Universidad Autónoma de Yucatán

Facultad de Matemáticas

"Desarrollo de un Applet en Java del Micro Algoritmo Genético Usando Optimización Multiobjetivo"

TESIS

Presentada por

Jimmy Josué Peña Koo

En opción al título de

Licenciado en Ciencias de la Computación

Mérida, Yucatán, Mayo de 2002

Dedicatoria

Este trabajo está dedicado a aquella persona que luchó más que yo, y que sabiamente me supo dar la mejor herencia: mis estudios, mi Mamá.

Agradecimientos

Mi primer agradecimiento es a Dios, por darme la sabiduría y sostenerme en cada paso.

A mi Mamá, por sus esfuerzos para sacarme adelante.

A Liz, por su ánimo constante y su compañía.

A Víctor Villanueva, por ser como un padre en todo el apoyo brindado desde mi infancia hasta esta etapa.

A mis asesores: al Dr. Carlos Coello Coello, quien sin conocerme confió dejándome este trabajo, por dedicarme tiempo y gran paciencia para compartir sus conocimientos; al MC Francisco Madera, gracias por tu apoyo como maestro y como amigo.

Al MC Fernando Curi, por tu amistad y apoyo durante mi paso por la facultad y no sólo durante la realización de este trabajo.

Al MIA Gregorio Toscano, por haber sido un asesor para la realización de éste trabajo.

A la “xkuela”, mi grupo de la IX generación, quienes formamos un buen grupo de compañeros y amigos.

Agradezco a CONACyT la beca terminal otorgada para la elaboración de esta tesis. Esta tesis se derivó del proyecto CONACyT titulado "Nuevos Paradigmas en Optimización Evolutiva Multiobjetivo" (Ref. 34201-A) cuyo responsable es el Dr. Carlos A. Coello Coello.

INDICE GENERAL

INTRODUCCIÓN

El presente trabajo tiene como objetivo principal, el desarrollo de una aplicación web para un Micro Algoritmo Genético, usando Optimización Multiobjetivo.

Este Micro Algoritmo fue desarrollado por el M.I.A. Gregorio Toscano Pulido, bajo la asesoría del Dr. Carlos A. Coello Coello.

En el capítulo 1, se hace una breve introducción a la Computación Evolutiva, remontando la discusión a la historia del surgimiento de esta área de investigación, sus precursores y sus tres grandes paradigmas.

En el capítulo 2, se define que es un Algoritmo Genético, se exponen sus principales operadores, se hace mención de los parámetros empleados en un AG, se exponen algunos tipos de AGs y sus principales usos.

En el capítulo 3, se habla de manera general de la Optimización Evolutiva con Objetivos Múltiples, haciendo principal mención de algunas de las principales técnicas evolutivas para la resolución de problemas de optimización multiobjetivo.

En el capítulo 4, se hace una revisión del Micro Algoritmo Genético Multiobjetivo, explicando las principales características de este algoritmo como: la reducción del chequeo de la no dominancia, la implementación de una malla adaptativa para el posiconamiento geográfico y el fundamento elitista. Se definen también los tipos de operadores y parámetros empleados en el algoritmo.

El capítulo 5, empieza con una breve introducción de las ventajas de usar Java para la programación de aplicaciones web. Luego se hace mención de las principales rutinas aplicadas en este lenguaje de programación y se cierra con las conclusiones obtenidas en el presente trabajo.

CAPITULO 1

1. Computación Evolutiva

"En lugar de envidiar la eficacia de la evolución natural, debemos emularla". John Holland.

1.1 Creacionismo

“Tú hiciste los cielos, y los cielos de los cielos, con todo su ejército, la tierra y todo lo que está en ella, los mares y todo lo que hay en ellos; y tú vivificas todas estas cosas”. Durante muchos años ha tenido más aceptación esta tesis del origen de las especies. Tomada del libro más antiguo, más vendido y que ha sido traducido a más idiomas, la Biblia. El creacionismo afirma que Dios creó a todas las especies del planeta de forma separada en un período de siete días terrestres. También nos enseña una jerarquía establecida por Dios donde el hombre ocupa la parte superior enseñoreando sobre todos los peces, aves y bestias de la tierra.

1.2 Evolucionismo

1.2.1 Antecedentes

En 1543, Nicolás Copérnico publicó "De revolutionibus orbium caelestium" [9], libro en el que propuso un nuevo y revolucionario sistema del mundo sobre la base de una Tierra en movimiento.

En 1859, tres siglos más tarde, Alfred Russell Wallace publicó "Sobre el origen de las Especies" [9], en donde también se vertían nuevas y radicales ideas sobre la naturaleza de la vida. Ambos personajes fueron prácticamente tratados de herejes y ambos han sido los artífices de sendas revoluciones científicas y del pensamiento humano.

En Noviembre de 1998 se dio a conocer la primera secuencia completa del genoma de un organismo eucariota, el del "gusano" Caenorhabditis elegans. El director de este proyecto era el genético molecular Sydney Brenner.

El 15 de Febrero de 2001, en *Nature*, y el 16 del mismo mes, en *Science*, se publicaba la secuencia "casi" definitiva de otro genoma, pero esta vez más importante: el humano. Afirmando que nuestro genoma es mucho más sencillo de lo que se creía. Sólo 30000 genes poseen los núcleos de nuestras células, y no hace más de un año aún se especulaba con que serían 80.000. Resultan ser no muchos más que los de una mosca del vinagre y apenas 11000 más que los del *Caenorhabditis elegans*.

La teoría de la evolución se ocupa de tres materias diferentes. La primera es el hecho de la evolución, es decir, que las especies vivientes cambian a través del tiempo y están emparentadas entre sí debido a que descienden de antepasados comunes. La segunda materia es la historia de la evolución, es decir, las relaciones particulares de parentesco entre unos organismos y otros y cuándo se separaron unos de otros los linajes que llevan a las especies vivientes. La tercera materia se refiere a las causas de la evolución de los organismos.

La investigación histórica de la evolución incluye la sucesión histórica de los organismos, precisar los ritmos de la evolución, la multiplicación y la extinción de especies, la colonización de islas y continentes, y otras muchas cuestiones.

En cuanto a los mecanismos o causas de la evolución, Darwin ya apuntó el de la selección natural [1, 5], y hoy se conocen algunos más.

1.2.2 Precursores

Carlos Linneo (1707-1778). En 1686, J. Ray define el concepto de especie con precisión. Busca los caracteres específicos, los que son más constantes. Según él el criterio más fiable de identidad específica es la filiación: nunca una especie nace de la simiente de otra y viceversa. Ray a pesar de ser un seguidor de la Biblia, escribe: *"sea cual fuere la antigüedad de la Tierra y de los cuerpos que hay en ella, la estirpe humana es reciente"* [9].

Esta proposición será retomada y transformada en dogma por un botánico sueco llamado Linneo. Primero construye una clasificación de los vegetales basada en los órganos sexuales, luego crea el llamado sistema binomial de nomenclatura, en 1753.

A los 28 años publica la primera edición de su obra *Sistema Naturae*. El sistema linneano contiene los términos del transformismo. Obliga a los clasificadores a prestar atención a las similitudes y diferencias con otras especies cuando quieren denominar una nueva. Linneo comenzó su carrera firmemente convencido del fijismo, pero al avanzar su trabajo y viendo las variaciones de las especies, sus dudas se acrecentaron. De hecho en las posteriores ediciones de su libro omitió las declaraciones sobre el fijismo.

Georges-Louis Lecrerc, conde de Buffón (1707-1788). Se dedicó a escribir el mundo entero, sus orígenes y cuanto encerraba, y acabó componiendo una enciclopedia sobre la naturaleza, en cuarenta y cuatro tomos, la *Histoire Naturelle, Généralle et Particulaire*, traducida a otros idiomas tan pronto como aparecían. Fue la obra científica más importante y más influyente de su siglo, y la más popular, ya que combinó descripciones redactadas con elegancia con historias sobre la vida de una cantidad apabullante de animales y plantas, además de

introducir discursos sobre astronomía, edad de la tierra y procesos vitales. Llegó a escribir: *toda familia, así animal como vegetal, tiene idéntico origen, e incluso todos los animales proceden de uno solo, que, en la sucesión de las eras ha producido todas las razas de los que ahora existen* [9].

Buffón observó, que las especies se multiplican más rápidamente que los alimentos, lo que implica una lucha por la supervivencia. Y también que había diferencias entre los individuos de la misma especie.

Jean-Baptiste Lamarck (1744-1829). La teoría de la evolución más estructurada de la época la elaboró este colaborador de Buffón y también profesor del Museo de Historia Natural de París. En el año 1800 pronuncia una conferencia inaugural en la que expone una teoría coherente sobre la transformación. Admite la existencia de una evolución de las especies y trata de darle una explicación racional. La idea central es que dicha evolución es obra de la naturaleza, que se vale de infinitos recursos para producir especies; entre ellos dos son los más importantes: el tiempo y las condiciones favorables.

Los efectos de estos factores determinan la transformación progresiva de las facultades de los organismos, que se fortalecen poco a poco, se diversifican y dan lugar a cambios que se transmiten a la descendencia.

Al aceptar la noción de Buffón de la gran edad del mundo, dedujo que las condiciones de la superficie terrestre debía haber sufrido grandes cambios, de modo que los seres vivos tuvieron que adaptarse a ellos. En su opinión, lo hicieron aprendiendo y luchando, tratando siempre de adaptarse, y, mientras tanto, alterando su forma y su comportamiento. El clásico ejemplo aducido para ilustrar la idea de Lamarck es el del alargamiento del cuello de la jirafa: por estirar una y otra vez el cuello para llegar mejor al alimento, consigue tener vértebras más largas.

Thomas Malthus (1766-1834). Publicó su “Essay on the principles of population” [9], en el que concluía que la población tiende a crecer en proporción geométrica, y el sustento se acrecienta en proporción aritmética, lo que significaba que habría lucha continua de los seres por la comida existente; sólo los más fuertes sobrevivirían en la contienda.

1.2.3 Neo-Darwinismo

Darwin infirió que en un medio ambiente de recursos alimentarios estables y con proliferación excesiva de los individuos, estos se enfrentarán a la lucha por sobrevivir. La segunda inferencia de Darwin concluye que en un mundo de poblaciones estables, en el que los individuos han de luchar para sobrevivir, sólo tienen posibilidad de hacerlo los que tienen mejores características, que sus crías probablemente heredarán. Esta desigual proporción de supervivencia es la selección natural.

De aquí Darwin infirió, al final, que el proceso de selección natural, si se cumple con intensidad suficiente y durante bastante tiempo, acarrea al cabo cambios muy perceptibles en una población y culmina en la aparición de una nueva especie.

Hacia 1839 y 1840 había llenado varios cuadernos. En 1842 los organizó y escribió un corto ensayo que bosquejaba su teoría. En 1844 redactó otro más extenso. Pero no publicó ninguno: le sobraban ejemplos para concluir que hablar de evolución en su sociedad y verse relegado a la repudia estaban íntimamente unidos. Así, en 1844 Darwin se desvió del asunto y comenzó a estudiar algo menos comprometido. El resultado fue una obra de cuatro tomos que aún hoy es un hito científico.

En 1855 Darwin vio publicado en una revista científica el siguiente artículo: "Sobre la ley que ha regido la aparición de especies nuevas". Su autor: un tal Alfred Russell Wallace. La tesis: la vida no se creaba sin cesar, sino que se desarrollaban poco a poco formas nuevas de las viejas, *"toda especie cobra*

existencia de modo que coincide en el tiempo y el espacio con otra preexistente y muy emparentada con ella".

Evidentemente, para ejercer su dominio sobre lo que él consideraba su obra debía publicar. Comenzó a escribir en 1856, y hacia junio de 1858, cuando llevaba doscientas cincuenta mil palabras, tuvo de nuevo noticias de Wallace en forma de manuscrito en el que Wallace hablaba de sus ideas. Hasta cierto punto eran éstas coincidentes con las de Darwin.

Finalmente, en 1859, el 24 de Noviembre, a los doce meses de haber recibido el manuscrito de Wallace, publicó su obra "Origin of Species", de la que Wallace recibiría un ejemplar y del cual opinó: "*Perdurará tanto como los Principios de Newton. El señor Darwin ha donado al mundo una ciencia nueva, y su nombre, a juicio mío, se destaca por encima del de muchos filósofos antiguos y modernos. ¡¡La fuerza de la admiración me impide decir más !!*".

Wallace nació catorce años después que Darwin. Gran observador y muy inteligente, se hizo topógrafo profesional, encontró sus primeros fósiles y advirtió la importancia científica de la geología. Se entregó al coleccionismo, rasgo que compartiría con Darwin

En estas condiciones, se preguntó las mismas cosas que Darwin, aunque parezca inverosímil: cómo y porqué cambian las especies y cómo se adaptan éstas a sus medios. Fue una de las coincidencias más portentosas de la historia de la ciencia. Refiriéndose a Darwin, escribió una vez: *"Ni en sueños me hubiera acercado yo a la perfección de su libro. Confieso mi agradecimiento de que no me incumbiera presentar la teoría al mundo"*.

La teoría de Wallace difiere de la de Darwin en algunas cuestiones importantes; por ejemplo, niega que la selección natural sea suficiente para dar cuenta del origen del hombre, lo cual requiere, según Wallace, la intervención divina directa.

También creyó que el proceso evolutivo había finalizado en los hombres y que la evolución sería imposible en adelante.

El biólogo alemán Auguste Weismann (1834-1914) era un buen microscopista, pero hacia 1885 comenzó a perder la vista y se concentró en aspectos teóricos de la herencia. Era de los que ya se centraban por aquella época en la estructura interna y funcionamiento de las células. Así, se convenció de que la base material de la herencia se hallaba en los cromosomas. Durante la fecundación, las instrucciones hereditarias de los progenitores se mezclan entre sí al unirse óvulo y espermatozoide. Weismann creía que esta combinación de instrucciones hereditarias determinaba la estructura del cuerpo.

También propuso una teoría nueva que postulaba la continuidad del "plasma germinal" [9] (los gametos), que se desarrollaban y transmitían el código genético de una generación a otra con independencia de los cambios en el resto del cuerpo. Es decir, que los gametos son sólo un vehículo para la transmisión de la línea germinal.

Esta teoría fue, obviamente, un duro golpe para los seguidores de Lamarck. Si había una barrera entre las células sexuales y el resto del cuerpo, era imposible que las características adquiridas durante la vida se incorporaran al código de la línea germinal.

Para demostrarlo, realizó un famoso experimento en el que cortó la cola a un grupo de ratones, y siguió su descendencia durante 22 generaciones sin encontrar ninguno que naciera sin ella.

A principios del siglo XX, el botánico holandés Hugo de Vries (1848-1935), propone una nueva teoría conocida como mutacionismo o mendelismo [9], que elimina la selección natural como fuente de evolución. De acuerdo con él y con otros genéticos, como William Bateson, hay dos tipos de variaciones en los

organismos: un tipo consiste en la variación ordinaria observada entre los individuos de una especie; otro tipo que consiste en las variaciones que surgen por mutación genética y que ocasionan grandes modificaciones de los organismos y que pueden dar lugar a diferentes especies: "Una nueva especie se origina de repente, es producida a partir de una especie preexistente sin ninguna preparación visible y sin transición".

Thomas Hunt Morgan, a partir de 1910, se propuso demostrar la "presunta falsedad" de las leyes de la herencia de Mendel. Por supuesto, que no demostró la falsedad de las leyes de Mendel, pero sí descubrió el entrecruzamiento cromosómico, fenómeno que es otra fuente de variabilidad, abriendo el camino para que él mismo demostrase que los genes se sitúan en los cromosomas.

Finalmente, como era de esperar, la polémica se resolvió en las décadas de los veinte y treinta, cuando se demostró que los caracteres cualitativos también dependen de la herencia mendeliana, pero de modo que un carácter viene dado por el efecto de varios genes. Varios genéticos pasaron entonces a demostrar matemáticamente que la selección natural, actuando de forma acumulativa sobre pequeñas variaciones, puede producir cambios importantes.

La teoría de la evolución ve en las poblaciones y en las especies las unidades básicas de la evolución. El cambio evolutivo se realiza mediante mutaciones genéticas, cambios en la estructura y número de los cromosomas, mutaciones de los portadores extranucleares de la herencia, recombinación, selección, aislamiento, hibridación y alteraciones casuales de las frecuencias génicas en poblaciones pequeñas o por fluctuaciones del tamaño de la población. Esto es, el Neo-Darwinismo [9], establece que toda la vida en el planeta puede ser explicada a través de: Reproducción, Mutación, Competencia y Selección.

1.3 Computación Evolutiva

En los últimos años los métodos de optimización y la búsqueda de algoritmos que presentan analogías con los procesos naturales, han cobrado más importancia, debido a que con ellos se pueden resolver ciertos problemas de ingeniería que sólo pueden abordarse mediante aproximación en las computadoras actuales. Esta tendencia se basa fundamentalmente en la observación de la destreza de los organismos vivos en la resolución de problemas, por lo que se debe emular la eficacia de la evolución natural. Por tanto podemos copiar a la naturaleza la manera de resolver aquellos problemas en los que no se pueden encontrar soluciones o que éstas no sean lo suficientemente satisfactorias.

Los paradigmas evolutivos actuales están inspirados en la teoría de la evolución de Darwin e intentan emular en lo posible a la Naturaleza. De esta forma, los cromosomas y los genes se suelen asociar de alguna forma a cadenas de bits o a vectores de números reales.

Por otra parte, diversas estrategias de selección imitan la selección natural. Sin embargo, la selección natural no es la única forma de selección que podemos encontrar en la Naturaleza.

Incluso las ideas evolucionan (evolución memética), compitiendo por el espacio cerebral, reproduciéndose y mutando; una idea puede ser una canción, una cadena de texto o una imagen de televisión.

En todos estos casos de evolución, es difícil identificar un sustrato que evoluciona, pero sí encontramos muchas otras características que nos hacen pensar en ellos como en algún tipo de evolución.

A pesar de que la mayoría de los investigadores afirman que los distintos paradigmas de computación evolutiva sólo difieren en cuanto a la representación y

a los operadores de individuo y de población, no existe una visión unificada para todos ellos, y mucho menos una herramienta que los unifique.

La Computación Evolutiva (CE) es un enfoque para abordar problemas complejos de búsqueda y aprendizaje a través de modelos computacionales de procesos evolutivos. Las implantaciones concretas de tales modelos se conocen como Algoritmos Evolutivos (AEs) [7]. El propósito de los AEs consiste en guiar una búsqueda estocástica haciendo evolucionar a un conjunto de estructuras y seleccionando de modo iterativo las más adecuadas.

Los AEs son procedimientos adaptativos (probabilísticos y heurísticos) de optimización y búsqueda que encuentran soluciones a problemas inspirados por los mecanismos de la evolución natural. La finalidad última es la "supervivencia del más apto" y el modo de conseguirlo es por "adaptación al entorno", esto es, los más aptos tienen más posibilidades de sobrevivir y como resultado, más oportunidades de transmitir sus características a las generaciones siguientes.

Para que la mejora de la CE sea efectiva sobre las técnicas clásicas de búsqueda determinista, se deberá disponer de: una población de posibles soluciones representada a través de individuos, un procedimiento de selección basada en la aptitud de los individuos y un procedimiento de transformación; esto es, de construcción de nuevas soluciones a partir de las disponibles actualmente. Sobre este esquema general se han desarrollado tres paradigmas fundamentales:

1.3.1 La Programación Evolutiva (Fogel)

Lawrence J. Fogel propuso en 1960 esta técnica [6, 8], en la cual la inteligencia se ve como un comportamiento adaptativo. Se hace evolucionar una población de máquinas de estados finitos sometiéndolas a transformaciones unitarias.
La programación evolutiva enfatiza los nexos de comportamiento entre padres e hijos, en vez de buscar emular operadores genéticos específicos. Es una abstracción de la evolución al nivel de las especies, por lo que no se requiere el

uso de un operador de recombinación (diferentes especies no se pueden cruzar entre sí). Asimismo, usa selección probabilística.

1.3.2 Las Estrategias Evolutivas (Rechenberg y Schwefel)

Fueron desarrolladas en 1964 en Alemania para resolver problemas hidrodinámicos de alto grado de complejidad por un grupo de estudiantes de ingeniería encabezado por Ingo Rechenberg [1].
Se hace evolucionar una población de números reales que codifican las posibles soluciones de un problema numérico y los tamaños de salto. La selección es determinística.

1.3.3 Los Algoritmos Genéticos (Holland)

Fueron desarrollados por John H. Holland a principios de los 1960s [1, 2]. Su motivación principal fue el aprendizaje de máquina. Se hace evolucionar una población de enteros binarios sometiéndolos a una cruza sexual (operador principal) la cual tiene mayor importancia que la mutación (operador secundario), y usa selección probabilística.

CAPITULO 2

2. Algoritmos Genéticos

Programas que evolucionan, simulando en cierto grado la selección natural y alcanzan a resolver problemas complejos, que ni siquiera quienes los crearon comprenden plenamente. John Holland.

2.1 Orígenes

Los Algoritmos Genéticos (AGs en adelante), denominados originalmente "planes reproductivos", fueron desarrollados por John H. Holland a principios de los 1960s, inspirándose en el proceso observado en la evolución natural de los seres vivos. Esto lo llevó a escribir su libro “Adaptation in Natural and Artificial Systems”, publicado por primera vez en 1975 [12]. En el año de 1989 John Koza comenzó a utilizar el concepto de Holland para la resolución de problemas y lograr ciertas tareas. A partir de entonces en la comunidad científica internacional se despertó un creciente interés hacia esta nueva técnica de optimización y búsqueda, la cual ha cobrado gran popularidad.

La importancia de este tipo de algoritmos en la actualidad se debe a que facilitan la solución de problemas con muy poca información, además de que ayudan en la optimización de ciertas funciones sumamente complejas. Por lo tanto, se puede destacar que sus principales objetivos son: mejorar la calidad de sus soluciones y aumentar la velocidad de ejecución (aunque esto último no se logre del todo).

Los Algoritmos Genéticos son métodos estocásticos de búsqueda ciega (no disponen de conocimientos específicos del problema, la búsqueda se basa en los

valores de la función objetivo) de soluciones cuasi-óptimas. En ellos se mantiene a una población que representa a un conjunto de posibles soluciones la cual es sometida a ciertas transformaciones y a un proceso de selección sesgado a favor de los mejores candidatos.

John Koza define a los AGs de la siguiente manera [13]: Es un algoritmo matemático altamente paralelo que transforma un conjunto de objetos matemáticos individuales con respecto al tiempo, usando operaciones modeladas de acuerdo al principio Darwiniano de reproducción y supervivencia del más apto, y tras haberse presentado de forma natural una serie de operaciones genéticas de entre las que destaca la recombinación sexual. Cada uno de estos objetos matemáticos suele ser una cadena de caracteres (letras o números) de longitud fija que se ajusta al modelo de las cadenas de cromosomas, y se les asocia con una cierta función matemática que refleja su aptitud.

2.2 Definición de Algoritmos Genéticos

Todo organismo viviente consiste de un grupo de células. Todas las células derivan de las divisiones sucesivas de una única célula.
Cada célula de un organismo está formada por un material citoplasmático, que contiene numerosas estructuras pequeñas del mismo tipo llamadas cromosomas.
El cromosoma consta de genes, los cuales son básicamente cadenas de Acido Desoxirribonucleico (ADN) que sirven como modelo a la creación del organismo. Cada gen codifica una proteína en particular, esta codificación puede ser la capacidad de una persona para pigmentar la piel, el cabello y los ojos. En la figura 2.1 observamos la imagen de una cadena de ADN.

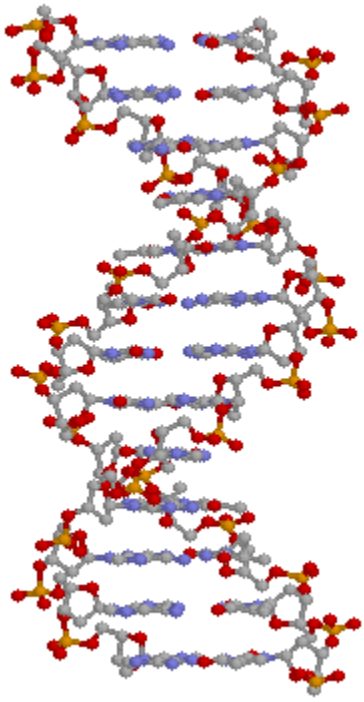


Figura 2.1 Imagen computarizada de una cadena de ADN.

El pensamiento base de los algoritmos genéticos es que dada la elevada tasa geométrica de reproducción de todos los seres orgánicos, su número tiende a crecer a ritmo exponencial, mientras que el espacio físico no lo hace en la misma proporción. Mientras esto ocurra, nacerán muchos más individuos de los que es posible que sobrevivan. En consecuencia, se recurrirá a la lucha por la sobrevivencia ya sea con individuos de la misma especie o de especies diferentes, o simplemente con el entorno, intentando modificar sus características adversas. Tomando en cuenta esto se desprende que un individuo, si actúa de un modo provechoso para él, tendrá una mayor probabilidad de sobrevivir y será seleccionado naturalmente. La teoría de la selección de las especies sostiene que aquellos individuos de una población que posean los caracteres más favorables, dejarán proporcionalmente más descendencia en la siguiente generación; si en adición, tales caracteres se deben a diferencias genéticas que pueden transmitirse a los descendientes, tenderá a cambiar la composición genética de la población, aumentando el número de individuos con dichas características. De esta forma, la población completa de seres vivos se adapta a las circunstancias variables de su

entorno. El resultado final es que los seres vivos tienden a perfeccionarse en relación con las circunstancias que los envuelven.

En su definición matemática [15], los AGs son algoritmos matemáticos de optimización de propósito general basados en mecanismos naturales de selección y genética. Los AGs han proporcionando excelentes soluciones a problemas complejos con gran número de parámetros y conforman un paradigma de búsqueda probabilística, inspirada en la teoría de la evolución de Darwin.

El objetivo principal de un AG, es evolucionar a partir de una población de soluciones para un determinado problema, intentando producir nuevas generaciones de soluciones que sean mejores que la anterior. Estos algoritmos operan en un ciclo simple: creación de la población inicial, selección y reproducción. Este último implica una recombinación y mutación del material genético de las soluciones.

Algunas de las características más notables de estos algoritmos son [10, 15]:

- No necesitan conocimientos específicos sobre el problema que intentan resolver.
- Se usan parámetros codificados como una cadena de longitud finita sobre un alfabeto finito.
- Poseen un mecanismo de paralelismo implícito que les permiten procesar un número mayor de esquemas que los representados explícitamente en la población.
- Usan operadores probabilísticos.
- Se afectan menos por los máximos locales (falsas soluciones).
- Están menos restringidos por continuidad y derivadas.

2.3 Descripción de un Algoritmo Genético Simple

La estructura general de un Algoritmo Genético Simple se ilustra con el siguiente segmento de pseudocódigo [10]:

```
t := 0;
generar población inicial, G(t);
evaluar G(t)
mientras (no se cumpla la condición de terminación) hacer
inicio
        t := t + 1;
        generar G(t) usando G(t-1);
        evaluar G(t);
fin
```

Figura 2.2 Estructura de un Algoritmo Genético Simple.

Un AG empieza con un conjunto inicial (población) de soluciones alternativas (individuos) para el problema a resolver, las cuales son evaluadas en términos de la adecuación de la solución. Los operadores de selección, cruza y mutación son aplicados para obtener nuevos individuos (descendientes) que constituyen una nueva población. La interacción de los operadores a los individuos más aptos conduce al incremento de la calidad de las soluciones durante el curso de muchas iteraciones (generaciones). Cuando se encuentra un criterio de terminación, finaliza el proceso de búsqueda y se presenta la solución. El AG utiliza un conjunto de parámetros que el usuario introduce para guiar el proceso evolutivo, tales como el tamaño de la población, el número máximo de generaciones y las probabilidades de cruza y mutación. A continuación se describirán cada una de estas etapas.

2.3.1 Representación

En un AG, las soluciones potenciales a un problema se representan mediante cadenas binarias de bits (0's y 1's) de una longitud determinada por el número de variables existentes en la solución y por el número de bits necesarios para codificarlas.

Figura 2.3 Representación de una solución potencial.

En términos biológicos, se llama cromosoma a una solución del problema en un AG. Así, los cromosomas están compuestos por unidades binarias denominadas genes. Al valor de un gen se le denomina alelo y a su posición en el cromosoma "lugar" (o locus, que es el término que se una en inglés). Al paquete genético total se le denomina genotipo y a la interacción del genotipo con su entorno se le denomina fenotipo. La adaptación de cada individuo depende de su fenotipo, el cual se puede inferir de su genotipo; es decir, puede calcularse a partir del cromosoma utilizando la función de decodificación.

Se deben tomar en cuenta los siguientes aspectos relacionados con la codificación de un problema [16, 17]:

- Se debe utilizar el alfabeto más pequeño posible para representar los parámetros. Normalmente se utilizan dígitos binarios.
- Las variables que representan los parámetros del problema deben ser discretizadas para poder representarse con cadenas de bits. Hay que utilizar suficiente resolución para asegurar que la salida tiene un nivel de precisión adecuado. Se presupone que la discretización es representativa de la función objetivo.
- La mayor parte de los problemas tratados con AGs son no lineales y muchas veces existen relaciones "ocultas" entre las variables que conforman la solución. Esta interacción es referida como epístasis, y es necesario tomarla en cuenta para una representación adecuada del problema.
- El tratamiento de los genotipos inválidos debe ser tomado en cuenta para el diseño de la codificación. Por ejemplo, si se requieren 1200 valores para representar una variable, se necesitarán al menos 11 bits, pero estos codifican un total de 2048 posibilidades, "sobrando" 848 patrones de bits no

necesarios. A estos patrones se les puede dar un valor cero de adaptación, ser substituidos por un valor real, o pueden simplemente descartarse.

2.3.2 Población inicial

Una cuestión a plantearse es el tamaño idóneo de la población. Parece intuitivo que las poblaciones pequeñas corren el riesgo de no cubrir adecuadamente el espacio de búsqueda, mientras que el trabajar con poblaciones de gran tamaño puede acarrear problemas relacionados con el excesivo costo computacional.

Una población inicial está formada por un conjunto de *m* (tamaño de la población) individuos, donde *m* es un parámetro de entrada al AG. Para obtener la población inicial se generan *m* cadenas aleatoriamente, pudiendo contener cada gen uno de los posibles valores del alfabeto con probabilidad uniforme. El procedimiento de inicialización de un individuo consiste simplemente en asignar, para cada gen de su cromosoma un valor aleatorio 0 ó 1. Una forma simple de obtener valores binarios aleatorios para cada gen de un cromosoma puede ser la siguiente:

- Generar un número real aleatorio $r \in [0,1]$
- Si $r \leq 0.5$ asignar 0, si no asignar 1

Con la decodificación del cromosoma obtendremos su fenotipo y la adecuación de la solución al entorno se obtiene a través de la función de evaluación.

2.3.3 Función de aptitud

Dado un cromosoma, la función de aptitud consiste en asignarle un valor numérico de adaptación, el cual se supone que es proporcional a la utilidad o habilidad del individuo representado. Por otra parte una dificultad en el comportamiento del AG puede ser la existencia de gran cantidad de óptimos locales, así como el hecho de que el óptimo global se encuentre muy aislado.
Adicionalmente debe ser rápida, ya que hay que aplicarla para cada individuo de cada población en las sucesivas generaciones, por lo cual, gran parte del tiempo de corrida de un algoritmo genético se emplea en la función de evaluación.

Un problema habitual presentado en la función de aptitud es la convergencia prematura [17]. Esta surge cuando existen individuos con una adaptación al problema muy superior al resto, los cuales dominan a la población a medida que avanza el algoritmo. Por medio de una transformación en la función, en este caso una compresión del rango de variación de la función o bien controlando el número de oportunidades reproductivas de cada individuo, se pretende que dichos superindividuos no lleguen a dominar la población. También en el caso contrario, una convergencia lenta del algoritmo se resolvería de manera análoga, pero en este caso efectuando una expansión del rango de la función.

2.3.4 Operadores Genéticos

Después de ser evaluadas todas las soluciones de la población en una generación, el proceso evoluciona hacia una nueva generación. Esta nueva generación sufrirá transformaciones dentro de un esquema básico de funcionamiento. Los cambios en la nueva generación están dados básicamente por medio de tres operadores genéticos: reproducción, cruza y mutación.

Selección

Consiste en hacer un muestreo, a partir de la población inicial, los cromosomas que se cruzarán en la siguiente generación de acuerdo al grado de bondad que aporten al problema (a mayor bondad, mayores oportunidades de ser seleccionados).

Los criterios más usados en la práctica son: por sorteo, universal o por ruleta y por torneos. Son muestreos estocásticos, en los cuales se asigna una probabilidad de selección o puntuación a cada elemento de la población con base en su bondad o función de aptitud. Se asumirá la notación siguiente: la puntuación p_i asociada al individuo x_i de la población $P=\{x_1, ..., x_n\}$.

Por Sorteo

Se consideran las puntuaciones como probabilidades de elección para formar la muestra, construyendo ésta a partir de k ensayos de una variable aleatoria con

dicha distribución de probabilidades. Para escoger k individuos se hace lo siguiente:

1.Se calculan las puntuaciones acumuladas así:

$q_0 = 0$

$q_i = p_1 + ... + p_i$ para toda i = 1,...,n

2. Se generan k números aleatorios simples r_j para toda j = 1,...,k

3. Para cada j = 1,...,k se elige el individuo x_i que verifique:

$q_{i-1} < r_j < q_i$

Es de notarse que existe la posibilidad de que un individuo pueda ser elegido en repetidas ocasiones dentro de una muestra. De la misma manera algún individuo puede no ser seleccionado nunca.

Por Ruleta

Este es similar al muestreo por sorteo sólo que ahora se genera un único número aleatorio simple r y con él se asignan todas las muestras de modo parecido a como se haría al girar una ruleta.

Se puede ver que el muestreo por ruleta es sencillo y rápido de implementar y en la práctica proporciona unas características análogas a las del muestreo por sorteo. Por este motivo suele usarse en sustitución de éste.

Por Torneos

Cada elemento de la muestra se toma eligiendo el mejor de los individuos de un conjunto de z elementos tomados al azar de la población base; esto se repite k veces hasta completar la muestra. El parámetro z suele ser un entero pequeño comparado con el tamaño de la población base.

Cruza

Después de haber formado la nueva población en la generación actual, se procede a aplicar el primer operador de recombinación, el operador de cruza o reproducción sexual de los individuos seleccionados. En esta etapa los individuos intercambian material cromosómico para generar descendientes, los cuales

formarán la siguiente generación. Las 2 formas más comunes de reproducción sexual son: uso de un único punto de cruza y uso de dos puntos de cruza.

Cuando se usa un único punto de cruza, se toman dos individuos y corta sus cromosomas en una posición seleccionada al azar, produciendo dos segmentos anteriores y dos posteriores. Los posteriores se intercambian para obtener los cromosomas nuevos.

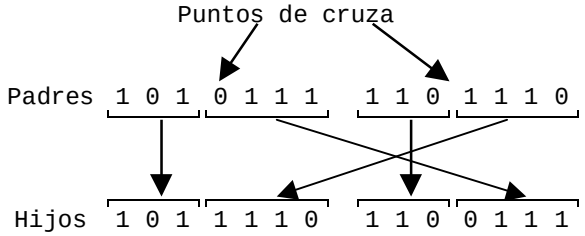


Figura 2.4 Operador de cruza basado en un solo punto.

Cuando se usan dos puntos de cruza, se procede de manera similar. Se escogen dos individuos, se escogen dos puntos aleatoriamente, obteniendo dos segmentos anteriores, dos medios y dos posteriores, luego se intercambian los segmentos de en medio para así obtener los cromosomas nuevos.

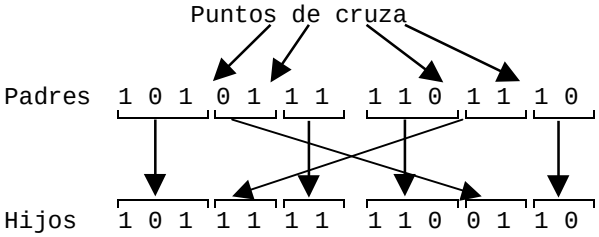


Figura 2.5 Operador de cruza basado en dos puntos.

La cruza se maneja como una puntuación que indica la frecuencia con la que se efectuará. De aquí que no todas las parejas se cruzarán, sino que algunas pasarán intactas a la siguiente generación. También existe la posibilidad de que uno o los dos puntos de cruza se encuentren en los extremos de la cadena, en cuyo caso sólo se hará una cruza usando un solo punto, o ninguno cruza, según corresponda. Hay otro operador propuesto originalmente en Alemania llamado elitismo, el cual consiste en mantener intacto a través de las generaciones al individuo más apto, por lo que no se cruza sino hasta que surge otro individuo mejor que él.

Mutación

Este operador aplica un cambio a uno de los genes de un cromosoma elegido aleatoriamente. Permite la introducción de nuevo material cromosómico en la población, tal y como sucede con sus equivalentes biológicos.

Se maneja como un porcentaje que indica con qué frecuencia se efectuará. Se aplica con poca frecuencia.

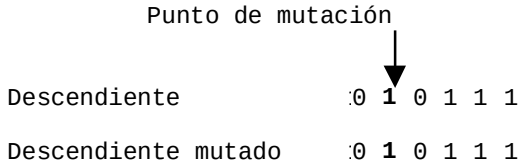


Figura 2.6 Mutación sencilla.

2.3.5 Parámetros

En cuanto a los parámetros que usa el AG, los más significativos son: el tamaño de la población, el número de generaciones, la probabilidad de cruza y la probabilidad de mutación.

La elección de posibles tamaños de la población para representaciones binarias de longitud fija, se puede calcular en función de la longitud de los cromosomas. Asimismo, el número esperado de generaciones hasta la convergencia puede verse como una función logarítmica del tamaño de la población.

Para las probabilidades de cruza y mutación, los estudios realizados apuntan hacia una alta probabilidad de cruza y una baja probabilidad de mutación.

Los siguientes valores han sido considerados como aceptables para un buen rendimiento de los AG en funciones de optimización [3, 10]:

- Tamaño de la población: 50 – 100.
- Probabilidad de cruza: mayor igual al 60%.
- Probabilidad de mutación: por lo general no supera el 5%.

2.4 Tipos de Algoritmos Genéticos

Cada tipo de AG se basa en la naturaleza, pero cada uno tomado de una metáfora distinta.

2.4.1 Algoritmos Genéticos Generacionales

Los AGs Generacionales [17] o Canónicos se asemejan a la forma en que se reproducen los insectos. En este caso una generación pone huevos, se aleja geográficamente o muere y es substituida por una nueva. Se realizan las cruzas en una piscina de individuos, los descendientes son puestos en otra. Al final de la fase reproductiva se elimina la generación anterior y se pasa a la nueva.

2.4.2 Algoritmos Genéticos de Estado Fijo

Se basan en otra representación de la naturaleza, usan el esquema generacional de los mamíferos y otros seres vivos caracterizados por su longevidad. En este tipo coexisten padres y sus descendientes, permitiendo que sus hijos sean educados por sus progenitores, pero luego también se genera competencia entre ellos.

En este tipo de AG, el operador de selección escoge a los individuos que serán padres, al igual que a los individuos a ser eliminados en esta población para dar lugar a los descendientes.

2.4.3 Algoritmos Genéticos Paralelos

La búsqueda genética es inherentemente paralela, pues al evolucionar recorre muchas soluciones, cada una representada por un individuo de la población. Al igual se nota el paralelismo en muchos AGs, al evolucionar en más de una población a la vez. Cada una de ellas está normalmente aislada geográficamente. Este tipo de metáfora biológica, origina dos modelos que toman en cuenta esta variación y utilizan varias poblaciones concurrentemente.

Modelo de Islas

Consiste en dividir una población en varias subpoblaciones (islas), en cada una de las cuales se ejecutará un AG. Cada cierto número de generaciones se efectúa un intercambio de información entre subpoblaciones. A este proceso se le denomina migración. La introducción de la migración hace que los modelos de islas sean capaces de explotar las diferencias entre las subpoblaciones, obteniendo así una fuente de diversidad genética. El índice de migración es muy importante, pues de éste depende la convergencia prematura de la búsqueda.

Es posible distinguir diferentes modelos de islas en función de su intercambio de información genética. A continuación se describen algunos.

Comunicación en Estrella

En este caso se selecciona una subpoblación como maestra (la que tiene mejor media con base en el valor de la función objetivo). Las otras son consideradas como esclavas. Todas las subpoblaciones esclavas mandan a sus mejores individuos a la subpoblación maestra y ésta a su vez manda a sus mejores individuos a cada una de las subpoblaciones esclavas.

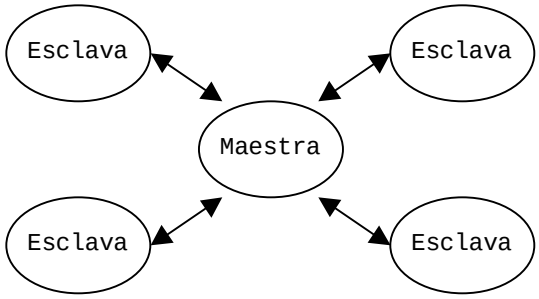


Figura 2.7 Modelo de islas, comunicación en estrella.

Comunicación en Red

En este tipo de comunicación no existe jerarquía entre subpoblaciones. Todas las subpoblaciones mandan a sus mejores individuos al resto de las subpoblaciones.

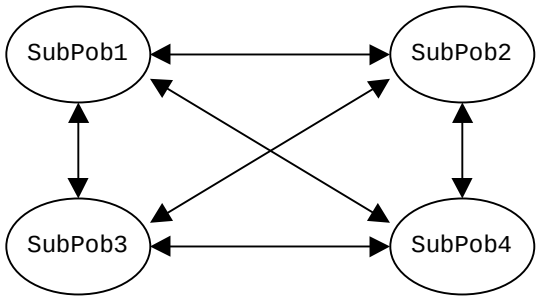


Figura 2.8 Modelo de islas, comunicación en red.

Comunicación en Anillo

Cada subpoblación envía a sus mejores individuos a otra subpoblación vecina, efectuándose la migración en un solo sentido de flujo.

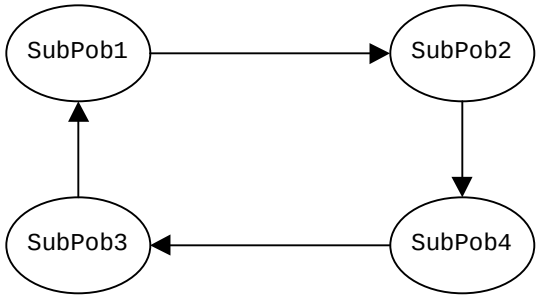


Figura 2.9 Modelo de islas, comunicación en anillo.

Modelo Celular

Este modelo coloca cada individuo en una matriz, donde cada uno sólo podrá reproducirse con los individuos que tenga a su alrededor, escogiendo al azar o al mejor adaptado. El descendiente pasa a ocupar una posición cercana.

No hay islas en este modelo pero hay efectos potenciales similares. Se da cuando la cruza está restringida a individuos adyacentes. Entonces, si dos individuos están separados por varios espacios, están tan aislados como si estuvieran en dos islas. Este caso es conocido como aislamiento por distancia.

Luego de la primera evaluación, los individuos continúan distribuidos al azar sobre la matriz. Entonces empiezan a emerger zonas con cromosomas y adaptaciones semejantes. La reproducción y selección local crean tendencias evolutivas aisladas; luego de varias generaciones, la competencia local resultará en grupos más grandes de individuos semejantes.

2.5 Usos de los Algoritmos Genéticos

Algunos ejemplos de las aplicaciones [1, 10] de estos algoritmos son:

- Parametrización de sistemas.
- Búsqueda de reglas en juegos.
- El problema del agente viajero.
- Enrutamientos.
- Resolución de sistemas de ecuaciones no lineales.
- Optimización (estructural, de topologías, numérica, combinatoria, etc.).
- Aprendizaje de máquina (sistemas clasificadores).
- Bases de datos (optimización de consultas).
- Reconocimiento de patrones (por ejemplo, imágenes).
- Generación de gramáticas (regulares, libres de contexto, etc.).
- Planeación de movimientos de robots.
- Predicción de series temporales.

2.6 Conclusiones

Los AGs resultan ser un tema muy extenso, por lo que aquí se ha presentado tan solo una introducción de las características y funcionamiento de estos algoritmos. Los AGs son una técnica robusta, y pueden tratar con éxito una gran variedad de problemas provenientes de diferentes áreas, incluyendo aquellos en los que otros métodos encuentran dificultades. El mayor campo de aplicación de los Algoritmos Genéticos se relaciona con aquellos problemas para los cuales no existen técnicas especializadas. Incluso en el caso en que dichas técnicas existan, y funcionen bien, pueden efectuarse mejoras de las mismas, hibridizándolas con los AGs. En cuanto a sus objetivos se puede mencionar que aumentan la velocidad de ejecución y mejoran la calidad de las soluciones.

CAPITULO 3

3. Optimización Evolutiva con Objetivos Múltiples

"Una asignación es eficiente, si no existe ninguna otra asignación que permita a todo el mundo disfrutar al menos del mismo bienestar y que mejore estrictamente el de algunas personas". Vilfredo Pareto, 1896.

3.1 Introducción

A principios de la Segunda Guerra Mundial, se forman grupos de especialistas militares conocidos como investigadores de operaciones (*operations researchers*). Estos grupos tenían el objetivo de manejar los problemas tácticos y estratégicos que enfrentaban los organismos militares.

En 1947, George Dantzig desarrolló el Método Simplex para resolver problemas de programación lineal. Diez años después, Churchman, Ackoff y Arnoff publicaron el primer libro sobre Investigación de Operaciones (I.O.).

La I.O., surge como toda ciencia, al converger el interés por resolver determinados problemas y el desarrollo de técnicas, instrumentos y métodos adecuados para la resolución de los mismos.

En general, la I.O. es la ciencia que proporciona las técnicas que permiten juzgar, ponderar y valorar cada uno de los datos de entrada de forma racional para poder alcanzar mejores soluciones.

Por lo anterior, la I.O. debe considerarse como un instrumento eficaz y preciso en la resolución de problemas con aplicaciones en muchas disciplinas. Ha sido desarrollada por economistas y matemáticos, con el fin de determinar el empleo de recursos limitados de forma óptima.

Cualquier problema en el que se requiera encontrar una solución o tomar decisiones, se puede plantear como un problema de optimización.
Sin embargo, en los procesos reales de toma de decisiones, generalmente, las soluciones factibles se ordenan teniendo en cuenta diferentes criterios que reflejen sus preferencias. Esto es, una empresa desea establecer sus decisiones óptimas no sólo con base en un criterio, por ejemplo el beneficio, sino teniendo en cuenta otros criterios como costo, volumen de ventas, riesgo, etc.

3.2 Optimización con Objetivos Múltiples

A lo largo de los años, en los problemas de optimización se ha considerado la existencia de un solo criterio u objetivo al definir problemas complejos. Por ejemplo, al hablar del diseño de una estructura para la cual se busca minimizar el peso y volumen del material usado, y al mismo tiempo maximizar su resistencia a la deformación y así obtener la máxima seguridad, se distinguirá un conflicto entre los objetivos, ya que la máxima resistencia demanda mayor volumen de material y en consecuencia mayor peso de la estructura. Estos objetivos no pueden ser combinados de manera simple en una sola función.

Al paradigma que permite considerar todas las funciones objetivo existentes en un problema de optimización se le conoce como Optimización con Objetivos Múltiples.

En la teoría de la decisión con objetivos múltiples [28, 31], se consideran los siguientes conceptos:

- Atributos: son los valores del tomador de decisiones que corresponden con la realidad y son medidos con independencia de sus deseos. Se expresan como una función matemática f(x) de las variables de decisión.
- Variables de decisión: son las cantidades numéricas cuyos valores representan soluciones para el problema de optimización. Estas soluciones

deben satisfacer las restricciones del problema específico para ser soluciones válidas.

- Objetivos: son las direcciones de mejora de los atributos. Hay sólo dos direcciones: máximo y mínimo. Luego, los objetivos implican la maximización o minimización de las funciones que corresponden a los atributos, esto es, Max f(x) ó Min f(x).
- Niveles de aspiración: es el nivel aceptable de logro para un atributo.
- Metas: se generan al combinar un atributo con el nivel de aspiración correspondiente. Las metas se representan como desigualdades y su expresión matemática será f(x) $\geq$, $\leq$, ó = t, donde el parámetro t representa el nivel de aspiración. A pesar de que se representan igual que las restricciones tradicionales, existe diferencia entre ambos conceptos en dependencia del significado que se le da al término de la derecha de la correspondiente desigualdad. Cuando se trata de una restricción tradicional, el término de la derecha debe alcanzarse para lograr una solución factible; cuando se trata de una meta, el término de la derecha es un nivel de aspiración deseado por el centro decisor que puede o no alcanzarse.
- Criterios: son los atributos, objetivos o metas que se consideran relevantes en el problema decisional.

La optimización con objetivos múltiples, fue definida por A. Osyczka en 1985 de la siguiente manera [11]: Optimización con objetivos múltiples (también llamada optimización con criterios múltiples, múltiple desempeño o vector de optimización) puede ser definida como el problema de encontrar un vector de variables de decisión, sujeto a ciertas restricciones, que optimiza una función vectorial cuyos elementos representan las funciones objetivo. Estas funciones forman una descripción matemática de los criterios de evaluación, los cuales usualmente están en conflicto unos con otros. Así, el término “optimizar” significa encontrar una solución que de valores aceptables para el diseñador en todas las funciones objetivo.

3.2.1 Definición del problema

La optimización con objetivos múltiples, consiste en encontrar un vector de variables de decisión

$$\bar{X}^* = [x_1^*, x_2^*, ..., x_n^*]^T$$

(donde n, es el número de variables de decisión), que satisfaga las m restricciones de desigualdad

$$g_i(\bar{X}) \geq 0 \qquad i = 1, 2, ..., m$$

las p restricciones de igualdad

$$g_i(\bar{X}) = 0 \qquad i = 1, 2, ..., p$$

(donde p < n, de lo contrario no quedarían grados de libertad para la optimización) y que optimice la función vectorial

$$\boldsymbol{f}(\bar{X}) = [f_1(\bar{X}), f_2(\bar{X}), ..., f_k(\bar{X})]^T$$

donde: $f_i(\bar{X})$, representa la función objetivo para el criterio i

$\bar{X} = [x_1, x_2, ..., x_n]^T$, es el vector de variables de decisión

k, es el número de objetivos o criterios de optimización.

Dichas funciones objetivo forman una descripción matemática de los criterios de desempeño que usualmente están en conflicto entre sí.

3.2.2 Óptimo de Pareto

La optimalidad, juega un papel muy importante en la resolución de problemas con un único objetivo en los que se permite determinar la solución óptima, que es una solución factible que da el mejor valor a la función objetivo. Sin embargo, en el

caso de múltiples objetivos no es aplicable este concepto de óptimo, ya que una solución que optimice un objetivo, en general no optimizará los objetivos restantes. Esto lleva a un nuevo concepto denominado “punto eficiente”, también conocido como punto inferior o no dominado. Además, en la optimización con objetivos múltiples no existe una solución óptima única, sino un conjunto de soluciones no dominadas, para las cuales no existen soluciones mejores en todos los objetivos del problema. A este conjunto de soluciones se denomina conjunto de Pareto, que recibe este nombre en honor del economista que lo desarrolló: Vilfredo Pareto (1848 - 1923).

Un conjunto de soluciones es un óptimo de Pareto cuando está formado por soluciones factibles, esto es, que cumplen las restricciones, tales que no existe otra solución factible que proporciona una mejora en un atributo sin producir un empeoramiento en al menos otro de los atributos.

De manera formal, un punto $\overline{X}^* \in$ F es óptimo de Pareto si para toda $\overline{X} \in$ F,

$f_i(\overline{X}^*) = f_i(\overline{X})$ para toda i $\in$ I, i=1, 2, ..., k

o, hay al menos una i $\in$ I, tal que

$f_i(\overline{X}^*) > f_i(\overline{X})$

3.2.3 Dificultades en la Optimización con Objetivos Múltiples

En un problema con objetivos múltiples, no resulta obvio determinar qué es lo óptimo. Los objetivos son usualmente conflictivos y contradictorios. Una solución puede ser muy buena con respecto a un objetivo, pero al mismo tiempo ser mala para otro.

Al considerar, por ejemplo, el diseño de un aparato electrónico, uno buscaría maximizar el desempeño del aparato dadas ciertas características electrónicas y

minimizar el costo de producir el aparato. Estos objetivos no pueden ser combinados en una sola función porque son inconmensurables, esto es, están medidos en unidades distancias.

Otro problema es la representación del espacio de decisiones. En problemas bi-objetivo esta representación es relativamente sencilla. Si pasamos a un problema con tres objetivos, al querer realizar una representación cartesiana, tendríamos que definir un gráfico de tres dimensiones, lo cual es más complicado. Con más de tres objetivos, este tipo de representaciones ya no es factible.

3.3 Optimización Evolutiva con Objetivos Múltiples

A partir de los trabajos desarrollados por Rosenberg a finales de los 1960s, con respecto a la posibilidad de usar la búsqueda genética como una forma de lidiar con objetivos múltiples, esta nueva área de investigación ha crecido considerablemente.

Los Algoritmos Genéticos evolucionan poblaciones de soluciones candidatas. Estos algoritmos no son métodos que se basan en mover un solo punto en el espacio de búsqueda. Debido a ello, el producto de la búsqueda de un AG no es un individuo, sino la información contenida en la población entera. Sin embargo, en numerosas aplicaciones se han empleado AGs para encontrar una sola solución que el usuario espera sea óptima o cercana al óptimo. La solución a un problema de optimización con objetivos múltiples es, en contraposición, un conjunto de puntos que puede ser infinito. Con base en lo anterior, los problemas de este tipo constituyen un área adecuada para que el enfoque de AGs demuestre un desempeño superior a otros métodos de búsqueda ciega, debido a que procesan varias soluciones a la vez.

3.4 Técnicas para la Optmización con Objetivos Múltiples

Se han desarrollado un gran número de técnicas para la optimización con objetivos múltiples, a partir de que el economista Vilfredo Pareto presentara en 1986 el concepto de solución compromiso. Estas técnicas se pueden clasificar en dos grandes grupos [20, 21, 30, 31]: tradicionales (Investigación de Operaciones) y alternativas (algoritmos evolutivos, método de Monte Carlo, recocido simulado, búsqueda tabú entre otras).

3.4.1 Técnicas Tradicionales

Las técnicas tradicionales son desarrolladas por la comunidad de I.O. Sin embargo son altamente limitadas y costosas para obtener una respuesta, a medida que crece el problema. Además, en caso de hallar soluciones no dominadas, éstas resultan ser una única solución o una porción limitada del frente de Pareto esperado.

En la siguiente tabla se presenta una clasificación de los principales métodos que se emplean en las técnicas tradicionales con sus clásicos exponentes.

Técnicas Tradicionales	
Métodos sin preferencias	Métodos del Criterio Global Método Multiobjetivo de los paquetes próximos
Métodos a priori	Método de la función valor Método Lexicográfico Programación de metas
Métodos a posteriori	Método de suma de pesos Método de ε restricciones
Métodos Interactivos	Método Interactivo de compromisos valuados Método de satisfacción de compromisos Método de Geoffrion-Dyer-Feinberg Técnica secuencial de optimización próxima Método de Tchebycheff Método de punto de referencia Método GUESS Búsqueda Light Beam Enfoque de referencia de dirección Método NIMBUS

Tabla 3.1 Técnicas tradicionales empleadas en la Optimización Multiobjetivo

3.4.2 Técnicas Evolutivas

En la siguiente tabla se presenta una clasificación de las principales técnicas evolutivas para optimización multiobjetivo.

Técnicas Evolutivas	
Formas Simplistas	Suma de pesos Programación de metas Satisfacción de metas Método de ε restricciones
Técnicas no basadas en óptimos de Pareto	Vector Evaluated Genetic Algorithm, VEGA Ordenamiento Lexicográfico Uso de géneros para identificar objetivos Uso de Min/Max con pesos Algoritmo genético no generacional Uso de pesos generados aleatoriamente y elitismo
Técnicas basadas en óptimos de Pareto	Multiobjective Genetic Algorithm, MOGA Non-dominated Sorting Genetic Algorithm, NSGA Niched Pareto Genetic Algorithm, NPGA NSGAII Pareto Archived Evolution Strategy, PAES

Tabla 3.2 Técnicas evolutivas empleadas en la Optimización Multiobjetivo

A continuación se hace una breve revisión de las principales técnicas empleadas en la Optimización Evolutiva Multiobjetivo.

Vector Evaluated Genetic Algorithm, VEGA

David Schaffer [29], basándose en GENESIS [23], modificó el operador de selección al genético simple, desarrollando un método capaz de lidiar con objetivos múltiples. A este método se le conoce actualmente como VEGA.

En esta técnica, el operador de selección genera un número de subpoblaciones igual al número de objetivos (k), y se efectúa selección proporcional de acuerdo al objetivo correspondiente. Posteriormente se mezclan las subpoblaciones obtenidas para generar una nueva población única, a la cual se le aplican lo operadores genéticos de cruza y mutación.

Multiobjective Genetic Algorithm, MOGA

Esta técnica fue propuesta por Fonseca y Fleming [22]. La jerarquía de un individuo está dada por el número de individuos por los cuales es dominado dentro de la población actual. De ahí que a los individuos no dominados se les asigne la jerarquía 1, mientras que el resto de la población es penalizada conforme es dominada por más individuos. La posición de jerarquía está dada por:

$$\text{Jerarquía}(x_i,t) = 1 + p_i^t$$

Donde:

x_i = individuo correspondiente al subíndice i.

t = la generación actual.

p_i^t = número de individuos que dominan al individuo perteneciente al subíndice i en la generación t.

Niched Pareto Genetic Algorithm, NPGA

El NPGA es un AG convencional y con selección de torneo binario basado en no dominancia. Cada vez que la selección de torneo necesita comparar dos individuos, el mejor se determina de acuerdo a los siguientes factores:

1. **a** domina a **b**, o **b** domina a **a**
2. **a** o **b** es dominado por al menos un individuo de un subconjunto de la población, y
3. en caso de empate, gana el que tenga menos individuos en su vecindad (o nicho).

A partir de los problemas resueltos por Horn y Nafpliotis [22], indican que el NPGA puede encontrar poblaciones distribuidas en forma más uniforme sobre el frente de Pareto que en el trabajo de David Schaffer.

Pareto Archived Evolution Strategy, PAES

Propuesta por Knowles y Corne [25]. Utiliza búsqueda local y un archivo histórico para almacenar las soluciones no dominadas que va encontrando a lo largo de la corrida. Para mantener la diversidad apropiada en este archivo histórico se

introdujo la idea de una malla adaptativa. Las ventajas que presentó esta malla adaptativa fueron visibles respecto al uso de los nichos, debido a que redujo el costo computacional, y para adaptarse no requiere de parámetros extra que son clave y necesarios para el uso de nichos y además afectan significativamente el desempeño del algoritmo.

3.5 Conclusión

En el presente capítulo, se definieron conceptos necesarios para la comprensión de la optimización con objetivos múltiples, y la importancia del uso de los algoritmos evolutivos para resolver estos problemas, debido a su flexibilidad, adaptabilidad y gran desempeño. También se describieron brevemente algunas de las principales técnicas empleadas en la optimización evolutiva con objetivos múltiples.

CAPITULO 4

4. Micro Algoritmo Genético Multiobjetivo

"El resultado obtenido por el MAGM es muy alentador, ya que podemos obtener de esta simple aproximación una importante representación del frente de Pareto a un bajo costo computacional". Coello y Toscano, 2001.

4.1 Introducción

En el pasado han sido propuestas varias técnicas para la solución de problemas de Optimización Evolutiva Multiobjetivo. No obstante, recientemente ha sido poco el énfasis puesto en el desarrollo de una técnica eficiente. Para que una técnica evolutiva multiobjetivo consuma poco tiempo máquina durante la corrida del algoritmo, ésta deberá centrarse en dos puntos principales: el tamaño de la población (a mayor tamaño, se requerirá mayor tiempo para el chequeo de no dominancia entre toda la población) y un mecanismo para guardar la diversidad del frente de Pareto (para almacenar las soluciones no dominadas en diferentes regiones geográficas).

El micro algoritmo genético multiobjetivo (MAGM) fue desarrollado por Carlos Coello y Gregorio Toscano [33, 34], basándose en la tendencia actual del desarrollo de técnicas eficientes. Aunado a esto, el algoritmo también se fundamenta en el elitismo, afirmando que se puede obtener un frente de Pareto con una mejor calidad de soluciones y un menor costo computacional.

De aquí que el cimiento ideológico del MAGM sea el siguiente:

1. Del desarrollo de técnicas eficientes, reducen el chequeo de no dominancia entre toda la población, ideando una manera de trabajar con pocos individuos para ser comparados.

2. Se adoptó una malla adaptativa como mecanismo de posicionamiento geográfico, y así poder mantener diversidad de las soluciones no dominadas.
3. Se introducen tres tipos de elitismo para obtener una mejor calidad de soluciones.

4.2 Algoritmo General

El funcionamiento general de la técnica, se puede observar gráficamente en la siguiente figura:

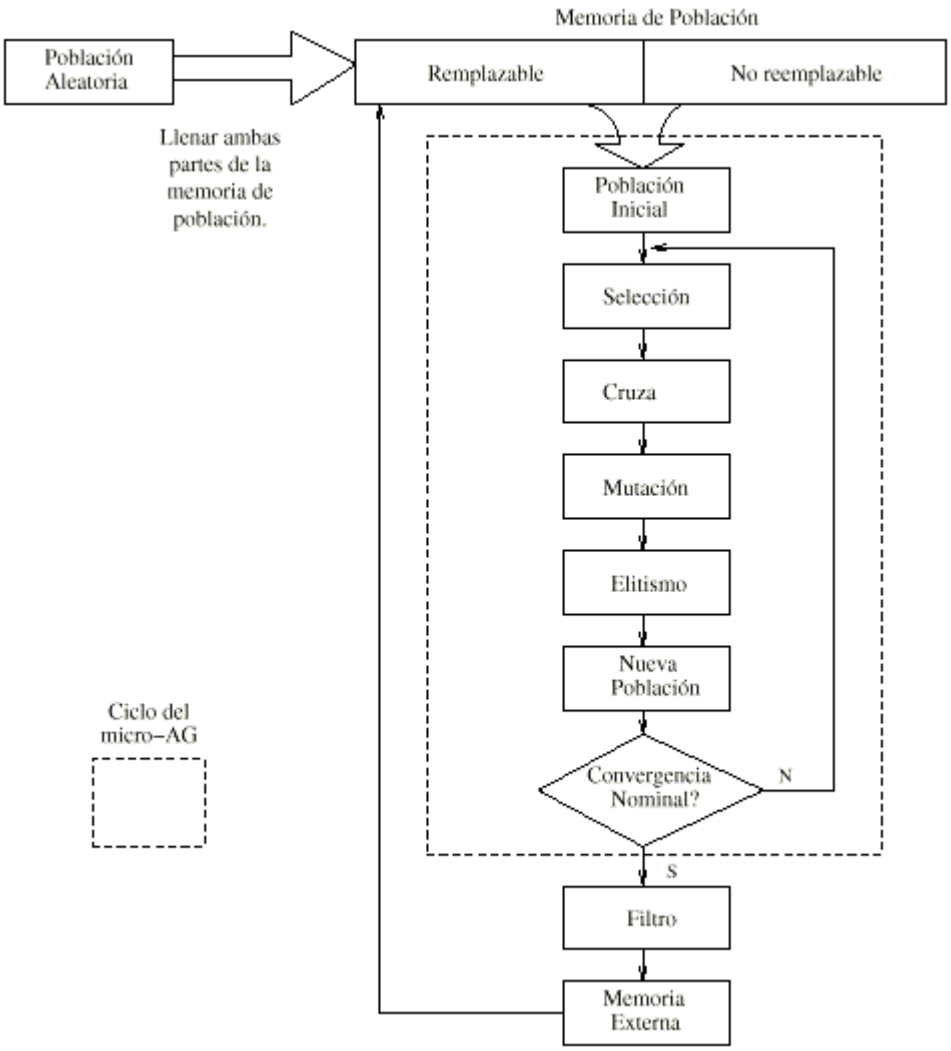


Figura 4.1 Micro Algoritmo Genético Multiobjetivo

Al iniciar el algoritmo se llena la memoria de población P, con individuos generados aleatoriamente. Esta memoria esta dividida en 2 partes: P^r población reemplazable y P^{nr} población no reemplazable. De esta memoria, el MAGM tomará de forma aleatoria una pequeña cantidad de individuos, los cuales serán nuestra población de trabajo P^t para irla evolucionando. Luego de la ejecución del MAGM durante un número de generaciones permitidas, se realiza un filtro para efecto del chequeo de no dominancia y así quedarse con 2 soluciones S_1, S_2 no dominadas. Estas 2 soluciones obtenidas, son comparadas contra un par de individuos de P^r. Si resultan no dominadas, ocupan su lugar en P^r y se anexan a la malla adaptativa (si cumplen con el elitismo). Este ciclo se irá repitiendo según el número de iteraciones dado.

A continuación, se describe el funcionamiento de cada una de las fases estratégicas que integran la técnica [35].

4.2.1 Memoria de población

La memoria de población (P) es generada inicialmente de forma aleatoria. Está dividida en dos partes: una porción reemplazable (P^r) y una no reemplazable (P^{nr}). Se estableció de esta manera, con el fin de que el MAGM no pierda diversidad, o bien, no converja hacia una porción única del frente de Pareto o hacia un frente de Pareto local.

P^{nr} a diferencia de P^r, no sufrirá cambios durante la corrida del algoritmo una vez inicializada. Esto hará que mantenga la diversidad requerida en el algoritmo.

La población de trabajo (P^t) es tomada de P (con cierta probabilidad), para que así resulte de una combinación de ambas partes de la memoria y no se permita un sesgo de la población hacia cierta región en el espacio de búsqueda, lo que dará la diversidad deseada a la población.

4.2.2 Cuerpo del MAGM

La evolución de P^t por parte del MAGM es realizada de manera tradicional, hasta que llegue a la convergencia nominal del proceso. En el caso de esta técnica la

convergencia nominal está dada por el número de generaciones máximas para el cuerpo del MAGM. Se utilizan los operadores genéticos convencionales, mismos que se describen a continuación.

Selección

Se emplea selección por torneo binario, utilizando dominancia como método de comparación. Su algoritmo es el siguiente:

- Barajar los individuos de la población
- Escoger *n* individuos (típicamente 2)
- Compararlos con base en su aptitud
- El ganador del torneo es el individuo más apto
- Se debe barajar la población, hasta seleccionar N padres

Cruza

Se utilizó la cruza de dos puntos (misma que se expuso en el capítulo 2) por ser la menos disruptiva de las de *n* puntos y evita algunos de los problemas de la cruza de un solo punto.

Mutación

Se utiliza mutación uniforme, lo que significa que el porcentaje de mutación utilizado no cambia a lo largo del proceso evolutivo (ver capítulo2).

Elitismo

Solamente un individuo no dominado es arbitrariamente seleccionado de la población en cada generación y copiado intacto a la siguiente.

4.2.3 Filtro

Aquí es donde observamos la condición elitista en la que se basó el desarrollo de la técnica. No solo deja pasar a los individuos no dominados S_1 y S_2 (si hay más

de un individuo no dominado) hacia un archivo histórico, sino que también verifica si tomarán su lugar en la memoria reemplazable, dado que dominen a sus correspondientes alojadas en P^r. De esta manera, P estará obteniendo una mejor oportunidad de converger hacia el frente de Pareto global.

También realiza otro tipo de elitismo sobre un cierto intervalo preestablecido. El ciclo de reemplazo toma soluciones no dominadas correspondientes a todas las regiones del frente de Pareto obtenido hasta el momento. Luego la memoria reemplazable es renovada con estos puntos. La diferencia entre el elitismo anterior y éste, es que el primero no garantiza que la memoria reemplazable tenga soluciones no dominadas globalmente y éste sí. Este segundo elitismo sólo se deberá aplicar en aquellas ocasiones en que los individuos del frente de Pareto sean al menos los mismos que en el conjunto P^r, pues de otra manera se puede llegar a una convergencia prematura.

4.2.4 Malla Adaptativa

Después de que el MAGM haya terminado un ciclo, se escogen los individuos no dominados de la población final, y se comparan uno por uno contra el frente de Pareto actual alojado en la memoria externa, pudiéndose presentar uno de los siguientes casos:

- Si no hay individuos en el archivo externo, entonces la solución actual es aceptada.
- Si la nueva solución es dominada por algún individuo del archivo externo, será descartada.
- Si ninguno de los individuos del archivo externo lo domina entonces será almacenado y se eliminan a los individuos dominados.
- Si el archivo ha alcanzado el máximo de soluciones permitidas, se invoca a la malla adaptativa.

Para mantener diversidad en el frente de Pareto, se usa una malla adaptativa similar a la propuesta por Knowles y Corne. El propósito es poder mantener las soluciones no dominadas de una manera uniforme y distribuida a lo largo del frente de Pareto. Una vez que el archivo donde se guardan las soluciones llega a

su límite, se dividen las soluciones no dominadas en diferentes regiones. De esta manera sólo se aceptarán individuos pertenecientes a regiones menos pobladas, o bien, pertenecientes a una región fuera de los límites previamente especificados para la malla.

La malla adaptativa es un espacio formado por hipercubos *k*-dimensionales, los cuales tienen tantas componentes como funciones objetivo *k* existan. Cada hipercubo es una región geográfica que puede contener un número no determinado de individuos. La cantidad de individuos trata de abarcar de manera uniforme la mayor cantidad de hipercubos posible.

A continuación, se despliega el pseudocódigo de la malla adaptativa.

```
function Malla_adaptativa(solución)
    begin
        solución1= x; x ∈ la región más poblada P
            if solución está fuera de rango then
                elimina solución
                almacenar solución en E
                actualiza localidades
            else if solucion1 en región más poblada que solución then
                elimina solución1
                almacenar solución en E
                actualiza localidades a la que pertenecía solución1 y a la
                que pertenece solución
            end if
end function
```

Figura 4.2 Pseudocódigo de la malla adaptativa.

4.2.5 Restricciones

En esta técnica, las restricciones son aplicadas en la selección, en la malla adaptativa y en la memoria de población de la siguiente manera:

- En la selección, se implementan para saber qué individuo tiene mejor aptitud para ser seleccionado; esto es, al comparar dos individuos primero se comparan por el número de restricciones violadas y el dominante será el que menos restricciones viole. Si ambos individuos violan igual número de restricciones, entonces se procede a realizar la comparación por medio de la dominancia de Pareto tradicional.
- En la malla adaptativa, no se guardará ningún individuo si algún individuo viola alguna restricción.

- En la memoria de población, éste puede ser alojado en P^r aún cuando viole alguna restricción (si domina a su competidor), para poder evolucionar y a la vez contribuya a llegar a una mejor calidad de soluciones.

```
function Micro-AG
    begin
        generar la población P de tamaño N y guardar su contenido en la
        memoria de población de tamaño M
        /*ambas porciones de M serán llenadas con soluciones aleatorias*/
        i = 0
        while i < Max do
            begin
                obtener la población de trabajo (P^t) de M
                repeat
                    begin
                      aplicar selección por torneo binario basado en no
                      dominancia
                      aplicar cruza de dos puntos y mutación uniforme a los
                      individuos seleccionados
                      aplicar elitismo (reteniendo solamente uno de los
                      vectores no dominados)
                      producir la próxima generación
                    end
                until convergencia nominal es alcanzada
                malla-adaptativa(vectores no dominados de P^t)
                copiar dos vectores no dominados de P^t a M
                if i mod ciclo de reemplazo
                then aplicar la segunda forma de elitismo
                i = i + 1
            end while
end function
```

Figura 4.2 Pseudocódigo del MAGM.

CAPITULO 5

5. Desarrollo del Applet

"Java, es entre muchas cosas: un lenguaje, una arquitectura, una plataforma para aplicaciones..." Nathan Meyers.

5.1 Java

5.1.1 Historia

Java fue concebido por James Gosling, Patrick Naughton, Chris Warth, Ed Frank y Mike Sheridan en Sun Microsystems Inc. en 1991. Este lenguaje se llamó inicialmente Oak, pero adoptó el nombre de Java en 1995.

El impulso inicial de Java no fue en Internet. La motivación principal fue la necesidad de un lenguaje portable, que sea independiente de la plataforma (es decir, con arquitectura neutral) que se pudiese utilizar para crear software para diversos dispositivos electrónicos, como hornos de microondas y controles remotos.

Durante la elaboración de los detalles de Java, apareció un segundo y más importante factor el cual daría un papel importante al lenguaje Java, el World Wide Web (la red mundial). A partir de entonces, Java ha sido impulsado al frente del diseño de los lenguajes de programación, ya que la red exigía programas portables.

Java se basa en C y C++. Sin embargo sería un gran error el pensar que Java es una versión de C++ para Internet. Tampoco fue diseñado para sustituir a C++. Java fue diseñado para resolver una serie de problemas y C++ para resolver otros problemas diferentes. Por esto, se seguirá dando la coexistencia de ambos lenguajes.

5.1.2 Applets y Aplicaciones Java

Pensando en Internet, esta característica es crucial para Java, ya que la red conecta ordenadores muy distintos. En cambio C++, es independiente de la plataforma sólo en código fuente, lo cual significa que cada plataforma diferente debe proporcionar el compilador adecuado para obtener el código máquina que tiene que ejecutarse.

Java incluye dos elementos: un compilador y un intérprete. El compilador produce un código de bytes que se almacena en un archivo para ser ejecutado por el intérprete Java denominado Máquina Virtual de Java.

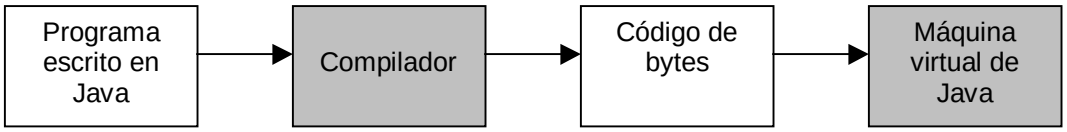


Fig. 5.1 Secuencia para obtener una aplicación Java.

Los códigos de bytes de Java son un conjunto de instrucciones correspondientes a un lenguaje máquina que no es específico de ningún procesador, sino de la Máquina Virtual de Java.

Casi todas las compañías de sistemas operativos y de navegadores, han implementado máquinas virtuales según las especificaciones publicadas por Sun MicroSystems, para que sean compatibles con el lenguaje Java. Para las aplicaciones de Internet (applets) la máquina virtual está incluida en el navegador y para las aplicaciones Java convencionales, puede venir con el sistema operativo, con el paquete Java, o bien puede obtenerla a través de Internet.

Una applet es una aplicación diseñada para ser transmitida por Internet y ejecutada en un navegador web compatible con Java. Una applet es un pequeño programa Java que se transfiere dinámicamente a través de la red, como si fuese una imagen, un archivo de sonido o de video. La diferencia principal es que una applet es un programa inteligente, no sólo una animación o un archivo de sonido. En otras palabras, es un programa que puede reaccionar ante las acciones del usuario y cambiar dinámicamente, no sólo ejecutar repetidamente la misma animación.

5.1.3 ¿Por qué Java?

Java no sólo ofrece portabilidad para sus aplicaciones, sino también otras características, tales como:

- Seguridad: Es posible transmitir los applets de forma segura sin temor a ser infectados por un virus o a recibir intentos de acceso maliciosos (como recoger información privada, como números de tarjeta de crédito). Esto se consigue limitando el programa a un entorno de ejecución Java y no permitiendo que acceda a otras partes de la computadora.
- Simple: Teniendo conocimientos previos de programación orientada a objetos, resulta fácil aprender este lenguaje. Aunado a ello, si se tiene conocimientos de programación en el lenguaje C++, dado que Java hereda sintaxis de este lenguaje, se facilita más el aprender este lenguaje.
- Orientado a objetos: Aunque tiene influencia de sus lenguajes predecesores, no fue diseñado para tener un código fuente compatible con otros lenguajes. Java se basa en ideas contemporáneas de orientación a objetos, consiguiendo un equilibrio entre el modelo purista (todas las cosas son objetos) y el modelo pragmático (quédate fuera de mi camino).
- Robusto: Es un lenguaje de tipografía rígida, por lo que permite una verificación exhaustiva al tiempo de la compilación, en busca de posibles problemas de no concordancia de letras. Requiere declaraciones explícitas de método, esto asegura que el compilador atrape errores en la invocación

del método, con lo que consigue programas más confiables. Una de las cosas que hace más simple a Java, es la carencia de punteros y aritmética de puntero. Esta característica también incrementa la robustez de los programas.

- Multihilo: Cumple el requisito del mundo real de crear programas interactivos en red. Dispone de herramientas para sincronizar múltiples procesos que hacen posible construir fácilmente sistemas interactivos.
- Dinámico: En todo momento, una clase se puede cargar en un intérprete de Java en ejecución. Las bibliotecas de código nativo, también se pueden cargar de manera dinámica. Lo que es de gran valía para la solidez de las applets, es que pequeños fragmentos de código binarios pueden ser actualizados dinámicamente en un sistema que está ejecutándose.
- Distribuido: Fue diseñado para el entorno distribuido de Internet, por lo que trabaja con el protocolo TCP/IP. Dispone de interfaces para que los objetos puedan ejecutar procedimientos de forma remota en un paquete de invocación de método remoto (RMI, remote method invocation). Esta característica aporta un nuevo nivel de abstracción en la programación cliente/servidor.

Con base en estas características, se optó por este lenguaje de programación para el desarrollo de una aplicación web del Micro Algoritmo Genético Multiobjetivo.

Esta aplicación tiene la finalidad de mostrar en un entorno gráfico, la diversidad de soluciones que se pueden obtener en el frente Pareto, ejecutando un algoritmo genético que trabaja con una población pequeña, usando optimización evolutiva multiobjetivo.

5.1.4 Requerimientos

Para la elaboración de esta aplicación web, se empleó un entorno de desarrollo Java. Este entorno, se obtuvo de la página de Sun MicroSystems Inc., http://www.sun.com. Siendo éste: Java™ 2 SDK, Standard Edition, Version 1.3.0. Tratándose del JDK 1.3 (Java Development Kit) para la plataforma Windows. Como editor de código fuente, usé Kawa Versión 4.01a de Tec-tools Inc.

En cuanto al hardware, el mínimo requerido es: Pentium a 166MHz o superior; 32 Mb de RAM para correr las aplicaciones Java, y 48 Mb son recomendados para ejecutar applets, muchos programas extensos requerirán de más RAM para un mejor desempeño; 65 Mb libres de disco duro antes de instalar Java™ 2 SDK son suficientes, pero si se baja e instala la documentación por separado, se necesitarán 120 Mb adicionales de disco duro.

5.2 Desarrollo del Applet

5.2.1 Algoritmo General

```
individuo poblacion = generaPoblacion(M, true, false, false);
individuo poblacionNueva = generaPoblacion(M, true, false, false);
PARETO = generaPoblacion(tamPARETO, false, false, false);
memoria = generaPoblacion(tamMem, true, true, true);

int m1 = 0, m2 = 0;
for (int iter = 0; iter < iTot; iter++)
{
        obtPoblacion(poblacion);
        for (int conta = 0; conta < gMax; conta++)
        {
                seleccion(poblacion, poblacionNueva);
                cruzaMutacion(poblacionNueva);
                if (iter < (sndElit * 2))
                        elitismo(poblacion, poblacionNueva);
                nuevaGeneracion(poblacion, poblacionNueva);
        }
        mejores(m1, m2, poblacion);
        memorizarMejores(m1, m2, poblacion);
        guardaPobEnPareto(m1,m2);
        sdoElitismo(iter+1);
}
```

En la figura anterior podemos ver el recorrido completo del algoritmo general, primeramente inicia los objetos: población de trabajo (*poblacion*), población generada después de aplicar operadores a la población de trabajo (*poblacionNueva*), conjunto de Pareto (*PARETO*) y memoria que tendrá la labor de dar diversidad a la población (*memoria*), los cuales son de tipo individuo. La clase *individuo* contiene tres atributos: genotipo (*genotipo*), aptitud de la función objetivo (*aptObj*) y localidad ocupada en la malla adaptativa (*loc*). Esta clase, se define de la siguiente manera:

```
class individuo
{
        static byte[][] genotipo;
        static double[][] aptObj;
        static long[] loc;

        public static void iniciaArreglos(int dimension)
        {
                genotipo = new byte [dimension][prueba.bits];
                aptObj = new double [dimension][prueba.nObjs];
                loc = new long [dimension];
        }
}
```

Fig. 5.3 Código en Java de la definición de clase individuo.

Después de iniciada la memoria de población, se inicia el ciclo que encierra toda la corrida del algoritmo. Este ciclo será por *itot* veces (número de iteraciones). Este parámetro hace referencia a la condición de paro del algoritmo y varía de acuerdo a la complejidad del mismo.

Dentro del ciclo de iteraciones del algoritmo, primero inicio la memoria de trabajo, *poblacion*. Se abre otro ciclo, el cual es el cuerpo del Micro-AG, éste correrá para el número de generaciones máximas dadas (*gmax*), donde *gmax* es el encargado de manejar la convergencia nominal. Al terminar el cuerpo del Micro-AG se aplica el Filtro llamando a los métodos: *mejores* y *memorizarMejores*. Por último se guarda en la malla adaptativa, *PARETO,* los individuos no dominados haciendo uso del método *guardaPobEnPareto* y se aplica el segundo elitismo por medio del método *sdoElitismo*.

5.2.2 Memoria de población

Para iniciar la memoria de población (*memoria*), se llama al método *generaPoblacion* con los cuatro parámetros siguientes: tamaño de la población (*tam*), se inciará o no el genotipo (*iniciar*), se evaluará o no para guardar su aptitud (*evaluar*) y se detectará o no si es un individuo no dominado para almacenar en *PARETO* (*lpareto*). El método *generaPoblacion* iniciará la población con *tam* individuos, generando de manera aleatoria sus genotipos, los evaluará y si resultan ser no dominados los almacenará en *PARETO*.

```
public static individuo generaPoblacion(int tam, boolean iniciar, boolean evaluar, boolean lpareto)
{
	individuo pueblo = new individuo();
	pueblo.iniciaArreglos(tam);
	if (iniciar)
	{
		int j;
		for (int i=0; i<tam; i++)
		{
			for (j=0; j<bits; j++)
			{
				pueblo.genotipo[i][j] = (byte) funRandom(base);
			}
			pueblo.loc[i] = 0;
			if (evaluar)
			{
				evaluaFuncion(pueblo.genotipo[i], pueblo.aptObj[i]);
			}
			else
				for (j=0; j<nObjs; j++) pueblo.aptObj[i][j] = 0;
				if (lpareto)
				funPareto(pueblo.genotipo[i], pueblo.aptObj[i], pueblo.loc[i]);
		}
	}
	return pueblo;
}
```

Fig. 5.4 Código en Java de la generación de la memoria de población.

5.2.3 Cuerpo del MAGM

El cuerpo del MAGM es un ciclo que va de 1 hasta *gmax*, este parámetro es el encargado de manejar la convergencia nominal. Dentro de este ciclo se aplican los operadores genéticos tradicionales: selección, cruza, mutación y elitismo.

Selección

Este método recibe dos parámetros población de trabajo (*pp*), y población actual seleccionada en la generación i (*ppn*). Para hacer la selección por torneo binario se usa la dominancia como método de comparación, misma que se encuentra en el método *domina.* Este método compara el elemento p[l] contra p[l+1] y devuelve: 1 si el primero domina al segundo, -1 si el segundo domina al primero, 0 si son iguales y 11 si ninguno domina.

```
public static void seleccion(individuo pp, individuo ppn)
{
	int[] ind = new int[M*2];
	for (int i = 0; i < (M*2); i++) ind[i] = funRandom(M);
	int l = 0, j;
	while (l < (M*2))
	{
	switch (domina(pp.genotipo[ind[l]], pp.aptObj[ind[l]], pp.genotipo[ind[l+1]],
pp.aptObj[ind[l+1]]))
		{
			case 11:
				if (funRandom(2) == 1)
					copia(ppn[(int) Math.floor(l/2)], pp[ind[l]]);
				else
					copia(ppn[(int) Math.floor(l/2)], pp[ind[l+1]]);
				break;
			case 1:
			case 0:
				copia(ppn[(int) Math.floor(l/2)], pp[ind[l]]);
				break;
			case -1:
					copia(ppn[(int) Math.floor(l/2)], pp[ind[l+1]]);
				break;
		}
		l += 2;
	}
}
```

Fig. 5.5 Código en Java de la selección por torneo binario.

Cruza y Mutación

En el método *cruzaMutacion*, se recorre de dos en dos individuos la población generada de la selección.

Primero se verifica según el porcentaje de cruza, *Pc,* si se aplica o no este operador. En caso de que así proceda, se cruzarán los individuos ppn[i] con ppn[i+1], con el método a dos puntos.

Después se verificará de acuerdo al porcentaje de mutación, *Pm*, si se aplica el porcentaje de mutación de forma separada en los individuos ppn[i] y ppn[i+1].

```
public static void cruzaMutacion(individuo ppn)
{
	byte temp;
	int punto0, punto1; //puntos de cruza
	for (int i = 0; i < M; i+=2)
	{
		if (Math.random() < Pc)
		{
			punto0 = funRandom(bits);
			do
				punto1 = funRandom(bits);
			while (punto0 == punto1);
			if (punto0 > punto1)
			{
				int aux = punto0;
				punto0 = punto1;
				punto1 = aux;
			}
			for (int j = punto0; j <= punto1; j++)
			{
				temp = ppn.genotipo[i][j];
				ppn.genotipo[i][j] = ppn.genotipo[i+1][j];
				ppn.genotipo[i+1][j] = temp;
			}
		}
		muta(ppn.genotipo[i]);
		muta(ppn.genotipo[i+1]);
		evaluaFuncion(ppn.genotipo[i], ppn.aptObj[i]);
		evaluaFuncion(ppn.genotipo[i+1], ppn.aptObj[i+1]);
	}
}
```

Fig. 5.6 Código en Java de los operadores Cruza y Mutación.

Elitismo

El operador Elitismo, tomará al primer individuo no dominado de la población pp, la cual resultó de la selección (población padre de la generación actual). Luego cada individuo de la población ppn (población descendiente de la generación actual) se

comparará con aquel padre seleccionado, *pp[mejor]*. Si algún descendiente resulta dominado, entonces será reemplazado por el padre, esto es, pasará intacto a la siguiente generación.

```
public static void elitismo(individuo pp, individuo ppn)
{
	int mejor = elMejor(pp);
	int l = 0, dominante = 0, j;
	while (l < M)
	{
		dominante = domina(pp.genotipo[mejor], pp.aptObj[mejor], ppn.genotipo[l],
ppn.aptObj[l]);
		if (dominante == 0) break;
		else if (dominante == 1)
		{
			copia(ppn[l], pp[mejor]);
			break;
		}
		l++;
	}
}
```

Fig. 5.7 Código en Java de la selección por torneo binario.

5.2.4 Filtro

El filtro resultó de la condición elitista que fundamenta al algoritmo. Se presenta en dos métodos. El primer método en ejecutarse es *mejores*, el cual memoriza los mejores individuos de la generación presente, de tal forma que memoriza primero al mejor, y si hay otro en la población, que sea diferente a éste y que el primero no domine a este individuo, entonces memoriza también al segundo, guardando la posición de estos dos individuos en las variables enteras *m1* y *m2*. Luego se ejecuta el segundo método *memorizarMejores*. Esta segunda función memoriza las soluciones resultantes del método anterior (*m1* y *m2*) en la población *memoria*, de tal manera que sólo introduce en la primera mitad de la memoria (memoria reemplazable) la cual generará la diversidad en la población.

```
public static void mejores(int m1, int m2, individuo pp)
{
	m2 = -1;
	m1 = elMejor(pp);
	int l = 0, bandera = 1;
	while (l < M && bandera == 1)
	{
		if (m1 != l)
		{
		switch (domina(pp.genotipo[m1], pp.aptObj[m1], pp.genotipo[l], pp.aptObj[l]))
			{
				case 11:
				case -1:
					m2 = l;
					bandera = 0;
					break;
				case 1:
				case 0:
					break;
			}
		}
		l++;
	}
}
```

Fig. 5.8 Código en Java que memoriza dos mejores individuos.

```
public static void memorizarMejores(int m1, int m2, individuo pp)
{
	memoriza(pp.genotipo[m1], pp.aptObj[m1], pp.loc[m1]);
	if (m2 != -1)
		memoriza(pp.genotipo[m2], pp.aptObj[m2], pp.loc[m2]);
}
```

Fig. 5.9 Código en Java que guarda los mejores individuos en memoria.

5.2.5 Malla Adaptativa

La malla adaptativa involucra varios métodos. Primero llama al método *guardaPobEnPareto*, el cual manda individuo a individuo de los ya memorizados por el método *mejores*, para ver si son miembros de la población *PARETO*. Si es alojable pero el límite de sus funciones no le permite decidir, entonces entra en función la malla adaptativa para poder almacenar dicho individuo. A continuación se tiene el código del método funPareto, en el cual se puede apreciar el funcionamiento de la malla.

```
public static void funPareto(byte[] cromosoma, double[] aptitud, long localidad)
{
	int mPoblado, r;
	if (restricciones(cromosoma) > 0) return;  //No guarda en PARETO si viola restricciones
	if (posPARETO == 0)     //verifica si está vacío el conjunto de Pareto
	{
		gPARETO(cromosoma, aptitud, localidad);     //guarda solución en PARETO
		return;
	}
	if (dominaAPareto(cromosoma, aptitud, localidad) == 0)
	{//nadie domina al individuo
		if(posPARETO+1 < tamPARETO)
			gPARETO(cromosoma, aptitud, localidad);
		else if (posPARETO+1 == tamPARETO)
		{
			gPARETO(cromosoma, aptitud, localidad);
			actualizaLoc();     //Actualiza localidades en la malla adaptativa
		}
		else
		{
			mPoblado = buscaMasPoblado();
			r = (int) cualLoc(aptitud);     //asigna una localidad
			if (r == -1)
			{
				int i = buscaIndConLoc(mPoblado);
				int j;
				for (j=0;j<bits;j++) PARETO.genotipo[i][j] = cromosoma[j];
				for (j=0;j<nObjs;j++) PARETO.aptObj[i][j] = aptitud[j];
				PARETO.loc[i] = localidad;
				actualizaLoc();
			}
			else if (masPoblado(mPoblado, r) == 1)
			{
				int i = buscaIndConLoc(mPoblado);
				localidad = r;
				int j;
				for (j=0;j<bits;j++) PARETO.genotipo[i][j] = cromosoma[j];
				for (j=0;j<nObjs;j++) PARETO.aptObj[i][j] = aptitud[j];
				PARETO.loc[i] = localidad;
				hipercubo[mPoblado]--;
				hipercubo[r]++;
			}
		}
	}
}
```

Fig. 5.10 Código en Java de la malla adaptativa.

5.3 Conclusiones

Los AGs fueron inspirados en la teoría de la evolución Darwiniana. Son algoritmos matemáticos de optimización de propósito general basados en mecanismos naturales de selección y genética, usados en la resolución de problemas de optimización y búsqueda.

La importancia de este tipo de algoritmos en la actualidad se debe a que facilitan la solución de problemas con muy poca información, además de que ayudan en la optimización de ciertas funciones sumamente complejas. Por lo tanto, se puede destacar que sus principales objetivos son: mejorar la calidad de sus soluciones y aumentar la velocidad de ejecución.

Los principales usos de los AGs son vistos en la parametrización de sistemas, búsqueda de reglas en juegos, enrutamientos, resolución de sistemas de ecuaciones no lineales, optimización (estructural, de topologías, numérica, combinatoria, etc.), aprendizaje de máquina, bases de datos (optimización de consultas), reconocimiento de patrones, generación de gramáticas (regulares, libres de contexto, etc.), planeación de movimientos de robots, predicción de series temporales.

Los AGs también son aplicados para lidiar con problemas con múltiples objetivos. La investigación en esta área se originó a partir de los trabajos presentados por Rosenberg a finales de los 1960s. De hecho existen varias técnicas ya desarrolladas para la resolución de este tipo de problemas. Estas técnicas se clasifican en dos grandes grupos: tradicionales (Investigación de Operaciones) y alternativas (algoritmos evolutivos, recocido simulado, algoritmo de Monte Carlo, búsqueda tabú entre otras).

El MAGM desarrollado por el Dr. Carlos Coello y el MC Gregorio Toscano, es una mejora de los AGs tradicionales. Es una técnica alternativa la cual trabaja con

pocos individuos para reducir el chequeo de no dominancia, con una malla adaptativa la cual hace posible tener una mejor diversidad en el conjunto de Pareto, y con fundamentos en la convicción elitista con el fin de lograr soluciones de mejor calidad. Con estas tres características fundamentales del MAGM, se logra reducir el costo computacional y obtener soluciones de calidad en menor tiempo que los algoritmos genéticos que trabajan con técnicas convencionales.
Al término del presente trabajo, se logró tener una aplicación web del MAGM, con la que podemos apreciar de una manera interactiva la forma de producir una diversidad de soluciones de buena calidad en un tiempo considerable.

El desarrollo de este trabajo de tesis, contó con el apoyo del proyecto CONACyT titulado “Nuevos Paradigmas en Optimización Evolutiva Multiobjetivo” (referencia 34201-A) cuyo responsable es el Dr. Carlos A. Coello Coello. Esta aplicación para internet es parte de los objetivos del proyecto antes citado.

También resulta de apoyo al proyecto “Paralelización de Algoritmos Genéticos” del cuerpo académico de la FMAT-UADY, cuyo responsable es el MC Luis Fernando Curi Quintal, pues se logra la implementación de un AG en programación lineal el cual está siendo analizado para ser implementado en un clúster en Linux.

En Mérida, Yucatán, se ha manifestado el interés hacia esta área de computación. De manera particular, el CINVESTAV-IPN está proponiendo la colaboración de FMAT mediante el uso de los AGs para obtener una mayor eficiencia en los resultados del análisis de las aguas subterráneas de la Península.

Aquí pude observar cómo la ciencia se complementa de diversas áreas, logrando así, un paso más hacia la innovación de nuevas técnicas para la resolución de problemas del mundo real a bajo costo computacional. Esta es una clara evidencia de la sinergia.

BIBLIOGRAFÍA

Bibliografía referente a Computación Evolutiva

[1] Coello Coello, Carlos A. “Notas del curso: Introducción a la Computación Evolutiva”, UADY, Noviembre de 2000.

[2] García Castellano, Francisco J. “Paralelización de una librería genérica de algoritmos evolutivos usando MPI”, Universidad de Granada, 2000.

[3] Jimenez Barrionuevo, Fernando “Computación Evolutiva”, Universidad de Murcia, Departamento de ingeniería de la información y las comunicaciones, 2001.

[4] Perez Serrada, Anselmo "Una introducción a la Computación Evolutiva”, marzo de 1996.

[5] Toscano Pulido, Gregorio “Computación Evolutiva: Un breve repaso histórico”, en XXIII Congreso Industrial de Ingeniería Industrial, ITM, Noviembre de 2001.

[6] http://www.ica.ele.puc-rio.br/pesquisa/ “Computación Evolucionária”, Inteligencia Computacional Aplicada PUC-Rio, Agosto de 2001.

[7] http://www.aircenter.net/gaia/ce/ceno_c.htm Herrán Gascón, Manuel; “Notas sobre Computación Evolutiva”, Agosto de 2001.

[8] http://www-cs.us.es/~delia/sia/html98-99/pag-alumnos/web12/principal.html, “Programación Genética”, Agosto de 2001.

[9] http://www.terra.es/personal/cxc_9747/home.html “De Evolutionibus Pasado, Presente y Futuro de una Revolución Científica”, Agosto de 2001.

Bibliografía referente a Algoritmos Genéticos

[10] Coello Coello, Carlos A. “Algoritmos Genéticos y sus Aplicaciones”.

[11] Colli Rivas, Lina M. “Tesis: Algoritmos Genéticos y su Aplicación a un Problema con Objetivos Múltiples”, Febrero de 1998.

[12] Holland, John H. “Adaptation in Natural and Artificial Systems”, University of Michigan Press, 1975, 211 p.

[13] Koza, John R. “Genetic Programming. On the Programming of Computers by Means of Natural Selection”, The MIT Press, 1992, 819 p.

[14] http://ccp.servidores.net/genetico.html Costa Portela, Carlos “Algoritmos Genéticos”, Junio de 2001.

[15] http://udlapvms.pue.udlap.mx/~is103380/AlgGeneticos/ARTRD.HTM Mendez Rosas, Miguel “Algoritmos Genéticos”, Junio de 2001.

[16] http://www.geocities.com/CapeCanaveral/9802/3d5ca000.htm Larrañaga, Pedro “Algoritmos Genéticos”, Departamento de Ciencias de la Computación e Inteligencia Artificial Universidad del País Vasco, Junio de 2001.

[17] http://www.iamnet.com/users/jcontre/genetic/ Contreras, J. “Tutorial sobre Algoritmos Genéticos”, Maracaibo, Junio de 2001.

[18] http://www.redcientifica.com/doc/doc199904260011.html Coello Coello, Carlos A. “Introducción a los Algoritmos Genéticos”, Junio de 2001.

[19] http://www.uv.es/~rmarti/genet.html Martí, Rafael “Algoritmos Genéticos”, Junio de 2001.

Bibliografía referente a Optimización Evolutiva con Objetivos Múltiples

[20] Coello Coello, Carlos A., An Updated Survey of GA-Based Multiobjective Optimization Techniques. Technical Report Lania-RD-98-08, Laboratorio Nacional de Informática Avanzada (LANIA), Xalapa, Veracruz, México, december 1998.

[21] Coello Coello, Carlos A. A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques, Knowledge and Information Systems, Vol. 1, No. 3, pp. 269-308, August 1999.

[22] Fonseca, C. M. y Flemming, P. J. “Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization”, En S. Forrrest (Editor), Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo, California, pp. 416-423, University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers, 1993.

[23] Grefenstette, J. J. “GENESIS: A System for Using Genetig Search Procedures”, Proceedingsof the 1984 Conference on Intelligente Systems and Machines, pp. 161-165, 1984.

[24] Horn, J., Nafpliotis, N., y Goldberg, D.E. (1994). A niched Pareto genetic algorithm for multiobjective optimization. In Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, Volume 1, 1994 (ICEC '94). Piscataway, NJ: IEEE Service Center, 82-87.

[25] Knowles, Joshua D. y Corne, David W. “Approximating of Nondominated Front Using the Pareto Archived Evolution Strategy”, Evolutionary Computation, 8(2):149-172, 2000.

[26] Luenberg, David E. “Programación Lineal y no Lineal”, Iberoamericana.

[27] Rios Insua, Sixto; Sixto Insua, David; Mateos, Alfonso y Martín, Jacinto “Programación Lineal y Aplicaciones. Ejercicios Resueltos”, AlfaOmega, Julio de 1998.

[28] Rodriguez Cotilla, Zoe “La toma de decisiones en las entidades agrarias”, Seminario Iternacional sobre Cooperativas, Universidad de la Habana, Febrero de 2000.

[29] Schaffer, J. D. “Multiple Objective Optimization with Vector Evaluated Genetic Algorithms”, en Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms, pp. 93-100, Lawrence Erlbaum, 1985.

[30] Serra de la Figueroa, Daniel “Métodos Cuantitativos para la toma de decisiones”, Universidad Pompeu Fabra.

[31] Toscano Pulido, Gregorio “La toma de decisiones mediante computación evolutiva, el estado del arte”, en XXIII Congreso Industrial de Ingeniería Industrial, ITM, Noviembre de 2001.

[32] Valenzuela Rendón, M. y Uresti Charre, E. (1996a). Un algoritmo genético no generacional para optimización multiobjetivo. XXVI Reunión de Intercambio de Experiencias en Investigacion y Desarrollo Tecnológico del Sistema ITESM, pp. 187-196, Monterrey, N.L., México, ITESM.

Bibliografía referente al Micro Algoritmo Genético Multiobjetivo

[33] Coello Coello, Carlos A. y Toscano Pulido, Gregorio, “A Micro-Genetic Algorithm for Multiobjective Optimization”, In Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, editors, First International Conference on Evolutionary Multi-Criterion Optimization, pages 126-140. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[34] Coello Coello, Carlos A. y Toscano Pulido, Gregorio, “A Micro-Genetic Algorithm for Multiobjective Optimization”, Lania-RI-2000-06, Laboratorio Nacional de Informática Avanzada, 2000.

[35] Toscano Pulido, Gregorio “Tesis: Optimización Multiobjetivo usando un Micro Algortimo Genético”, Universidad Veracruzana -LANIA, Septiembre de 2001.

Bibliografía referente al Desarrollo del Applet

[36] Ceballos, Francisco J. “Java 2, Curso de Programación”, AlfaOmega, Abril de 2001.

[37] Flanagan, David “Java en pocas palabras”, Mc Graw Hill, 2ª edición, Septiembre de 1998.

[38] Griffith, Steven W.; Chan, Mark C. y Iasi, Anthony F. “1001 Tips para programar con Java”, Mc Graw Hill, Noviembre de 1997.

[39] Meyers, Nathan “Programación Java en Linux”, Prentice Hall, 2000.

[40] Naughton, Patrick y Schildt, Herbert “Java, Manual de Referencia”, Mc Graw Hill, 1997.

[41] http://www.sun.com, Sun Microsystems Inc., Junio de 2001.

ANEXO

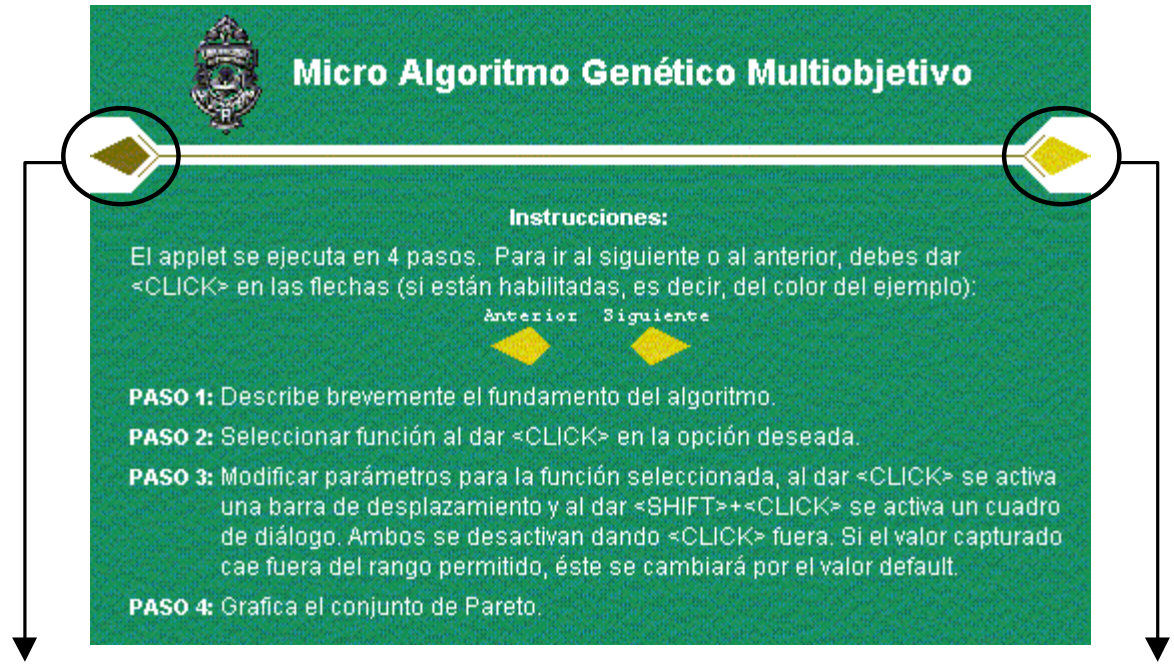
Manual de Usuario del Applet del MAGM

Para la elaboración de esta aplicación web, se empleó el entorno de desarrollo Java, el JDK 1.3 (Java Development Kit) para la plataforma Windows. Este software es gratuito y se puede obtener de la página de Sun MicroSystems Inc.

Para poder ejecutarlo se requiere como mínimo: un procesador Pentium a 166MHz o superior; 48 Mb de RAM; tener instalado algún navegador que soporte y que tenga activado la Máquina Virtual Java.

El applet desarrollado es una aplicación en la que podemos observar el desempeño del MAGM, a medida que modificamos sus parámetros iniciales. Se compone por una secuencia de pantallas, las cuales nos van guiando hasta la representación gráfica de los puntos del conjunto de Pareto de la función de prueba que resulte seleccionada.

Al iniciar la aplicación, se visualizará una pantalla de instrucciones, en la cual se comentan los pasos que se estarán realizando durante la ejecución del applet.
Para poder avanzar un paso se deberá dar un CLICK en la flecha que se encuentra del lado derecho y para retroceder se dará CLICK en la flecha del lado izquierdo. En caso de ser la primera pantalla, la flecha de la izquierda estará desactivada, esto se notará por la diferencia en el color de las flechas; de manera homóloga se verá desactivada la flecha de la derecha en la última pantalla.



Flecha para retroceder (desactivada) Flecha para avanzar (habilitada)

Al darle CLICK a la flecha para avanzar, nos presentará una breve descripción del fundamento básico del algoritmo.

La siguiente pantalla, nos presenta un menú con cinco opciones de funciones prueba, para seleccionar una de ellas que será nuestro problema a aplicarle el algoritmo. Las funciones son las siguientes:

Kursawe

Minimizar $f_1(\bar{x}) = \sum_{i=1}^{n-1}(-10\exp(-0.2\sqrt{x_i^2 + x_{i+1}^2}))$

Minimizar $f_2(\bar{x}) = \sum_{i=1}^{n}(|x_i|^{0.8} + 5sin(x_i)^3)$

donde: $-5 \le x_1, x_2, x_3 \le 5$

DEB Bimodal

Minimizar $f_1(x_1, x_2) = x_1$

Minimizar $f_2(x_1, x_2) = \dfrac{g(x_2)}{x_1}$

donde $g(x_2) = 2.0 - \exp\left\{-\left(\dfrac{x_2 - 0.2}{0.004}\right)^2\right\} - 0.8\exp\left\{-\left(\dfrac{x_2 - 0.6}{0.4}\right)^2\right\}$

sujeto a: $0.1 \le x_1 \le 1.0, \quad 0.1 \le x_2 \le 1.0$

Kita

Maximizar $f_1(x, y) = -x^2 + y$

Maximizar $f_2(x, y) = 0.5x + y + 1$

sujeto a : $\dfrac{1}{6}x + y - \dfrac{13}{2} \le 0, \quad \dfrac{1}{2}x + y - \dfrac{15}{2} \le 0,$

$\dfrac{5}{x} + y - 30 \le 0$

y $0 \le x \le 7.0, \quad 0 \le y \le 7.0$

Srinivas

Minimizar $f_1(x,y) = (x-2)^2 + (y-1)^2 + 2$

Minimizar $f_2(x,y) = 9x - (y-1)^2$

sujeto a : $0 \geq x^2 + y^2 - 225$

$0 \geq x - 3y + 10$

y $-20 \leq x, y \leq 20$

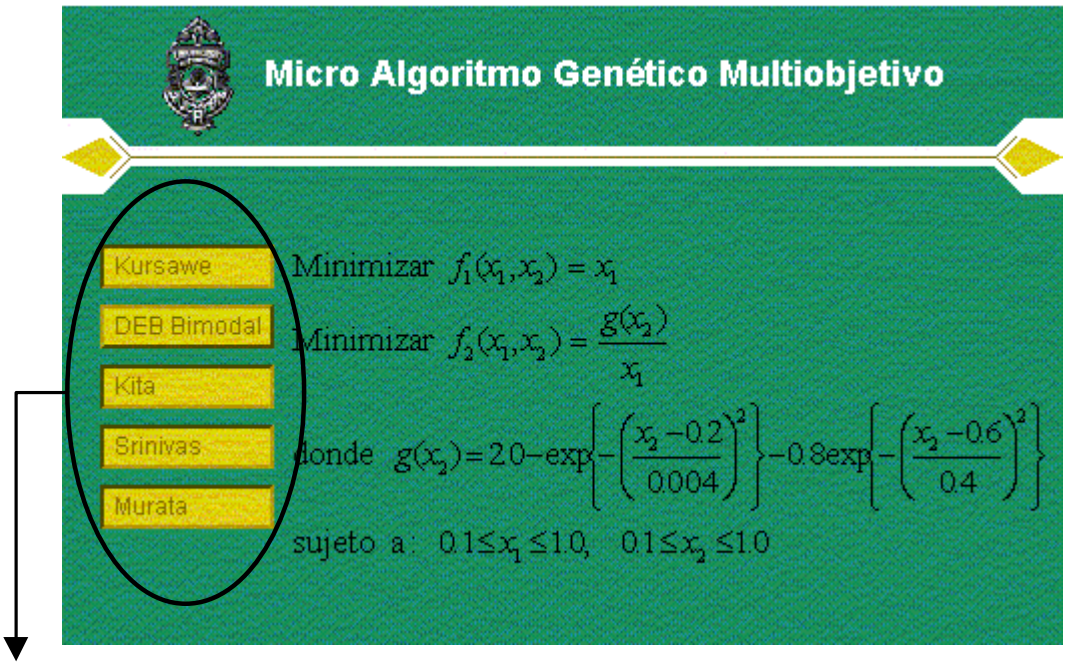
Murata

Minimizar $f_1(x,y) = 2\sqrt{x}$

Minimizar $f_2(x,y) = x(1-y) + 5$

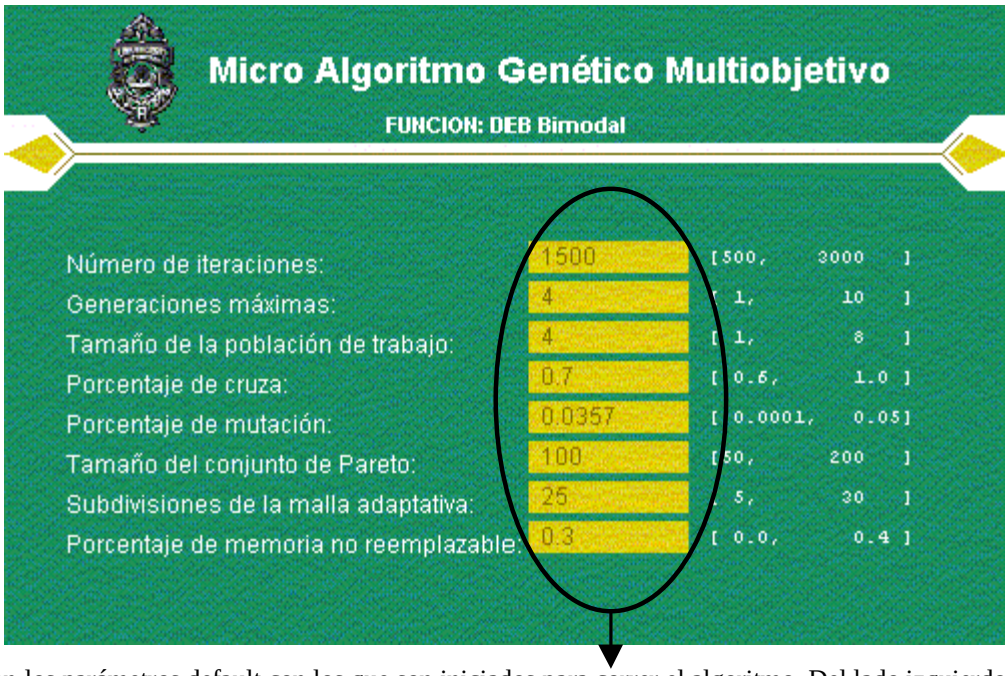
donde : $1 \leq x \leq 4$

$1 \leq y \leq 2$



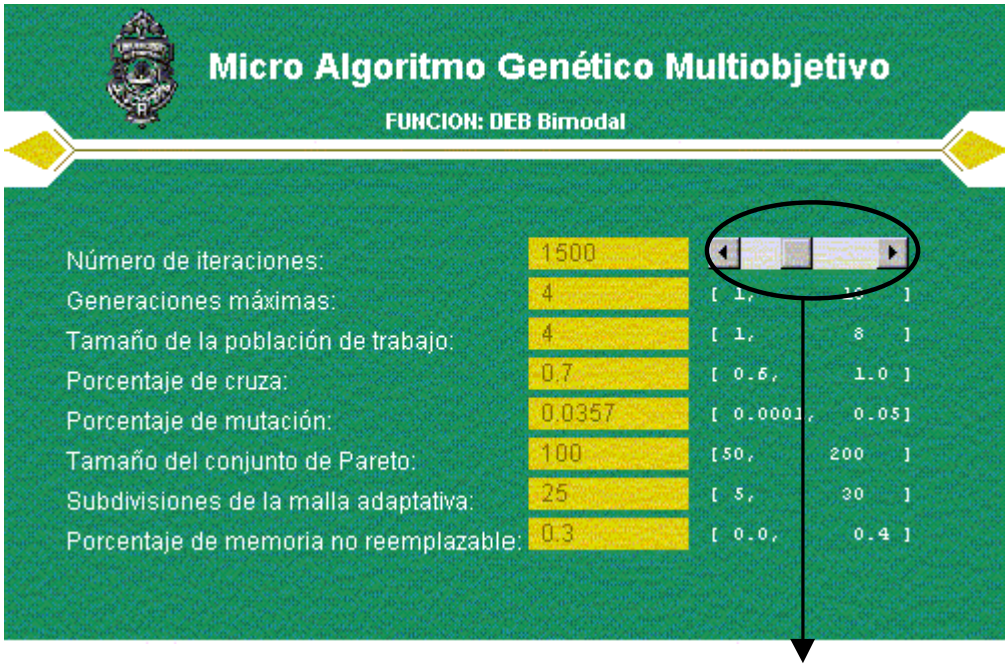
Menú para seleccionar la función de prueba a ejecutar. Como se puede observar, en este caso está seleccionada la función DEB Bimodal, pues el marco del botón resulta ser de un color diferente al de los otros. Para cambiar de selección basta con dar CLICK en la opción deseada.

Después de seleccionar nuestra función ahora podemos modificar los parámetros con los cuales correrá nuestro algoritmo. Es de notarse que a partir de ahora en la parte superior, nos será indicada que función de prueba fue seleccionada.



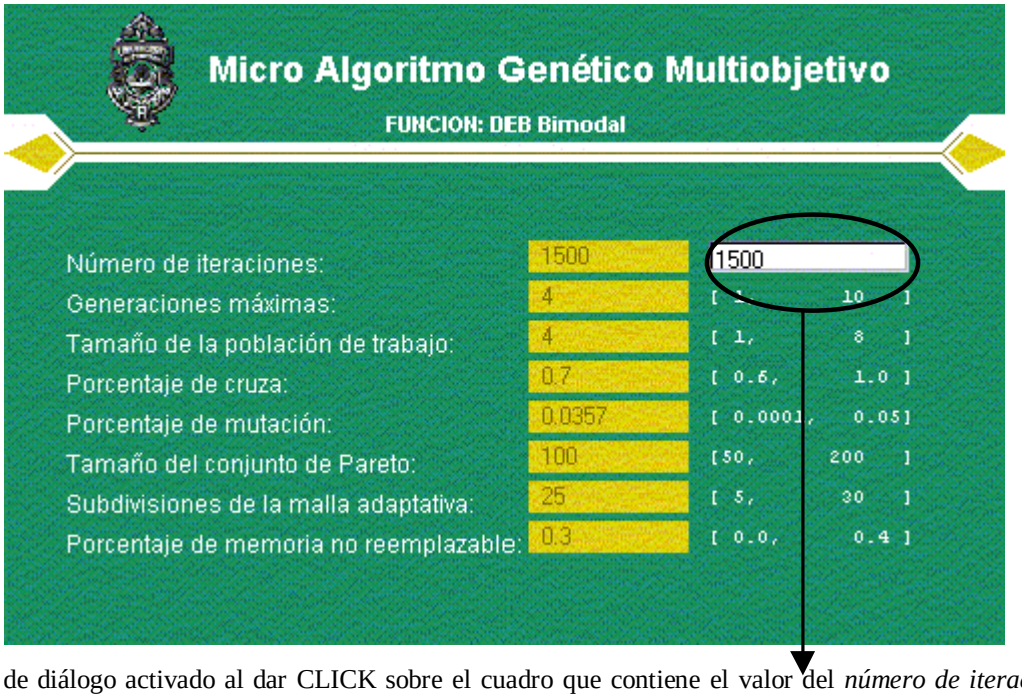
Estos son los parámetros default con los que son iniciados para correr el algoritmo. Del lado izquierdo vemos la descrpción y del lado derecho tenemos los límites inferior y superior respectivamente, para cada uno.

Hay dos maneras de modificar los parámetros dando un CLICK sobre los cuadros que vemos seleccionados en la figura anterior. En este caso observaremos que aparece un nuevo objeto, una barra de desplazamiento, con la que podremos modificar el valor de nuestro parámetro.



Barra de desplazamiento activada al dar CLICK sobre el cuadro que contiene el valor del *número de iteraciones*. Para desactivarla se dará un CLICK fuera de la misma.

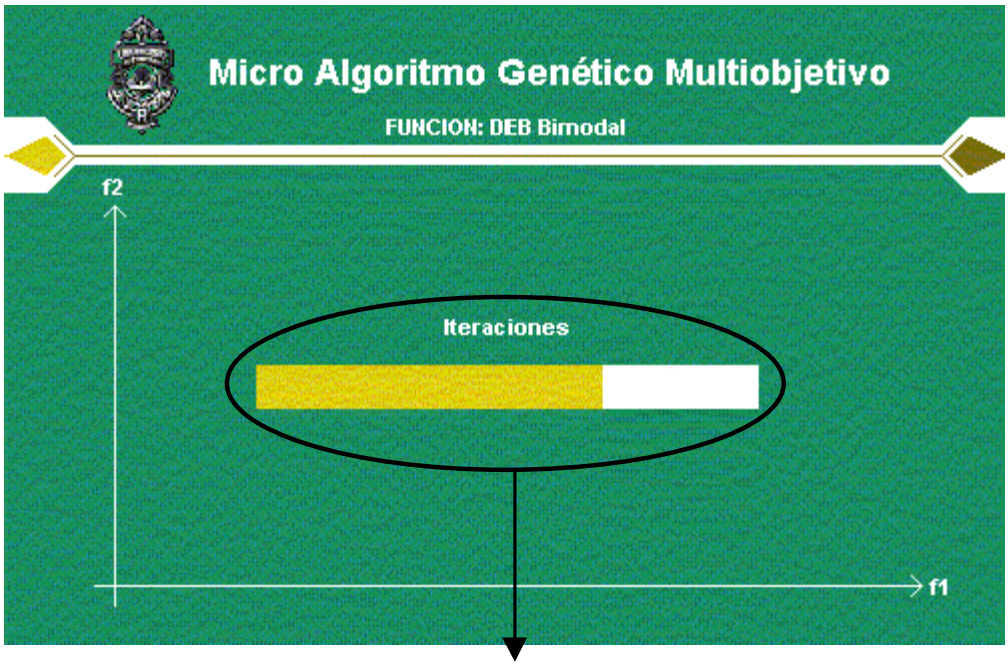
Otra manera de hacer cambios a los parámetros será haciendo SHIFT + CLICK, sobre el cuadro del parámetro que deseamos modificar. En este caso se activará un cuadro de diálogo, en el cual capturaremos de manera directa nuestro nuevo valor. En caso de capturar un valor fuera del rango del límite inferior y el superior, se omitirá el valor capturado y se restaurará el valor default.



Cuadro de diálogo activado al dar CLICK sobre el cuadro que contiene el valor del *número de iteraciones*. Para desactivarlo se dará un CLICK fuera del mismo.

Una vez capturados los parámetros deseados, podremos ver la última parte de nuestra aplicación. En este paso, sólo nos resta ver el comportamiento que presentan los puntos pertenecientes al conjunto de Pareto.

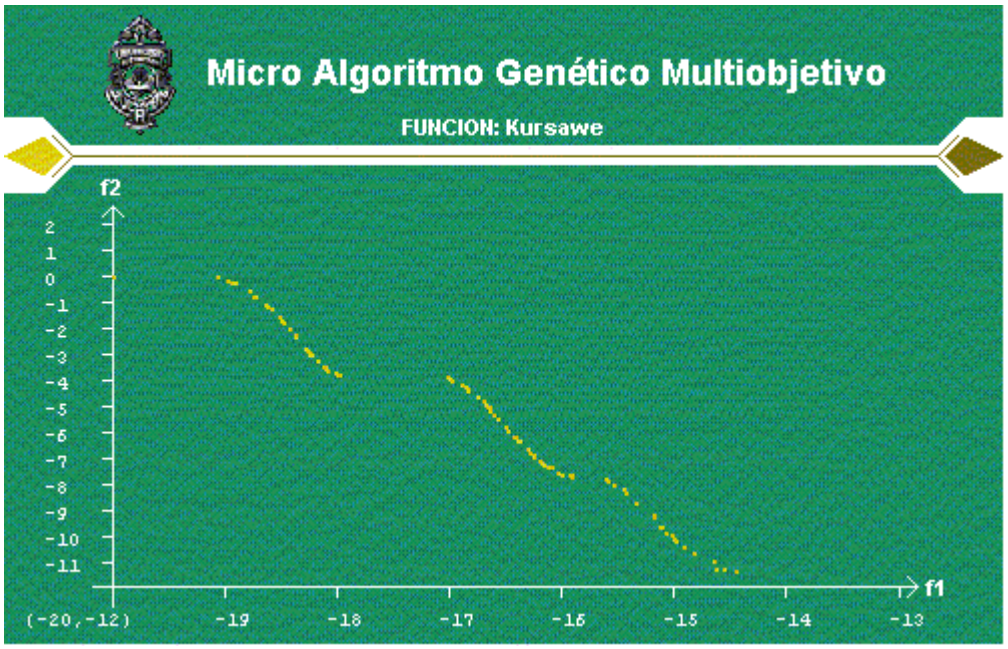
Mientras está corriendo el algoritmo, nos muestra gráficamente el porciento de avance de acuerdo al total de iteraciones dado, por medio de una barra de progreso la cual se completa hasta el 100%, indicándonos que el algoritmo ha terminado de ejecutarse de manera satisfactoria y está preparándose para graficar los resultados almacenados en el conjunto de Pareto.



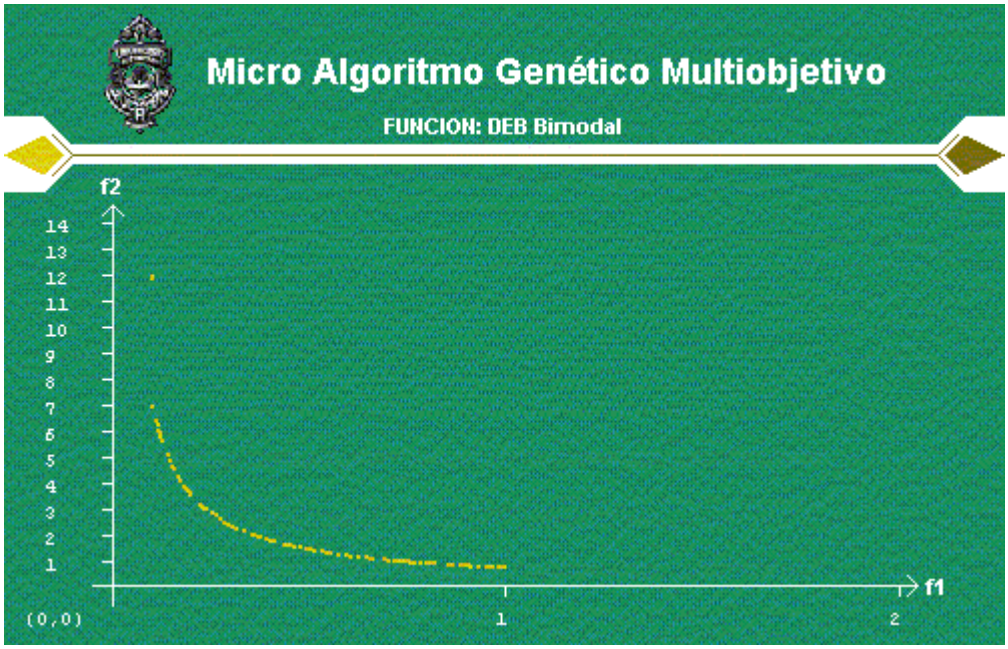
Barra de progreso del algoritmo, complementándose cuando el algoritmo llega al número de iteraciones dado.

A continuación, se presentan las gráficas resultantes de la aplicación, para cada una de las funciones prueba.

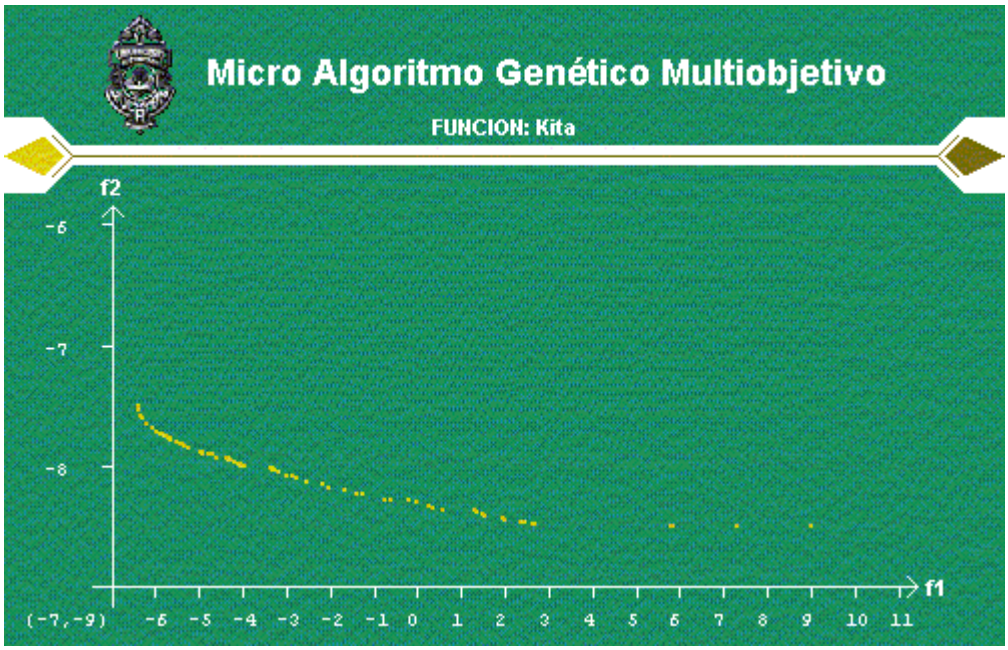
Kursawe



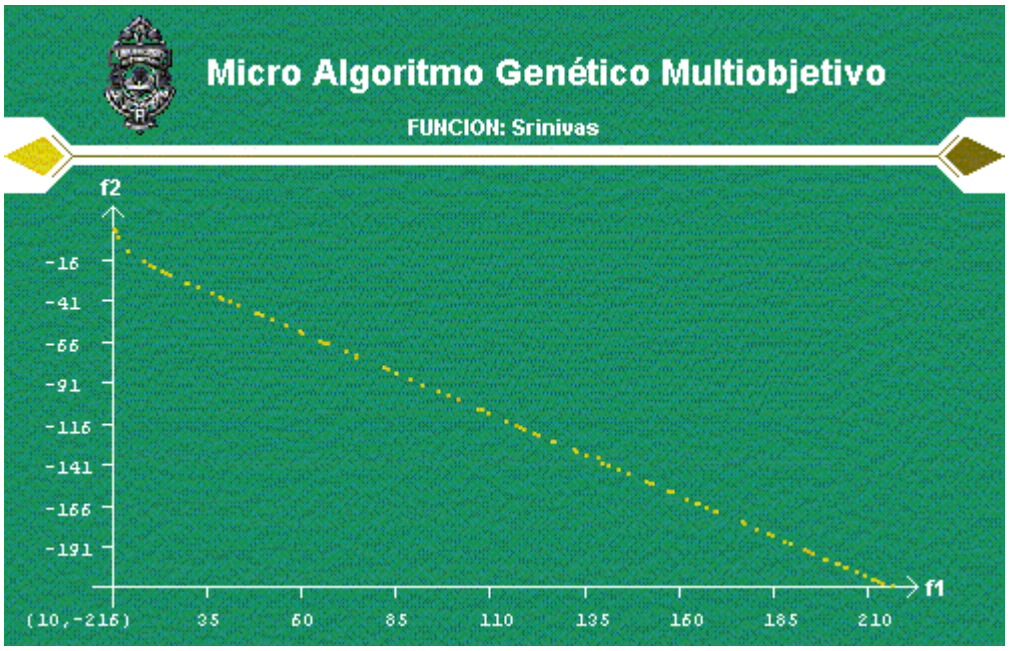
DEB Bimodal



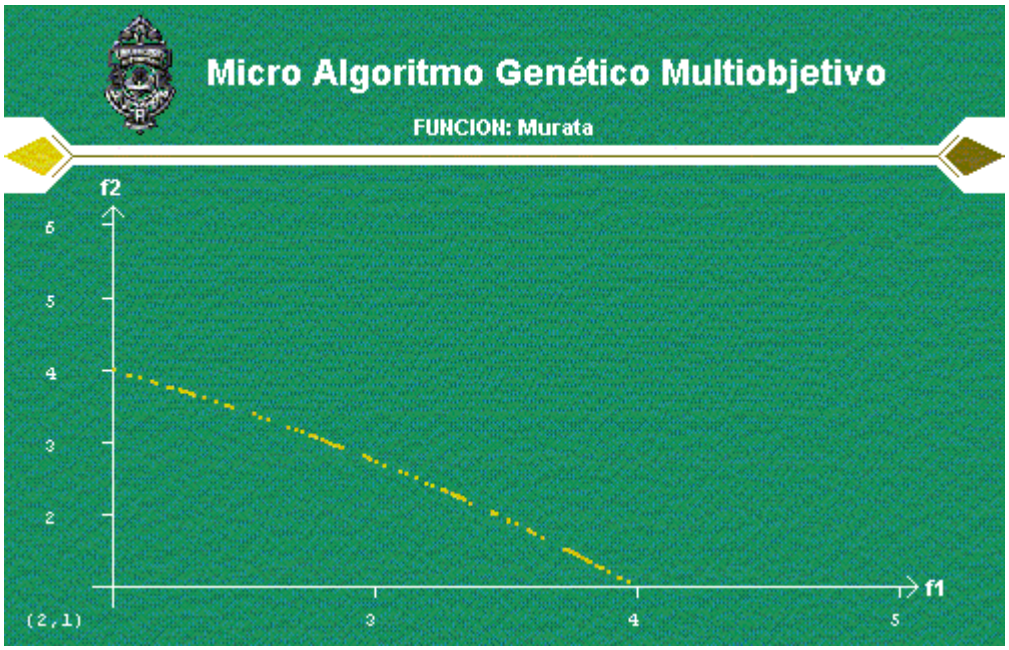
Kita



Srinivas



Murata



Si al terminar la ejecución de una función de prueba se desea comparar con otra, deberá dar CLICK en la flecha de la izquierda para seleccionar sus nuevos parámetros y luego ejecutar de nueva cuenta el algoritmo.