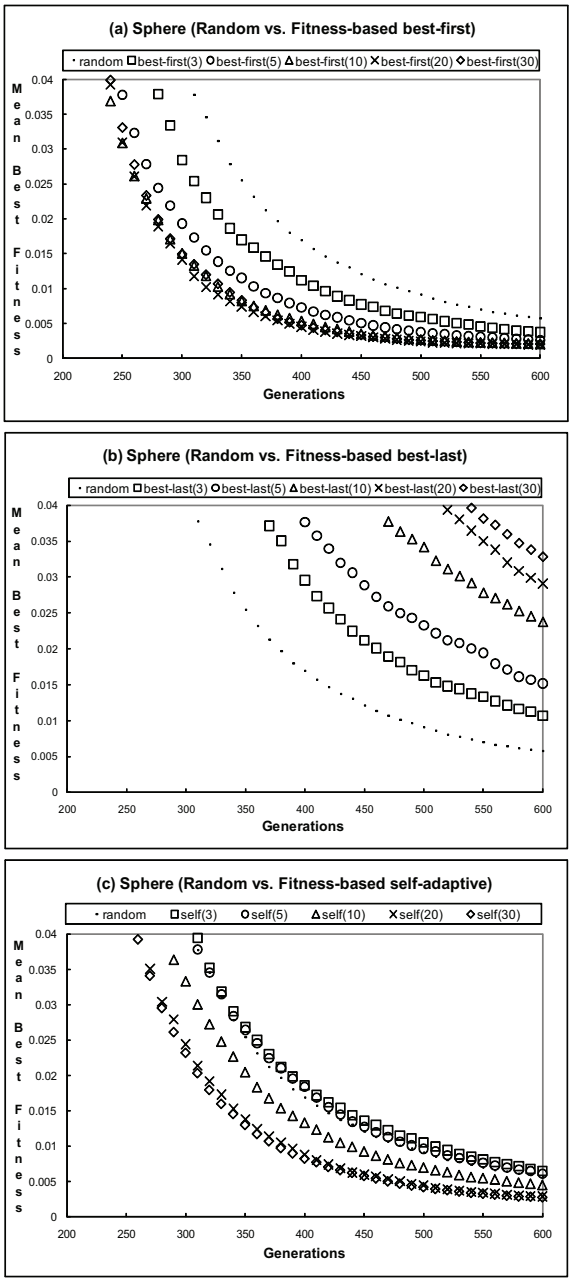

Figure 5: Mean best fitness results for the Sphere minimization problem under RANDOM, BEST-FIRST (a), BEST-LAST (b), and SELF-ADAPTIVE (c) fitness-based mating strategies. For the advanced strategies, mating size is varied from 3 to 30.

similarity-based mating follow a similar pattern to those in Figure 5 for fitness-based mating. However, in the case of similarity-based BEST-FIRST strategy, more generations are needed in order to outperform RANDOM mating, as shown in Figure 6(a). This is a disadvantage of similarity-based BEST-FIRST compared to fitness-based BEST-FIRST in the case of Sphere function.

5.1.2 Experiments for Schwefel Function

The Schwefel function [18, pages 292-293] is defined in the following way:

$$f_2(x) = \sum_{i=1}^{n} x_i \cdot \sin\left(\sqrt{|x_i|}\right).$$

This function has a maximum at $(x_1 = 420.9687, \ldots, x_n = 420.9687)$ whose value is $n \cdot 418.9829$. The experiments were designed for $n = 10$, $-500 \leq x_i \leq 500$ $\forall i \in \{1, \ldots, 10\}$, and 100 bits were used to represent each variable; consequently, chromosomes with 1000 genes were created. The population size was 100 individuals. Due to the complexity of Schwefel function, five hundred runs were performed for each experiment.

Figure 7 depicts the evolution of the mean best fitness for the Schwefel function and fitness-based mating strategies. The opposite performance to that of the Sphere function is to some extent obtained for BEST-LAST and BEST-FIRST versus RANDOM mating. Firstly, Figure 7(b) shows that BEST-LAST mating strategy performs better than the RANDOM strategy in the case of Schwefel function. In general, the performance improvement obtained by BEST-LAST increases with the mating size. Secondly, BEST-FIRST mating strategy performs worse than the RANDOM strategy, as shown in Figure 7(a). This behavior gets worse as mating size increases. On the other hand, contrarily to the case of Sphere function shown in Figure 5(c), fitness-based SELF-ADAPTIVE mating for Schwefel function behaves worse as mating size grows. As depicted in Figure 7(c), although $\gamma \in \{3, 5\}$ outperforms RANDOM mating, that is not the case for $\gamma \in \{10, 20, 30\}$.

Figure 8 contains the evolution of the mean best fitness for Schwefel function when similarity-based mating strategies are used. In general, the results for similarity-based mating follow a similar pattern to those in Figure 7 for fitness-based mating. However, in the case of similarity-based strategies, the outperformances of both BEST-LAST and SELF-ADAPTIVE strategies with respect to RANDOM strategy are superior to those obtained from their fitness-based counterparts. This represents an advantage of similarity-based strategies over fitness-based strategies in the case of Schwefel function.

5.2 Mating Index Experiments

The present section empirically compares RANDOM mating and BEST-αTH mating. In general, the greather α is, the higher the degree of exploration is. The following comparisons are progressively made throughout the rest of this section:

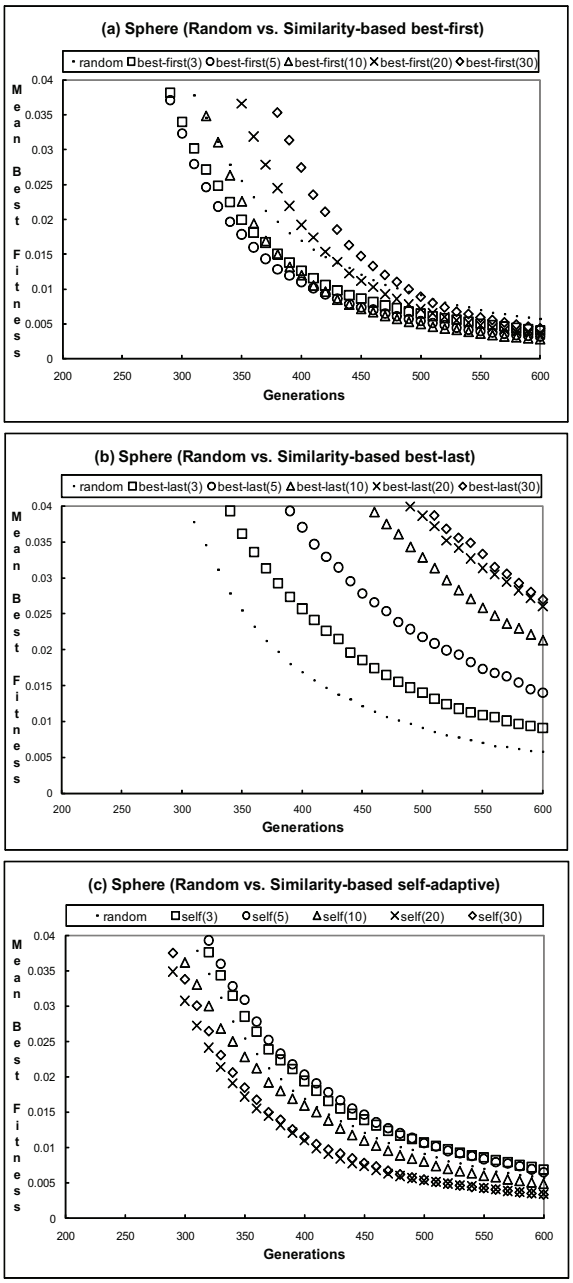

Figure 6: Mean best fitness results for the Sphere minimization problem under RANDOM, BEST-FIRST (a), BEST-LAST (b), and SELF-ADAPTIVE (c) similarity-based mating strategies. For the advanced strategies, mating size is varied from 3 to 30.

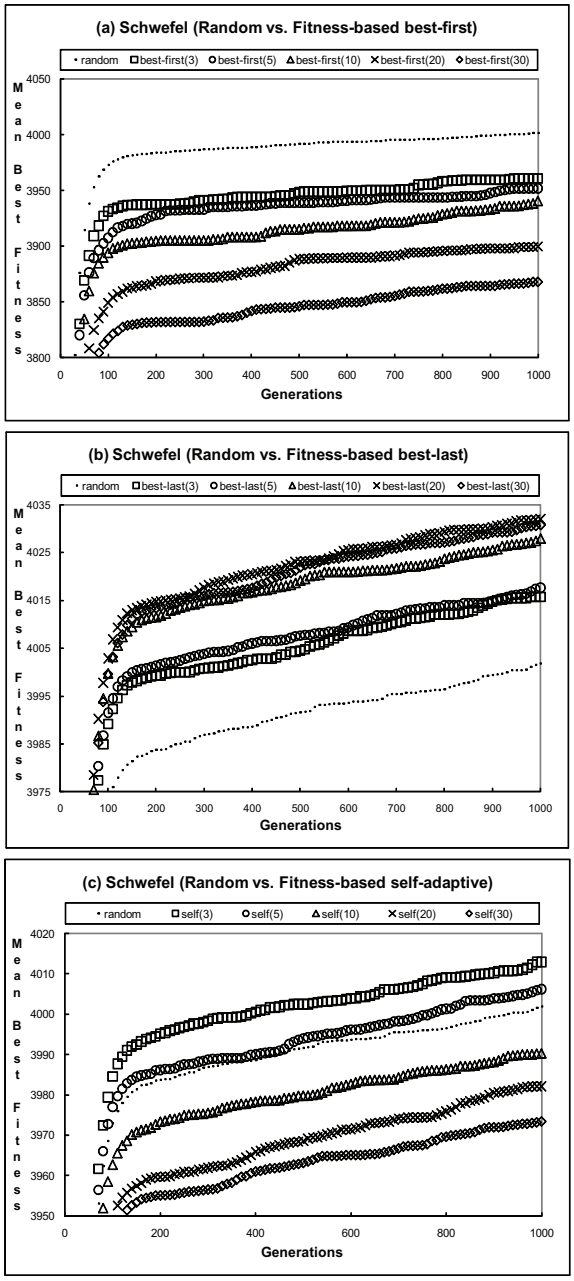

Figure 7: Mean best fitness results for the Schwefel maximization problem under RANDOM, BEST-FIRST (a), BEST-LAST (b), and SELF-ADAPTIVE (c) fitness-based mating strategies. For the advanced strategies, mating size is varied from 3 to 30.

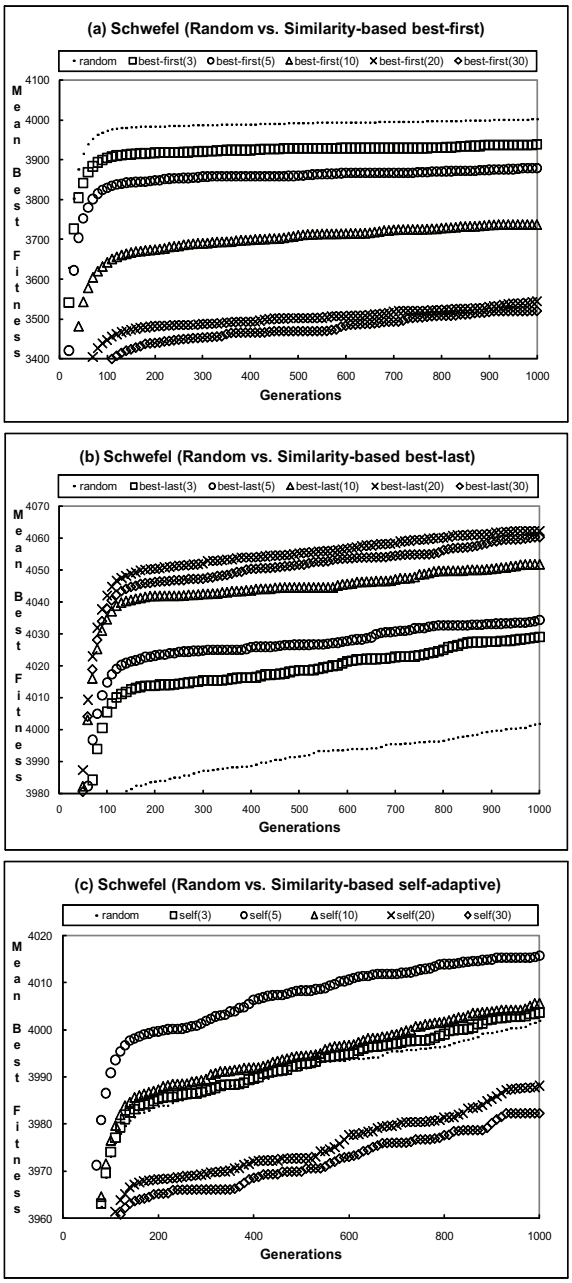

Figure 8: Mean best fitness results for the Schwefel maximization problem under RANDOM, BEST-FIRST (a), BEST-LAST (b), and SELF-ADAPTIVE (c) similarity-based mating strategies. For the advanced strategies, mating size is varied from 3 to 30.

Parameter / **Function**	**Sphere**	**Rastrigin**	**Schwefel**
Dimensions	10	5	5
Bits per Dimension	10	10	20
Chromosome Length	100	50	100
Runs	100	250 for similarity criterion 500 for fitness criterion	
Generations	400	400	250
Individuals	50	50	50

Table 1: Parameters used in the experiments of Section 5.2.1.

(a) *unimodal* vs. *multimodal* fitness functions: Besides Sphere and Schwefel functions, Rastrigin function is now evaluated. Regarding multimodality, Rastrigin function lies between Sphere and Schwefel functions.

(b) *fitness-based* vs. *similarity-based* mating

(c) *traditional* (RANDOM) vs. *advanced* (BEST-αTH) mating strategies

(d) varying *mating indexes*: The range of explored values is $\alpha \in \{2, 5, 10, 15, 20\}$ for a constant mating size of value $\gamma = 20$.

5.2.1 Experiments for Sphere, Rastrigin, and Schwefel Functions

Rastrigin function [15] is defined as follows:

$$f_3(x) = 10 \cdot n + \sum_{i=1}^{n} \left(x_i^2 - 10 \cdot \cos(2 \cdot \pi \cdot x_i)\right).$$

This function has a minimum at $(x_1 = 0, \ldots, x_n = 0)$ whose value is 0. Rastrigin function is based on Sphere function with the addition of cosine modulation to produce many local optima; therefore, in terms of multimodality Rastrigin function is more complex than Sphere function and less complex than Schwefel function[1]. The experiments were designed for $n = 5$, $-5.12 \leq x_i \leq 5.12$ $\forall i \in \{1, \ldots, 5\}$, and 10 bits were used to represent each variable; in this way, chromosomes with 50 genes were generated. Table 1 summarizes the parameters used in these experiments for the three functions considered: Sphere, Rastrigin, and Schwefel.

Figure 9 illustrates the evolution of the mean best fitness for the three functions under both similarity-based and fitness-based criterion. For BEST-αTH strategies, the mating size is kept constant ($\gamma = 20$), while the mating index is assigned values $\alpha \in \{2, 5, 10, 15, 20\}$.

The following results can be derived from Figure 9:

[1]In Schwefel function, the global maximum is distant from the next best local maximum, which makes convergence to frequently take place in the wrong direction.

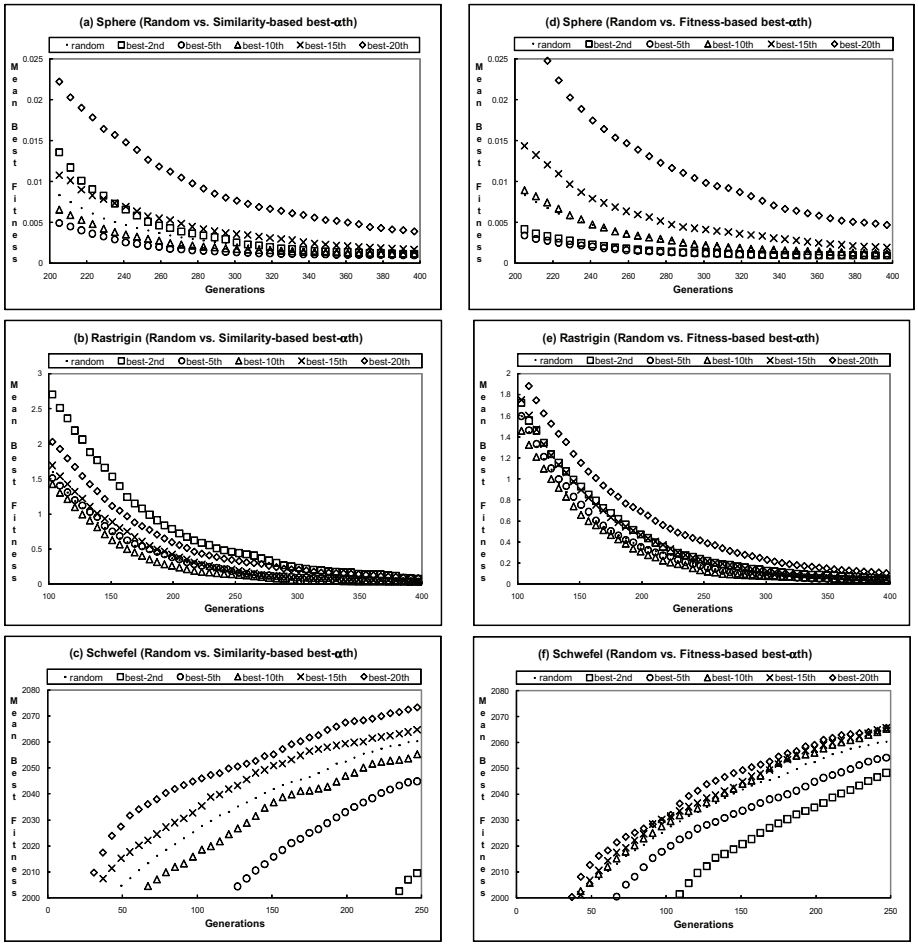

Figure 9: Mean best fitness results for the Sphere, Rastrigin, and Schwefel optimization problems under RANDOM and (similarity-based and fitness-based) BEST-αTH mating strategies. For the advanced strategy, mating size is equal to 20 and mating index is varied from 2 to 20.

1. As depicted in Figure 9(a), lower mating indexes produce better performance in the case of Sphere function, where RANDOM mating is outperformed by BEST-αTH when $\alpha \leq 10$.

2. Figure 9(c) shows that higher mating indexes lead to better performance for Schwefel function, and RANDOM mating is outperformed by BEST-αTH when $\alpha > 10$. This is the opposite behavior to that observed for Sphere function.

3. Figure 9(b) includes an interesting result. While the middle mating indexes ($\alpha \in \{5, 10\}$) outperform RANDOM mating, both the lowest ($\alpha \in \{2\}$) and the highest ($\alpha \in \{15, 20\}$) mating indexes suffer from a poorer performance than that of RANDOM mating. In other words, in the case of Rastrigin function as defined in this section, neither a pure explorative estrategy ($\alpha \sim \gamma$) nor a pure exploitative one ($\alpha \sim 2$) are good options, and the mating index parameter provides us with intermediate strategies ($\alpha \sim \frac{\gamma}{2}$) leading to the best performance. This is a novel result from this work.

4. Figures 9(d), 9(e), and 9(f) for fitness-based strategies show qualitatively similar results to those just described for similarity-based strategies in points 1-3.

6 Discussion

Before discussing the experimental results obtained in Section 5, the statistical significance of such results needs to be established. In other words, for each figure in Section 5 (from Figure 5(a) to Figure 9(f)), it has to be determined whether or not the six graphs in the figure come from the same distribution and, therefore, the differences are due to random effects (null hypothesis). In order to do that, ANOVA tests were performed on the best-fitness data for the last generation depicted in each of the six graphs contained in Figures 5(a) through 9(f). For each graph, one hundred samples were taken. Table 2 includes the results for ANOVA tests on groups of samples constituted by mean best fitness values under $\{\text{RANDOM}, \gamma = 3, \gamma = 5, \gamma = 10, \gamma = 20, \gamma = 30\}$ mating strategies in Section 5.1 (Figures 5(a) through 8(c)) and $\{\text{RANDOM}, \alpha = 2, \alpha = 5, \alpha = 10, \alpha = 15, \alpha = 20\}$ mating strategies in Section 5.2 (Figures 9(a) through 9(f)). Besides F value, probability p of the corresponding result assuming the null hypothesis is shown. The values for p in Table 2 can be considered as significant enough to discard the null hypothesis for the data obtained, suggesting that mating has an important influence on GA performance.

6.1 Discussion on Mating Size Experiments

The experimental results obtained in Section 5.1 for BEST-FIRST, BEST-LAST, and SELF-ADAPTIVE mating strategies suggest that BEST-FIRST mating is the

Figure	Function	F	F_{critical}	p
5(a)	Sphere	148.367	2.229	4.704E−102
5(b)	Sphere	95.771	2.229	6.4498E−74
5(c)	Sphere	88.373	2.229	2.019E−69
6(a)	Sphere	10.373	2.229	1.4855E−9
6(b)	Sphere	40.309	2.229	1.0403E−35
6(c)	Sphere	5.112	2.229	1.3378E−4
7(a)	Schwefel	10.399	2.229	1.4042E−9
7(b)	Schwefel	2.367	2.229	0.03838
7(c)	Schwefel	3.128	2.229	0.00849
8(a)	Schwefel	80.781	2.229	1.2221E−64
8(b)	Schwefel	2.929	2.229	0.01269
8(c)	Schwefel	2.505	2.229	0.02935
9(a)	Sphere	81.266	2.229	5.9728E−65
9(b)	Rastrigin	2.273	2.229	0.04596
9(c)	Schwefel	12.158	2.229	3.1114E−11
9(d)	Sphere	90.772	2.229	6.7666E−71
9(e)	Rastrigin	2.784	2.229	0.01695
9(f)	Schwefel	3.933	2.229	0.00162

Table 2: ANOVA test results for the experiments in Section 5. When $F \geq F_{\text{critical}}$, the result is significant at the level of probability $p = 0.05$.

best option for unimodal problems, while Best-Last mating is the best option for highly multimodal problems. When the degree of multimodality is unknown, the Self-Adaptive mating approach behaves differently depending on the problem at hand. In unimodal problems, Self-Adaptive mating clearly outperforms Random mating as γ increases. In multimodal problems, Self-Adaptive mating produces better results than Random mating for middle and low γ values. For mating size values in the range $\gamma \in [3, 10]$, Self-Adaptive mating performs better than (or at least comparably to) Random mating over the experiments in Section 5.1. However, for multimodal problems, Self-Adaptive mating does not offer a good robust behavior for all the mating sizes.

An example of how Self-Adaptive mating works is shown in Figure 10. This figure depicts the mean α value in the population, generation by generation, for the two cases in which Self-Adaptive strategy had the best behavior in Section 5.1: fitness-based mating for Sphere function and similarity-based mating for Schwefel function. The results are averaged over one hundred runs for Sphere function and over five hundred runs for Schwefel function. In both cases, the mating size was assigned value $\gamma = 20$. Only the results for the first one hundred generations are depicted, since this is the range in which significant changes are obtained for the mean mating index. From Figure 10, the population mean mating index value in the initial population (approximately $\alpha = 11$) decreases rapidly for Sphere function. This is in accordance with

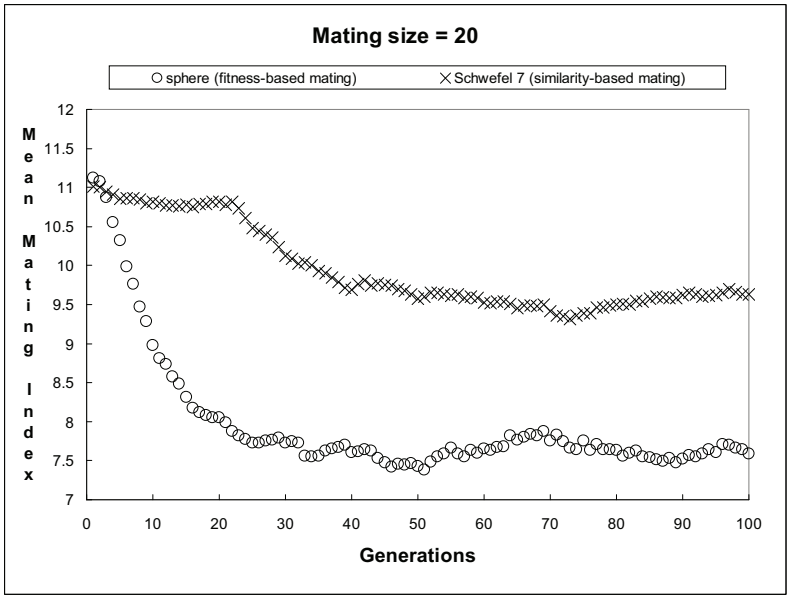

Figure 10: Mean α value in the population for Sphere and Schwefel functions under SELF-ADAPTIVE strategy with $\gamma = 20$.

the experimental results obtained in Section 5.1.1, which showed that BEST-FIRST (where α is small) produces the best results for Sphere function. On the other hand, the mean mating index for Schwefel function is greater than that for Sphere function throughout the generations. This is what should be expected taking into account that, from Section 5.1.2, BEST-LAST (where α is large) produces the best results for Schwefel function. However, due to the fact that individuals with high mating index may produce lethal individuals after recombination in the case of Schwefel function, the graph for this function is not as close to $\alpha = 20$ as the graph for Sphere function is to $\alpha = 2$. This explains the results obtained in Section 5.1, in which SELF-ADAPTIVE mating applied to Sphere function produced similar results to those obtained through BEST-FIRST strategy, while SELF-ADAPTIVE mating applied to Schwefel function could not reach the good results produced by BEST-LAST strategy.

While fitness-based strategies produce better results than similarity-based strategies for unimodal problems like Sphere function optimization, similarity-based strategies outperform fitness-based strategies in the case of multimodal problems like Schwefel function optimization. However, an important advantage of fitness-based strategies is that they lead to substantial computation time savings as shown in Table 3. The differences in computation times between fitness-based and similarity-based mating derive from the operations described in Step 3 of Algorithm 1.

		Sphere			Schwefel		
		t_f	t_s	t_s/t_f	t_f	t_s	t_s/t_f
Best-First	$\gamma = 3$	7.37	22.05	2.99	37.94	111.32	2.93
	$\gamma = 5$	7.23	36.85	5.1	38.39	198.82	5.18
	$\gamma = 10$	7.46	67.15	9	38.79	369.9	9.54
	$\gamma = 20$	7.62	115.94	15.21	38.74	680.56	17.57
	$\gamma = 30$	7.45	154.87	20.79	38.29	944.99	24.68
Best-Last	$\gamma = 3$	7.5	21	2.8	38.24	123.79	3.24
	$\gamma = 5$	7.42	36.86	4.97	37.98	203.7	5.36
	$\gamma = 10$	7.57	66.16	8.74	37.86	389.39	10.28
	$\gamma = 20$	7.63	124.88	16.37	35.88	687.93	19.17
	$\gamma = 30$	7.87	166.38	21.14	36.33	956.08	26.32
Self-Adaptive	$\gamma = 3$	8.81	26.86	3.05	45.32	129.42	2.86
	$\gamma = 5$	8.2	57.82	7.05	43.8	298.66	6.82
	$\gamma = 10$	8.89	203.54	22.89	43.05	1056.43	24.54
	$\gamma = 20$	9.06	675.39	74.55	43.91	3636.85	82.82
	$\gamma = 30$	9.38	1304.7	139.09	44.31	7235.56	163.29

Table 3: Mean computation times (in seconds) for each run of the experiments in Section 5.1, with t_f and t_s referring to fitness-based and similarity-based mating respectively. The number of function evaluations is the same for all cases in this table: *number of generations* $\times$ *population size*, due to the fact that Step 2 in Algorithm 1 requires all the individuals in the current population to be evaluated previously to mating.

6.2 Discussion on Mating Index Experiments

The mating index experiments in Section 5.2 show an important result derived from the present work. As a consequence of parameterizing the degree of exploration applied in the mating phase, new mating strategies have arisen apart from the traditional random one and apart from the pure exploratory/exploitative strategies also found in the literature. The introduction of the mating index allows us to define a whole range of exploratory degrees in a GA, and it turns out that some fitness landscapes can be better optimized by adopting intermediate exploratory strategies. To the best of our knowledge, this intermediate strategies have never been applied in the literature. Given an arbitrary fitness landscape, it remains to investigate how to calculate the most appropriate α value for the optimization problem.

7 Conclusion

Most of the existing approaches to mating in GAs apply restrictions based on similarity between individuals. The novel mating approach introduced in this work considers also an alternative fitness-based criterion for defining mating strategies, which is compared to the widespread similarity-based criterion. The fitness-based criterion offers important advantages regarding computation time savings and, in cases like unimodal function optimization, greater efficiency to approach the optimum in fewer generations.

An important group of mating methods for GAs, for instance *assortative mating* [8, 11, 16], use mating strategies that select just the most similar or the most dissimilar individual from a set of candidates. In our novel approach, a parameter called *mating index* (see Section 3) allows any of the candidates to be chosen. In this way, if a similarity-based criterion is considered, a candidate with an arbitrary degree of similarity can be obtained or, if a fitness-based criterion is considered, a candidate with an arbitrary fitness can be selected. Therefore, a wide spectrum of mating strategies can be investigated by varying the mating index. Intermediate mating indexes happen to be the best option in cases where either a pure exploratory strategy of a pure exploitative strategy yield poor performance.

The novel mating approach facilitates the definition of a Self-Adaptive mating strategy in which each individual has its own mating preference (or mating index). In this way, the fittest individuals determine the most successful mating strategies generation by generation. While Self-Adaptive mating does not perform well on all problems we investigate, it is a good strategy in the case of unimodal problems and also in the case of mutimodal problems when low or middle mating sizes are used. Strategies such as Best-First and Best-Last greatly outperform Random mating in different types of problems. While unimodal problems greatly benefit from using Best-First strategy, the same applies to multimodal problems in the case of Best-Last strategy.

A future research topic derived from the present work is the definition

of a mating strategy that deterministically controls mating index parameter throughout the GA generations. In this way, although the mating index would be the same for every individual in the population, it would change from generation to generation. A possible scheme would consist of assigning $\alpha = \gamma$ at GA initialization and letting α be a monotonically non-increasing function of the number of generations. (Both linear and non-linear reduction schemes could be possible.) This deterministic scheme applies more exploration in the initial generations of the GA, when promising search areas are sought, and applies more exploitation in the final generations, when population diversity has decreased.

Another future research topic is the inclusion of mating size parameter γ as a local parameter in the chromosome of each individual. The performance of the new SELF-ADAPTIVE strategy, resulting from including mating size γ along with mating index α as local parameters, should be compared to the strategies defined in this work.

Acknowledgements

Severino F. Galán was supported by the Spanish Ministry of Science and Innovation through grant JC2007-00110 from its 2007 José Castillejo Program. This material is based upon work by Ole J. Mengshoel supported by NASA under awards NCC2-1426 and NNA07BB97C.

References

[1] L. B. Booker. *Intelligent Behaviour as an Adaptation to the Task Environment*. PhD thesis, Department of Computer and Communication Science, University of Michigan, Ann Arbor, MI, 1982.

[2] G. Chakraborty and B. Chakraborty. Ideal marriage for fine tuning in GA. In *Proceedings of the 1999 IEEE International Conference on Systems, Man, and Cybernetics (IEEE SMC'99)*, volume 1, pages 631–636, Tokyo, Japan, 1999.

[3] H. P. Cohoon, S. U. Hedge, W. N. Martin, and D. Richards. Punctuated equilibria: A parallel genetic algorithm. In *Proceedings of the 2nd International Conference on Genetic Algorithms (ICGA'87)*, pages 148–154, MIT, Cambridge, MA, 1987. Lawrence Erlbaum, Hillsdale, NJ.

[4] S. De, S. K. Pal, and A. Ghosh. Genotypic and phenotypic assortative mating in genetic algorithm. *Information Sciences: an International Journal*, 105(1-4):209–226, 1998.

[5] K. Deb and D. E. Goldberg. An investigation of niche and species formation in genetic function optimization. In *Proceedings of the 3rd International Conference on Genetic Algorithms (ICGA'89)*, pages 42–50, George Mason University, Fairfax, VA, 1989. Morgan Kaufmann, San Francisco, CA.

[6] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer, Berlin, 2003.

[7] L. J. Eshelman and J. D. Schaffer. Preventing premature convergence in genetic algorithms by preventing incest. In *Proceedings of the 4th International Confer-*

ence on Genetic Algorithms (ICGA'91), pages 115–122, San Diego, CA, 1991. Morgan Kaufmann, San Francisco, CA.

[8] C. Fernandes, R. Tavares, C. Munteanu, and A. Rosa. Using assortative mating in genetic algorithms for vector quantization problems. In *Proceedings of the 2001 ACM Symposium on Applied Computing (ACM SAC'2001)*, pages 361–365, Las Vegas, NV, 2001. ACM, New York.

[9] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[10] J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, MI, 1975. Second edition, The MIT Press, Boston, MA, 1992.

[11] C. F. Huang. An analysis of mate selection in genetic algorithms. Technical Report CSCS-2001-002, Center for the Study of Complex Systems, University of Michigan, 2001.

[12] B. Manderick and P. Spiessens. Fine-grained parallel genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms (ICGA'89)*, pages 428–433, George Mason University, Fairfax, VA, 1989. Morgan Kaufmann, San Francisco, CA.

[13] W. N. Martin, J. Lienig, and J. P. Cohoon. Island (migration) models: evolutionary algorithms based on punctuated equilibria. In T. Bäck, D. B. Fogel, and Z. Michalewicz, editors, *The Handbook of Evolutionary Computation*, pages C6.3:1–16. Institute of Physics Publishing and Oxford University Press, 1997.

[14] K. Matsui. New selection method to improve the population diversity in genetic algorithms. In *Proceedings of the 1999 IEEE International Conference on Systems, Man, and Cybernetics (IEEE SMC'99)*, volume 1, pages 625–630, Tokyo, Japan, 1999.

[15] H. Mühlenbein, D. Schomisch, and J. Born. The parallel genetic algorithm as function optimizer. *Parallel Computing*, 17:619–632, 1991.

[16] G. Ochoa, C. Mädler-Kron, R. Rodriguez, and K. Jaffe. Assortative mating in genetic algorithms for dynamic problems. In F. Rothlauf, J. Branke, S. Cagnoni, D. W. Corne, R. Drechsler, Y. Jin, P. Machado, E. Marchiori, J. Romero, G. D. Smith, and G. Squillero, editors, *Applications of Evolutionary Computing*, volume 3449, pages 617–622. Springer, 2005.

[17] E. Ronald. When selection meets seduction. In *Proceedings of the 6th International Conference on Genetic Algorithms (ICGA'95)*, pages 167–173, Pittsburgh, PA, 1995. Morgan Kaufmann, San Francisco, CA.

[18] H. P. Schwefel. *Numerical Optimization of Computer Models*, volume 26. Wiley, Chichester, UK, 1981.

[19] R. E. Smith and C. Bonacina. Mating restriction and niching pressure: results from agents and implications for general evolutionary computation. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2003)*, pages 1382–1393, Chicago, IL, 2003. Springer.

[20] R. E. Smith, C. Bonacina, P. Kearney, and W. Merlat. Embodiment of evolutionary computation in general agents. *Evolutionary Computation*, 8(4):475–493, 2000.

[21] W. M. Spears. Simple subpopulation schemes. In *Proceedings of the 3rd Annual Evolutionary Programming Conference*, pages 296–307, San Diego, CA, 1994. World Scientific.

[22] C. K. Ting, S. T. Li, and C. Lee. On the harmonious mating strategy through tabu search. *Information Sciences: an International Journal*, 156(3-4):189–214, 2003.

[23] T. Unemi and M. Nagayoshi. Evolution of learning robot team via local mating strategy. In *Fourth European Conference on Artificial Life*, Brighton, UK, 1997. Poster Session.

[24] P. M. White and C. C. Pettey. Double selection vs. single selection in diffusion model GAs. In *Proceedings of the 7th International Conference on Genetic Algorithms (ICGA'97)*, pages 174–180, Michigan State University, East Lansing, MI, 1997. Morgan Kaufmann, San Francisco, CA.