Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

**Unidad Zacatenco**

**Departamento de Computación**

# Un algoritmo memético para optimización de espacios restringidos

Tesis que presenta

**Miriam Pescador Rojas**

para obtener el Grado de

**Maestra en Ciencias**

en la Especialidad de

**Computación**

Director de la Tesis

**Dr. Carlos Artemio Coello Coello**

**México, D.F.** **Diciembre, 2010**

# Agradecimientos

En primer lugar quiero expresar mi gratitud al Centro de Investigación y de Estudios Avanzados CINVESTAV y a todas las personas que forman parte de esta casa de estudios, quienes a través de sus funciones brindan la formación de estudios de excelencia en investigación y posgrado. Por haber sido un segundo hogar donde día a día viví agradables experiencias académicas.

Agradezco el apoyo del Consejo Nacional de Ciencia y Tecnología CONACYT y del Consejo Mexiquense de Ciencia y Tecnología COMECYT. Manifiesto mi más profundo reconocimiento por la labor que desempeñan al aportar ayuda financiera y motivar el desarrollo de investigación en nuestro país.

Mi especial reconocimiento y admiración a mi asesor de tesis el Dr. Carlos A. Coello Coello, quien con sus innumerables e invaluables aportaciones, observaciones y consejos hizo posible el desarrollo de esta tesis.

A mis revisores de tesis el Dr. Gregorio Toscano Pulido y el Dr. Luis Gerardo de la Fraga por el tiempo que dedicaron para examinar esta tesis y porque gracias a sus comentarios he obtenido un mejor trabajo.

A cada uno de los doctores del departamento por sus enseñanzas en los cursos que imparten, gracias a ellos he logrado una muy completa formación en ciencias en computación.

Al grupo EVOCINV ya que en cada uno de los seminarios de investigación me ilustraron con sus presentaciones y me aportaron buenos consejos para desarrollar mi trabajo. Al Dr. Gregorio Toscano Pulido y Dr. Guillermo Leguizamon por ampliar mis conocimientos en el área de la computación evolutiva.

A mis compañeros y amigos de la maestría, Fermín, Jorge, Rosa, Gustavo, Arturo y René, por su compañía en esta etapa de mi vida, por compartir gratos momentos y hacer que la estancia en el departamento de computación fuera más amena.

*Esta tesis esta dedicada especialmente a mi mamá y mis hermanos, a quienes agradezco de todo corazón su amor, cariño, comprensión y apoyo incondicional.*

*Agradezco por fomentar en mí el deseo por la sabiduría y el conocimiento, gracias por abrir las puertas del mundo ante mi curiosidad e inquietudes.*

*Agradezco su apoyo, su guía y confianza en la realización de mis sueños. Soy afortunada por contar siempre con su amor, comprensión y ejemplo. El esfuerzo realizado se convirtió en nuestro triunfo, los amo y en todo momento los llevo conmigo.*

*Sinceramente muchas gracias.*

*Miriam Pescador Rojas*

# Resumen

La optimización numérica tiene una amplia aplicación en diversas disciplinas que modelan problemas del mundo real, como por ejemplo, las mejoras al rendimiento de un sistema, la planeación de procesos, la asignación de recursos financieros, el diseño en ingeniería y el desarrollo de modelos. Estos problemas presentan criterios a maximizar o minimizar y/o restricciones que deben ser cumplidas.

En la solución de problemas de optimización se toman decisiones mediante la elección sistemática de valores de las variables de decisión, las cuales se definen en un espacio de búsqueda específico. Las restricciones, por su parte, se refieren a que una solución es aceptable sólo si se cumplen ciertas condiciones pre-establecidas por el usuario o por los requerimientos de diseño.

Se han desarrollado numerosas técnicas de programación matemática para resolver distintos tipos de problemas de optimización. Sin embargo, su uso resulta inadecuado en diversos casos: por ejemplo, cuando las funciones no son diferenciables o presentan discontinuidades, cuando la función objetivo es altamente no lineal, cuando el espacio de búsqueda está altamente restringido o cuando la dimensionalidad del problema es muy elevada.

Una alternativa para resolver este tipo de problemas son los denominados algoritmos meméticos (AMs), cuyo funcionamiento se basa en la combinación estratégica de mecanismos de búsqueda global con mecanismos de búsqueda local. Mientras un mecanismo global explora el espacio completo de búsqueda, el local explota determinadas regiones de éste, a fin de refinar las soluciones obtenidas.

La principal contribución de esta tesis es el diseño y la implementación de un algoritmo memético para la solución de problemas de optimización en espacios restringidos. Este algoritmo se compone de lo siguiente: un mecanismo de búsqueda global basado en evolución diferencial (ED), un método para el manejo de restricciones denominado jerarquías estocásticas (JE) y un procedimiento de búsqueda local que usa el operador de cruza simplex (SPX), el cual explota 2 vecindarios definidos por las $n$ mejores y las $n$ peores soluciones de cada generación.

El algoritmo memético propuesto es validado usando un conjunto de problemas de prueba estándar tomado de la literatura especializada. Sus resultados se comparan con respecto a los de cuatro algoritmos representativos del estado del arte en el área. Así mismo, se realiza un análisis estadístico de resultados, empleando el análisis de varianza (ANOVA) y un método de remuestreo (*bootstrap*).

Los resultados obtenidos indicaron que el algoritmo memético propuesto fue el que encontró los mejores resultados al conjunto de pruebas adoptado, presentando además un comportamiento robusto.

# Abstract

Numerical optimization has a wide applicability in a variety of disciplines that model real-world problems, such as performance improvements of a system, process planning, financial resources assignment, engineering design and model development. These problems have criteria to be maximized or minimized and/or constraints that must be satisfied.

When solving optimization problems, decisions are made through the systematic selection of values of the decision variables, which define a specific search space. Constraints, on the other hand, refer to considering a solution as acceptable, only if it fulfills certain conditions that are pre-established by the user of by the design requirements.

Numerous mathematical programming techniques have been developed to solve different types of optimization problems. However, their use is inappropriate in several cases: for example, when the objective functions are non-differentiable or present discontinuities, when the objective function is highly nonlinear, when the search space is highly constrained or when the dimensionality of the problem is too high.

An alternative to solve this type of problems is to use the so-called memetic algorithms (MAs), which are based on a strategic combination of global search mechanisms with local search mechanisms. While a global mechanism explores all of the search space, the local mechanism exploits certain regions within it, aiming to refine the solutions previously obtained.

The main contribution of this thesis is the design and implementation of a memetic algorithm for solving constrained optimization problems. This algorithm is composed by the following elements: a global search mechanism based on differential evolution (DE), a constraint-handling mechanism called stochastic ranking (SR) and a local search procedure that adopts simplex crossover (SPX), which exploits 2 neighborhoods defined by the $n$ best and the $n$ worst solutions of each generation.

The proposed memetic algorithm is validated using a set of standard test problems taken from the specialized literature. Its results are compared with respect to those produced by four algorithms representative of the state-of-the-art in the area. Furthermore, a statistical analysis of results is also performed, using the analysis of variance (ANOVA) and a resampling method (*bootstrap*).

The results obtained indicated that the proposed memetic algorithm found the best overall results in the test problems adopted, having a robust behavior as well.

# Índice general

# Lista de tablas

# Lista de figuras

# Lista de Algoritmos

Capítulo 1

# Introducción

La optimización tiene una enorme aplicabilidad en diversas ramas del conocimiento. Con su uso se busca obtener las mejores soluciones posibles a modelos matemáticos que representan problemas del mundo real. En la formulación de dichos modelos, pueden presentarse funciones del tipo: *mono-objetivo*, con un solo objetivo a cumplirse; *multi-objetivo*, con dos o más objetivos a cumplirse; con *restricciones*, en los cuales las soluciones deben estar dentro de ciertos límites; así como funciones *multimodales*, que presentan muchos mínimos locales y *multivariable*, las cuales involucran muchas variables de decisión.

En la solución de problemas de optimización se toman decisiones para maximizar o minimizar determinados criterios, mediante la elección sistemática de valores de las variables de decisión, las cuales se definen en un espacio de búsqueda específico. Las restricciones, por su parte, significan que la decisión es posible sólo si se cumplen ciertas condiciones.

Las técnicas clásicas de programación matemática buscan máximos o mínimos de una función, empleando métodos numéricos que usan derivadas o su aproximación. Sin embargo, estos métodos resultan inadecuados en diversos casos, por ejemplo, cuando las funciones no tienen derivadas, cuando el espacio de búsqueda es muy restringido, cuando la función es discontinua, etc. Estas situaciones vuelven más complejo un problema de optimización.

Para solucionar problemas de optimización que se definen en espacios de búsqueda complejos, se han usado los métodos de programación llamados heurísticas o metaheurísticas, los cuales buscan buenas soluciones(es decir, casi óptimas) a un costo computacional razonable. La desventaja de su uso es que no pueden garantizar factibilidad u optimalidad de las soluciones que producen [96].

Existen técnicas de optimización que combinan sinérgicamente conceptos de las metaheurísticas con mecanismos de mejora local. A este tipo de técnicas se les denomina algoritmos meméticos (AMs) [62]. Este tipo de técnica adopta una metaheurística como su mecanismo de búsqueda global, el cual tiene como objetivo encontrar regiones prometedoras del espacio de búsqueda, mientras que el buscador local realiza un refinamiento de las soluciones encontradas. Ambos esquemas trabajan en forma cooperativa, mediante un balance entre la exploración y la explotación de las regiones del espacio de búsqueda.

En esta tesis abordamos la resolución de problemas mono-objetivo con restricciones empleando AMs. Para ello enfocamos nuestro estudio en el diseño, implementación y acoplamiento de mecanismos de búsqueda global y local, así como en el manejo de restricciones del problema. Para generar nuestra propuesta examinamos algoritmos competitivos

del estado del arte, con la finalidad de extraer características que logran un buen desempeño en la solución de diversas funciones tomadas de la literatura especializada.

## 1.1. Antecedentes y motivación

En los años 1600 surgen los primeros métodos de optimización. En esta época se desarrolló el cálculo diferencial gracias a las contribuciones de Isaac Newton y Gottfried Wilhelm von Leibniz. Posteriormente fueron establecidos los fundamentos de cálculo de variaciones, por Johann Bernoulli, Leonhard Euler, Joseph-Louis Lagrange y Karl Weierstrass. Para problemas con restricciones, se involucraron funciones y métodos de multiplicadores de Lagrange.

A pesar de estas aportaciones, no fue sino hasta mediados del siglo XX, con la aparición de las computadoras digitales, que fue posible la implementación de los algoritmos de optimización existentes. Las computadores digitales fueron el parteaguas en el desarrollo de nuevos métodos y la creación de diversas áreas bien definidas en teoría de la optimización.

En 1947 surgió el método simplex de George Dantzig y el principio de optimalidad para problemas de programación dinámica anunciado por Richard Bellman en 1957. Esto sentó las bases para el desarrollo de métodos de optimización en espacios restringidos. Sin embargo, el trabajo de Harold W. Kuhn y Albert W. Tucker en 1951, en torno a las condiciones necesarias y suficientes para la solución a un problema de optimización, estableció los fundamentos para investigación en optimización no lineal. El trabajo de C.W. Carroll y el de Anthony V. Fiacco y Garth P. McCormick, tiene gran importancia porque permitió resolver una amplia gama de problemas con restricciones usando funciones de penalización.

En la segunda mitad del siglo XX el advenimiento de las heurísticas marca el inicio de la *computación evolutiva*. Estos algoritmos imitan comportamientos encontrados en la naturaleza como: el principio de la "supervivencia del más apto", el comportamiento social y cooperativo emergente de organismos agrupados en colonias, la denominada inteligencia cumular, la propagación e influencia de la cultura en una población, entre otros. Su funcionamiento radica en generar soluciones iniciales que van siendo modificadas con la finalidad de mejorar el valor de la función objetivo en cada iteración. En la implementación de estos métodos se introducen parámetros probabilísticos, que necesitan ser calibrados por el usuario con el objetivo de incrementar el desempeño del algoritmo en un problema dado.

Las tendencias actuales sugieren la recombinación de diversos mecanismos para explorar y explotar el espacio de búsqueda del problema. Una vez que se ha ubicado una zona factible se recomienda utilizar un mecanismo de búsqueda local que agilice la convergencia al óptimo global, sin perder la diversidad de soluciones a fin de evitar caer en óptimos locales. Diversidad y convergencia son conceptos que generalmente se contraponen, por lo tanto debe existir un balance entre ambos.

Un buscador local tiene el objetivo de explotar determinadas regiones. Por lo tanto,

está asociado al uso de estructuras de entorno que reflejan el concepto de proximidad o vecindad. Su función es generar soluciones vecinas que mejoren el entorno de la(s) solucion(es) actual(es).

## 1.2. Planteamiento del problema

Resolver un problema de optimización con restricciones difiere considerablemente a resolver uno sin restricciones, ya que comúnmente se modifica de manera importante el espacio de búsqueda del problema, lo que puede producir un cambio en la ubicación de la solución óptima global. Por lo tanto, se debe guiar la búsqueda involucrando las funciones de restricción en el algoritmo. Existen diversos métodos para el manejo de las restricciones [86], y su desempeño está en función de la forma en la que estos se combinan con el método de búsqueda utilizado.

En la actualidad existen diversos trabajos relacionados con el uso de buscadores locales para espacios discretos [18], donde es más clara su utilización, debido a que el espacio de búsqueda es reducido, haciendo más fácil la exploración del vecindario total de soluciones. En contraste, los buscadores locales para problemas definidos en espacios continuos, deben realizar movimientos pequeños con una alta precisión.

A pesar de que los buscadores locales aceleran la convergencia al óptimo global, el inconveniente es que suelen caer en óptimos locales, que a veces están muy alejados del óptimo global del problema. Por eso es indispensable tener una estrategia que mantenga la diversidad de las soluciones, para evitar quedar atrapado en óptimos locales.

La introducción y el diseño de buscadores locales para espacios de búsqueda continuos no es simple y depende en gran medida del tipo de problema a resolver. Para su diseño deben definirse: la forma de generar un vecindario, la frecuencia de uso, las soluciones sobre las que se trabajará, el tiempo de ejecución, la estrategia de reemplazo y el mantenimiento de la diversidad. De no ser así, su uso se vuelve innecesario y no beneficiará al algoritmo de búsqueda global.

## 1.3. Objetivos

El objetivo general de esta tesis es diseñar e implementar un AM que resuelva problemas de optimización definidos en espacios restringidos y continuos. Por lo tanto, es necesario un estudio de diversos mecanismos de búsqueda global y local, para identificar ventajas y desventajas de su uso. Para cumplir el objetivo principal de esta tesis se definieron los siguientes objetivos particulares:

- Investigar mecanismos competitivos de búsqueda global y local para espacios continuos.
- Analizar características de diseño de diversos buscadores locales.

- Implementar un algoritmo de búsqueda global para espacios continuos.
- Implementar una estrategia para el manejo de las restricciones del problema.
- Implementar un algoritmo de búsqueda local para espacios continuos.
- Definir aspectos que identifiquen que un buscador local tienen buen desempeño en espacios continuos.
- Acoplar el buscador local y global, así como el manejo de restricciones para producir un AM para problemas de optimización restringidos.
- Realizar pruebas con funciones estándar tomadas de la literatura especializada.
- Medir el desempeño del algoritmo.
- Comparar con AMs del estado del arte.
- Corroborar los beneficios del uso de buscadores locales.
- Realizar el estudio y análisis estadístico de los resultados obtenidos.

## 1.4. Contribuciones

Las principales contribuciones de esta tesis son las siguientes:

- Un estudio de las características de diversos buscadores locales.
- Un AM competitivo con respecto a medidas de desempeño reportadas en la literatura especializada.
- Un diseño experimental con funciones de prueba que validan la propuesta.
- Un análisis estadístico para el ajuste de parámetros de la propuesta.

## 1.5. Organización de la tesis

El resto de la tesis está organizada de la manera siguiente:

- **Capítulo 2. Optimización:** Introduce la definición general de un problema de optimización y describe cada uno de los elementos que lo definen. Presenta la solución de problemas de optimización mediante métodos de programación matemática y algoritmos de computación evolutiva.
- **Capítulo 3. Motores de búsqueda global:** Define un mecanismo de búsqueda global específicamente para espacios continuos. Menciona AEs del estado del arte, como los algoritmos genéticos con codificación real, la optimización mediante cúmulo de partículas y la evolución diferencial. Menciona diversos métodos para el manejo de restricciones en los AEs.

- **Capítulo 4. Buscadores locales para espacios continuos:** Describe diferentes métodos y técnicas como algoritmos de búsqueda local, clasificándolos en: métodos numéricos tradicionales de optimización con uso de derivadas, métodos de búsqueda directa, métodos heurísticos y operadores genéticos basados en vecindarios. Se presentan las diferencias entre cada uno y la forma en la que trabajan.

- **Capítulo 5. Algoritmos meméticos en la literatura:** Aborda AMs recientes de la literatura especializada, enfocándose en aquellos que hacen uso de una heurística para codificación real y buscadores locales con métodos de búsqueda directa u operadores genéticos basados en vecindarios, tanto para problemas con y sin restricciones. En este punto se presenta de manera detallada el algoritmo híbrido para problemas mono-objetivo sin restricciones en el que se basa nuestra propuesta.

- **Capítulo 6. Propuesta de algoritmo memético para problemas con restricciones:** Describe las modificaciones realizadas al algoritmo DEahcSPX tomado de la literatura a fin de adaptarlo a problemas de optimización con restricciones. Se explica el diseño del algoritmo híbrido, resolviendo las interrogantes en el diseño de buscadores locales que se mencionan en el planteamiento del problema de la tesis.

- **Capítulo 7. Validación del método propuesto:** Expone la manera en la que se realizó el diseño experimental y la selección de los parámetros de control del algoritmo. Se presenta una comparativa entre nuestra propuesta y AMs del estado del arte. Además, se muestra el efecto de la utilización del buscador local.

- **Capítulo 8. Análisis estadístico:** Propone el ajuste de parámetros mediante un análisis estadístico y presenta los nuevos resultados del algoritmo con este ajuste.

- **Capítulo 9. Conclusiones y trabajo a futuro:** Presenta un resumen detallado del trabajo de tesis. Desarrolla una discusión de las contribuciones de la nueva propuesta en comparación con algoritmos competitivos de la literatura. Destaca los principales resultados del trabajo y finalmente describe como trabajo a futuro algunas posibles mejoras al algoritmo y comparaciones o análisis estadísticos que pudiera extender este trabajo.

- **Apéndice A. Funciones de prueba:** Define el conjunto de 22 funciones de prueba tomado de la literatura especializada, así como sus características y principales dificultades en su resolución.

- **Apéndice B. Gráficas de convergencia:** Para cada una de los experimentos realizados en la tesis se presentan las gráficas de convergencia, gráficas de caja e histogramas para visualizar el comportamiento del algoritmo propuesto.

Capítulo 2

# Optimización

Las técnicas de optimización están presentes en cualquier problema que involucre la toma de decisiones, sobre todo en ingeniería o en economía. La tarea del tomador de decisiones implica seleccionar, entre varias alternativas, la mejor solución que cumpla con sus requerimientos. La medida del valor de estas alternativas está descrita por una función objetivo o un índice de desempeño.

En diseño, construcción y mantenimiento de sistemas, los ingenieros deben tomar decisiones tanto tecnológicas como administrativas en diferentes etapas, con el objetivo final de minimizar el esfuerzo requerido o maximizar el beneficio obtenido. En cualquier situación práctica este esfuerzo o beneficio puede ser expresado como una función con ciertas variables de decisión. Por lo tanto, la optimización se define como el proceso de encontrar las condiciones que darán el valor máximo o mínimo de una función.

Los métodos de búsqueda del óptimo son también conocidos como técnicas de programación matemática, las cuales son ampliamente estudiadas en el área de investigación de operaciones. Esta área es una rama de las matemáticas relacionada con la aplicación de métodos científicos y técnicas para la toma de decisiones que establecen cuáles son las soluciones deseables para un problema de optimización.

## 2.1. Definición de un problema de optimización

El problema de optimización general que nos interesa para los fines de esta tesis se define como:

Encontrar $\vec{x} = [x_1, x_2, ..., x_n]^T$ que minimice [1] $f(\vec{x})$ sujeto a:

$$\left\{ \begin{array}{l} g_i(\vec{x}) \leqslant 0, \quad i = 1, 2, ..., m \\ \\ h_j(\vec{x}) = 0, \quad j = 1, 2, ..., p \end{array} \right\} \tag{2.1}$$

donde $\vec{x}$ es un vector $n-$dimensional llamado *vector de diseño*, $f(\vec{x})$ es la *función objetivo*, $g_i(\vec{x})$ y $h_j(\vec{x})$ son las *restricciones de desigualdad e igualdad*, respectivamente. El problema formulado en la ecuación (2.1) es conocido como el *problema de optimización restringido*.

[1] Sin pérdida de generalidad, nos referimos sólo a problemas de minimización, ya que un problema de maximización puede transformarse a uno de minimización. Ver figura 2.1.

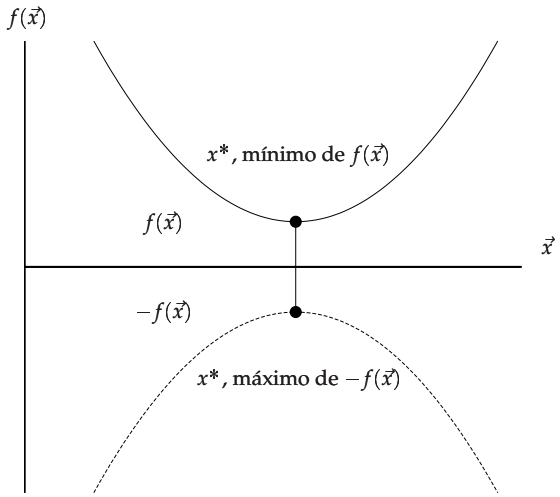



Figura 2.1: Mínimo y máximo de una función objetivo

**Vector de diseño**

Cualquier sistema o componente de ingeniería está definido por un conjunto de cantidades, algunas de las cuales son vistas como variables durante el proceso de diseño. En general, ciertas cantidades son usualmente fijas y se conocen como parámetros preasignados. Las otras cantidades son tratadas en el diseño del proceso como variables de decisión o de diseño $x_i,\ i = 1, 2, ..., n$. Las variables de diseño son colectivamente representadas como un vector de diseño $\vec{x} = [x_1, x_2, ..., x_n]^T$.

Cada punto $n-$dimensional en el espacio de diseño es llamado *punto de diseño* y representa una solución posible al problema [72].

**Restricciones de diseño**

En la mayor parte de los problemas de ingeniería, las variables de diseño no pueden elegirse de manera arbitraria, sino que deben satisfacer ciertos requerimientos. Las condiciones que deben satisfacerse para producir un diseño aceptable son llamadas, genéricamente, *restricciones de diseño*.

Las restricciones que representan limitaciones en el comportamiento o el desempeño del sistema son denominadas *restricciones funcionales o de comportamiento*. Las restricciones que representan limitaciones físicas tales como disponibilidad, facilidad de fabricación y transportabilidad son denominadas *restricciones geométricas*.

**Función objetivo**

Los procedimientos de diseño convencional tienen como objetivo encontrar modelos aceptables o adecuados, que satisfagan la funcionalidad y requerimientos de un problema. Por lo tanto, la formulación de un problema de optimización debe identificar las variables

de diseño que servirán para la construcción de un modelo matemático, destinado a dar una buena representación del problema. Un ejemplo de carácter económico pudiera ser maximizar un beneficio y/o minimizar un costo. Comúnmente, se presenta más de un diseño aceptable, por lo que se debe elegir el mejor de un conjunto de alternativas. Dicho modelo es una función escalar de las variables de diseño, $f : \mathbb{R}^n \rightarrow \mathbb{R}$, conocida como *función objetivo*. Cuando se presentan múltiples funciones objetivo (por lo regular en conflicto), se tiene un *problema de optimización multiobjetivo* [78, 86].

**Óptimo global**

Un óptimo global (suponiendo minimización) es aquél que representa el valor más pequeño de $f(\vec{x})$, en todo el espacio de búsqueda del problema, ver figuras 2.2 y 2.3.

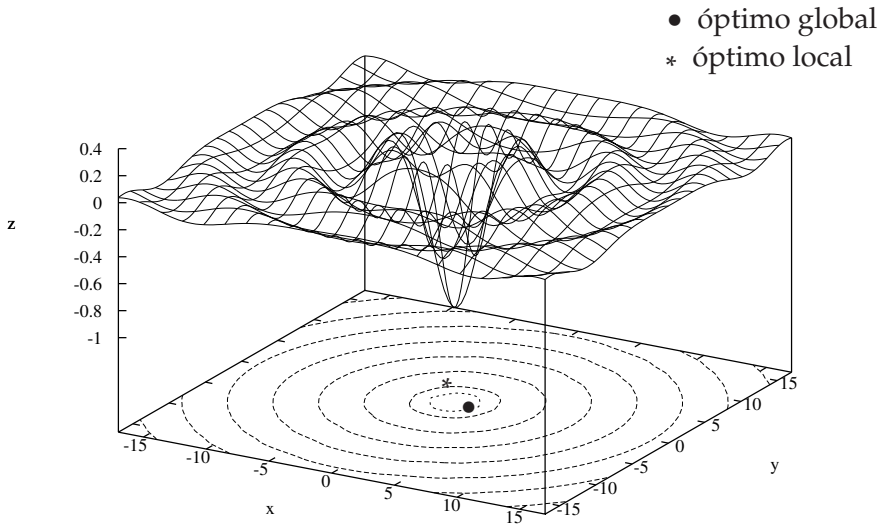



Figura 2.2: Ejemplo de un óptimo global y un óptimo local de un problema

**Definición:** Dada una función $f : \Omega \triangle \mathbb{R}^n \rightarrow \mathbb{R}$, $\Omega \neq \varnothing$, para $\vec{x^*} \in \Omega$ el valor $f^* = f(\vec{x}) > -\infty$ es llamado *mínimo global*, si y sólo si:

$$\forall \vec{x} \in \Omega : f(\vec{x^*}) \leqslant f(\vec{x}) \tag{2.2}$$

De esta manera, el vector $\vec{x^*}$ es un punto mínimo global, $f$ es la función objetivo y el conjunto $\Omega$ es el subespacio de soluciones factibles. El problema de determinar un punto mínimo global es llamado *problema de optimización global*. En este caso, se tendrá un único *óptimo global*.

**Óptimo local**

Presuponiendo un problema de minimización, el óptimo local representa el valor más pequeño de $f(\vec{x})$ en la vecindad de algún vector $\vec{x}$, ver figura 2.2.

**Definición:** Para $\hat{\vec{x}} \in \Omega$ el valor $\hat{f} = f(\hat{\vec{x}})$ es llamado mínimo local, si y sólo si:

$$\exists \varepsilon \in \mathbb{R}, \varepsilon > 0 : \forall \vec{x} \in \Omega : |\vec{x} - \hat{\vec{x}}| < \epsilon \Rightarrow \hat{f} \leqslant f(\vec{x}) \tag{2.3}$$

En otras palabras, un ambiente $-\varepsilon \cup_\varepsilon (\hat{\vec{x}}) = \{\vec{x} \in \Omega : |\vec{x} - \hat{\vec{x}}| < \varepsilon\}$ existe, tal que $\hat{f}$ es el valor factible más pequeño de la función objetivo, dentro de ese ambiente. Las funciones con más de un máximo o mínimo son llamadas *funciones multimodales* (ver figura 2.3).

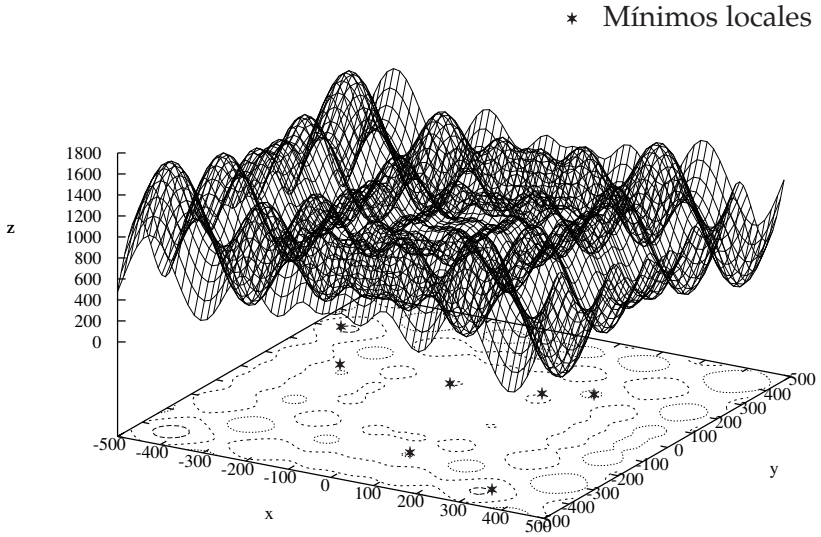



Figura 2.3: Ejemplo de multimodalidad (función de Schwefel)

En análisis numérico, un método iterativo es llamado *convergente localmente*, si las aproximaciones sucesivas producidas por el método garantizan la convergencia a un punto muy cercano a la solución global [6]. Sin embargo, existen funciones con soluciones óptimas locales, dentro de un vecindario en el espacio de búsqueda, que se encuentran muy alejadas del óptimo global, como se observa en la figura 2.3.

## 2.2. Espacios restringidos

Presuponiendo un problema de optimización con restricciones de desigualdad únicamente, $g_i(\vec{x}) \leqslant 0$, el subconjunto de valores de $\vec{x}$ que satisfacen la ecuación $g_i(\vec{x})$, forman una hipersuperficie en el espacio de diseño que es llamada *superficie restringida*. Este es un

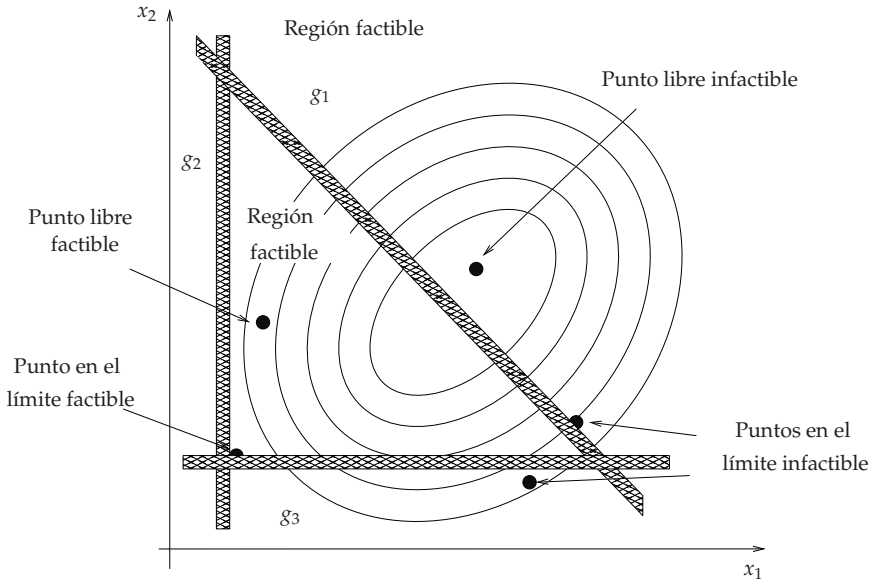



Figura 2.4: Caso hipotético de un espacio restringido

subespacio $(n-1)-$dimensional, donde $n$ es el número de variables de diseño. La superficie restringida divide el espacio de diseño en dos regiones: uno donde $g_i(\vec{x}) \leqslant 0$ y otro en el cual $g_i(\vec{x}) > 0$. Los puntos que tienden a estar en la hipersuperficie que satisface las restricciones $g_i(\vec{x})$, es decir, donde $g_i(\vec{x}) \leqslant 0$, son puntos factibles o aceptables, mientras que los que están en la región donde $g_i(\vec{x}) > 0$ son infactibles o inaceptables. La colección de todas las superficies restringidas $g_i(\vec{x}), i = 1, 2, ..., m$, es llamada *superficie de restricción compuesta*.

La figura 2.4 muestra un espacio de diseño bi-dimensional hipotético. A un punto de diseño que está sobre al menos una superficie restringida, se le conoce como *punto límite*, y la restricción asociada a él es una *restricción activa*. Los puntos de diseño que no estén en ninguna de las regiones son llamados *puntos libres*. A cualquier punto que satisface todas las restricciones del problema se le conoce como *punto factible*. El subconjunto de puntos factibles queda definido como:

$$\{\vec{x} \in \mathbb{R}^n : h_i(\vec{x}) = 0 \ (i = 1, ..., m), g_j(\vec{x}) \leqslant 0 \ (j = 1, ..., p)) \tag{2.4}$$

y se le denomina *conjunto factible* [18]:

**Características de los problemas restringidos**.

En la presencia de restricciones, un problema de optimización puede presentar las siguientes características:

- Las restricciones pueden no tener efecto en el punto óptimo. Esto es, el valor óptimo del problema restringido es el mismo que el del problema sin restricciones, tal como se muestra en la figura 2.5. En tal caso se podrían emplear métodos de optimización sin restricciones para su resolución. Sin embargo, en la mayoría de los problemas es ex-

tremadamente difícil identificar tal situación. De tal forma, se establece la suposición de que las restricciones tienen influencia en el óptimo.

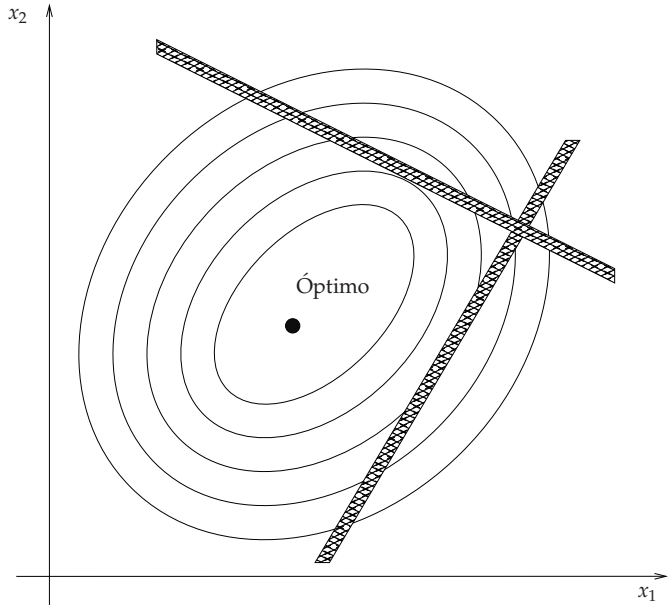



Figura 2.5: Las restricciones no tienen efecto sobre el punto óptimo

- La solución única puede tener lugar en el borde de la zona factible. En este caso las restricciones influyen en la localización del punto óptimo. Este efecto se observa en la figura 2.6, donde el óptimo global de la función (también llamado óptimo irrestricto) no se localiza dentro de la zona factible.

- Las restricciones pueden incrementar el número de óptimos locales, aún cuando la función objetivo posea un único óptimo global. Esto se observa en la figura 2.7.

## 2.3. Métodos de programación matemática

Los métodos de programación matemática pueden ser clasificados en *técnicas de programación matemática, técnicas estocásticas y métodos estadísticos*. Las técnicas de programación matemática son útiles para encontrar el mínimo de una función de varias variables sujeta a un conjunto de restricciones. Las técnicas estocásticas se usan para analizar problemas descritos por un conjunto de variables aleatorias con una distribución de probabilidad conocida. Finalmente, los métodos estadísticos nos permiten analizar datos experimentales y construir modelos empíricos para obtener la representación más precisa posible del problema real.

Los métodos clásicos de optimización son algoritmos determinísticos que se caracterizan por tener reglas específicas para moverse entre una solución y otra. Estos métodos han

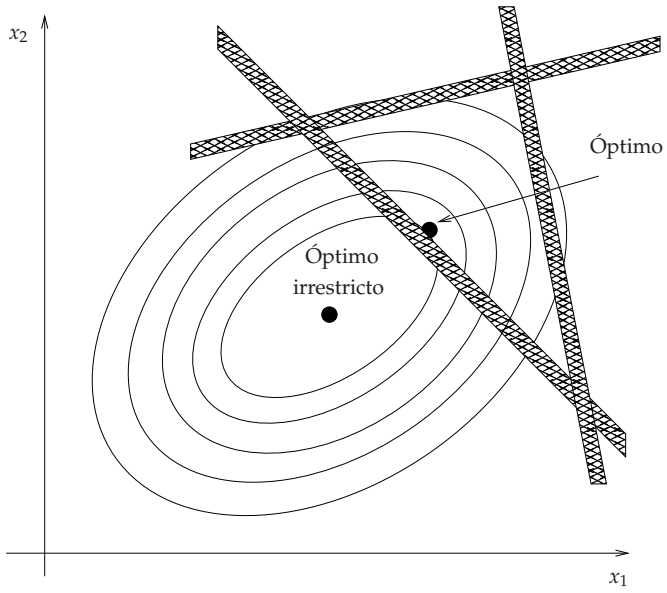



Figura 2.6: El óptimo del problema restringido difiere del óptimo del problema irrestricto encontrándose en el límite de la restricción.

sido utilizados desde hace tiempo y han sido aplicados exitosamente en muchos problemas de diseño, sobre todo en el área de ingeniería [16, 72]. Estos métodos han sido clasificados en dos grandes grupos: los *métodos basados en derivadas y los métodos de búsqueda directa*. A continuación se describe cada uno de ellos (ver figura 2.8):

- *Métodos basados en derivadas.* Estos métodos utilizan información de la derivada de la función para moverse entre una solución y otra. Ejemplos de este tipo de procedimientos son los métodos de bisección, secante y Newton-Raphson que se utilizan para resolver problemas unidimensionales. Mientras que el gradiente conjugado de Fletcher-Reeves, el gradiente descendente de Cauchy, los métodos Quasi-Newton (p.ej. el método de Davidon-Fletcher-Powell (DFP) y el método de Broyden-Fletcher-Goldfarb-Shanno (BFGS)) fueron diseñados para resolver problemas multidimensionales.

- *Métodos de búsqueda directa.* Este tipo de métodos no necesitan ninguna información de la derivada de la función. De igual manera, existen métodos de búsqueda directa para solucionar problemas unidimensionales (como los métodos de la sección dorada y el de Fibonacci, entre otros) y métodos para resolver problemas multidimensionales (como los algoritmos de Hooke-Jeeves, Nelder-Mead y las direcciones conjugadas de Powell, entre otros).

Desafortunadamente, los métodos de búsqueda directa no garantizan convergencia al óptimo global. En la mayoría de los casos, estos métodos dependen de un punto inicial de búsqueda; en funciones multimodales es común que queden atrapados en óptimos locales.

Existen también métodos de programación matemática para resolver problemas con restricciones. Por ejemplo, para problemas que tienen sólo restricciones de igualdad se

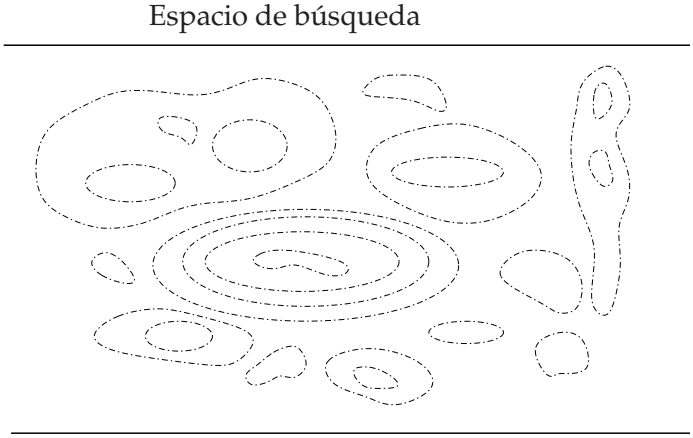



Figura 2.7: Incremento en el número de óptimos locales.

| Algoritmos de optimización | | | |
|---|---|---|---|
| Métodos clásicos | | Métodos estocásticos | |
| Basados en derivadas | De búsqueda directa | Basados en poblaciones | |
| Bisección | Fibonacci | Cómputo evolutivo | Recocido simulado |
| Newton-Raphson | Sección dorada | Algoritmos bio-inspirados | Búsqueda tabú |
| Descenso empinado | Hooke-Jeeves | Evolución diferencial | |
| Gradiente conjugado | Nelder-Mead | | |
| Método Quasi-Newton | Direcciones conjugadas | | |

Figura 2.8: Una posible taxonomía de los algoritmos de optimización

puede usar el método de los *multiplicadores de Lagrange*. Si el problema tiene restricciones de desigualdad, pueden usarse métodos de penalización. Sin embargo, es importante hacer énfasis en que algunos de estos métodos pueden emplearse sólo si la función objetivo y las restricciones son diferenciables.

## 2.4. Computación evolutiva

El término *computación evolutiva* o *algoritmos evolutivos* engloba una serie de técnicas inspiradas biológicamente por los principios de la teoría Neo-Darwiniana de la evolución natural. Ideas como la "selección natural", que fue introducida por Darwin en su libro de 1856, titulado el *El origen de las especies*, plantean que una especie que no sufriera cambios se volvería incompatible con su ambiente, ya que éste tiende a cambiar con el tiempo. Asimismo, las similitudes entre hijos y padres observadas en la naturaleza le sugirieron a Darwin que ciertas características de las especies eran hereditarias, y que, de generación a generación, ocurrían cambios cuya principal motivación era hacer a los nuevos individuos más aptos para sobrevivir.

En 1866, Gregor Mendel propuso tres leyes básicas que gobernaban el paso de una característica de un miembro de una especie a otro [54]. Alrededor de 1900, el botáni-

co Hugo De Vries reprodujo las leyes de la herencia de Mendel, e impulsó la teoría de las "mutaciones espontáneas", expandiendo con ella la teoría Darwiniana de la evolución [95]. Según De Vries, los cambios en las especies no eran graduales y adaptativos como afirmaba Darwin, sino más bien abruptos y aleatorios (es decir, al azar).

La teoría evolutiva propuesta originalmente por Charles Darwin en combinación con el seleccionismo de August Weismann y la genética de Gregor Mendel, se conoce hoy en día como el paradigma *Neo-Darwiniano*. Este paradigma establece que la historia de la vida en nuestro planeta puede ser explicada a través de un puñado de procesos estadísticos que actúan sobre y dentro de las poblaciones y especies [35]: la reproducción, la mutación, la competencia y la selección. Estos 4 conceptos son la base de los algoritmos evolutivos.

La evolución natural fue vista como un proceso de aprendizaje desde los 1930 por W. D. Cannon, quien plantea que el proceso evolutivo es algo similar al aprendizaje por ensayo y error que suele manifestarse en los humanos [7]. El célebre matemático inglés Alan Mathison Turing reconoció también una conexión entre la evolución y el aprendizaje de máquina [93].

En términos generales, para simular el proceso evolutivo en una computadora se requiere:

- Codificar las estructuras de datos que representan las soluciones del problema a resolverse.
- Contar con mecanismos que permitan generar nuevas soluciones, emulando procesos como la reproducción y la mutación.
- Establecer una función de aptitud, que permita comparar a los individuos entre sí.
- Plantear mecanismos de selección y reemplazo de los individuos.

La figura 2.9 muestra el comportamiento de un algoritmo evolutivo (específicamente un algoritmo genético). Se presenta una gráfica con las curvas de nivel que representan el valor de la función objetivo o función de aptitud. En la primera generación o iteración del proceso, se tiene una distribución de soluciones (padres) en todo el espacio del problema. El siguiente cuadro indica la generación (iteración) 50, donde se puede observar que las soluciones (hijos) candidatas están agrupadas cerca de los valores mínimos de la función. Por último, en la generación 100 se puede apreciar cómo las soluciones son cada vez más parecidas y se aglomeran cerca y sobre el óptimo global de la función.

## 2.5. Paradigmas de la computación evolutiva

Aunque hoy en día es cada vez más difícil distinguir las diferencias entre los distintos tipos de algoritmos evolutivos existentes, por razones sobre todo históricas, suele hablarse de tres paradigmas principales:

- Programación evolutiva
- Estrategias evolutivas

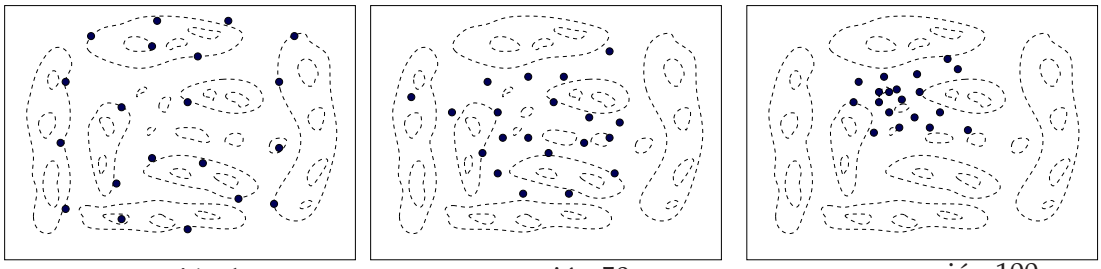



Figura 2.9: Ilustración gráfica del proceso evolutivo de un algoritmo genético.

- Algoritmos genéticos

Cada uno de estos paradigmas se originó de manera independiente y con motivaciones muy distintas.

### 2.5.1. Programación evolutiva

Lawrence J. Fogel et al. [25] concibieron el uso de la evolución simulada en la solución de problemas (sobre todo de predicción). Su técnica, denominada "Programación Evolutiva" [24] consistía básicamente en hacer evolucionar autómatas de estados finitos, los cuales eran expuestos a una serie de símbolos de entrada (el ambiente), y se esperaba que, eventualmente, serían capaces de predecir las secuencias futuras de símbolos que recibirían. Fogel utilizó una función de "pago" que indicaba qué tan bueno era un cierto autómata para predecir un símbolo, y usó un operador modelado en la mutación para efectuar cambios en las transiciones y en los estados de los autómatas que tenderían a hacerlos más aptos para predecir secuencias de símbolos. Esta técnica no consideraba el uso de un operador de recombinación, dado que se pretendía modelar el proceso evolutivo al nivel de las especies y no al de los individuos.

### 2.5.2. Estrategias evolutivas

En 1963, Peter Bienert, Ingo Rechenberg y Hans-Paul Schwefel estudiaban un problema de optimización de perfiles hidrodinámicos, el cual era imposible de resolver analíticamente o usando métodos tradicionales como el del gradiente. Para resolverlo, Ingo Rechenberg decidió desarrollar un método de ajustes discretos aleatorios inspirado en el mecanismo de mutación que ocurre en la naturaleza. Los resultados iniciales de esta técnica, a la que denominó "estrategia evolutiva", fueron presentados al Instituto de Hidrodinámica de su universidad el 12 de junio de 1964 [22]. Posteriormente, Rechenberg realizó cambios al método y lo aplicó a otros problemas de ingeniería. Hans-Paul Schwefel, por su parte, se dio a la tarea de implementar la técnica en una computadora y a extenderla, a fin de hacerla más general.

Las estrategias evolutivas $(1 + 1)$ son el modelo más sencillo dentro de este paradigma. En ellas la población está compuesta por un solo individuo y sólo se usa el operador de mutación. La técnica consiste en mutar al individuo y sustituir al anterior si es más

apto que su progenitor; en caso contrario, permanecerá el individuo anterior. Un aspecto característico de las estrategias evolutivas es que las tasas de mutación de cada gen se codifican como parte del genoma[2], por lo que también se encuentran sujetas a la evolución.

### 2.5.3. Algoritmos genéticos

En 1960, John H. Holland se interesó en estudiar los procesos lógicos involucrados en la adaptación. Inspirado por los estudios realizados en aquella época con autómatas celulares y redes neuronales, Holland se percató de que el uso de reglas simples podría generar comportamientos flexibles, y visualizó la posibilidad de estudiar la evolución de comportamientos en un sistema complejo. Holland advirtió que un estudio de la adaptación debía reconocer que [36, 37]:

a) la adaptación ocurre en un ambiente,

b) la adaptación es un proceso poblacional,

c) los comportamientos individuales pueden representarse mediante programas,

d) pueden generarse nuevos comportamientos mediante variaciones aleatorias de los programas, y

e) las salidas de dos programas normalmente están relacionadas si sus estructuras están relacionadas.

De tal forma, Holland vio el proceso de adaptación en términos de un formalismo en el que los programas de una población interactúan y mejoran con base en un cierto ambiente que determina lo apropiado de su comportamiento. El combinar variaciones aleatorias con un proceso de selección (en función de qué tan apropiado fuese el comportamiento de un programa dado), debía entonces conducir a un sistema adaptativo general.

Este sistema fue desarrollado hacia mediados de los 1960s, y se dio a conocer en el libro que Holland publicase en 1975, donde lo denominó "plan reproductivo genético" [37], aunque después se popularizó bajo el nombre (más corto y conveniente) de "algoritmo genético". Aunque concebido originalmente en el contexto del aprendizaje de máquina, el algoritmo genético se ha utilizado mucho en optimización.

[2]En biología, se denomina genoma a la colección completa de genes (y por tanto cromosomas) que posee un organismo. En computación evolutiva, un genoma es el conjunto de variables del problema, codificadas en cada individuo. Dado que los algoritmos evolutivos suelen usar cromosomas haploides, el término genoma y cromosoma se suelen usar de manera intercambiable, siendo el segundo más común que el primero.

Capítulo 3

# Motores de búsqueda global

## 3.1. Modelos de AEs para problemas en espacios continuos

La optimización de funciones definidas sobre dominios continuos ha sido un campo muy activo de investigación, que ha dado lugar a la creación de diversos algoritmos tanto determinísticos como estocásticos. Estos algoritmos deben relacionar el espacio de búsqueda del problema con un esquema de representación y codificación de números reales, proporcionado por una precisión computacional adecuada.

El esquema de representación de los algoritmos evolutivos (AEs) (principalmente en algoritmos genéticos) toma conceptos de la biología tales como: "cromosoma", "gene", "genotipo", "fenotipo", "individuo" y "alelo", entre otros. La figura 3.1 muestra gráficamente la definición de estos términos.

Se denomina **cromosoma** a una estructura de datos que contiene una cadena de parámetros de diseño o genes. Esta estructura de datos puede almacenarse como una cadena de bits, un arreglo de enteros o un vector de números reales. Por otro lado, un **gene** es una subsección de un cromosoma que (usualmente) codifica el valor de un solo parámetro.

**Genotipo** es la codificación (p. ej. binaria o real) de los parámetros que representan una solución del problema a resolver. En tanto, el **fenotipo** es la decodificación del cromosoma, es decir, los valores obtenidos al pasar del esquema de representación al valor usado por la función objetivo. En el caso de la codificación real (CR) no se necesita una decodificación para evaluar la solución.

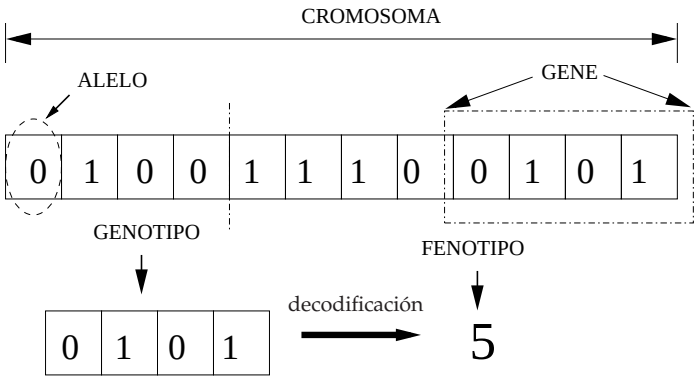



Figura 3.1: Ejemplo de un esquema de codificación binaria de una solución. A la cadena binaria completa se le considera como "individuo" que tiene un "cromosoma".

Se nombra **individuo** a un solo miembro del conjunto (**población**) de soluciones potenciales a un problema. Cada individuo contiene un cromosoma que representa una solución posible del problema a resolver.

Los problemas del mundo real usualmente son definidos en espacios continuos con una alta dimensionalidad, por lo que se requiere de una cierta precisión mínima para las variables; de lo contrario, una mala representación puede dejar fuera elementos cercanos al óptimo global o, incluso, al mismo óptimo. Varias técnicas de computación evolutiva evitan este problema usando números reales en la cadena cromosómica, como es el caso de las estrategias evolutivas [81] y la programación evolutiva [23], entre otros. Dichos algoritmos se describen brevemente a continuación.

### 3.1.1. Algoritmos genéticos con codificación real

Los algoritmos genéticos (AGs) son técnicas de búsqueda que proporcionan buenas soluciones a problemas de optimización de diversas áreas. Comúnmente los tipos de problemas que abordan están definidos en espacios de búsqueda muy complejos, por ejemplo funciones discontinuas o altamente restringidas. Los AGs se basan en procesos existentes en los organismos biológicos y en principios de la evolución natural de las especies. Estos algoritmos procesan una población de cromosomas, la cual representa el espacio de búsqueda de las soluciones y usan 3 operadores principales: selección, cruza y mutación.

En un inicio, los AGs eran representados con cadenas binarias [97], en donde un gen era una serie de bits y un cromosoma una cadena de la forma $< b_1, b_2, ..., b_m >$, donde cada bit se denomina "alelo". Sin embargo, otros tipos de codificación fueron considerados para la representación de estos algoritmos, tales como la CR [50], la cual puede parecer particularmente natural cuando se aborda problemas con variables continuas. En este esquema, un cromosoma es un vector de números de punto flotante de tamaño fijo e igual a la longitud del vector solución. De esta manera, cada gen representa una variable del problema, ver figura 3.2. En CR los valores de los genes se ven obligados a permanecer en el intervalo establecido por las variables que representan, por lo que los operadores genéticos deben trabajar con esta restricción.

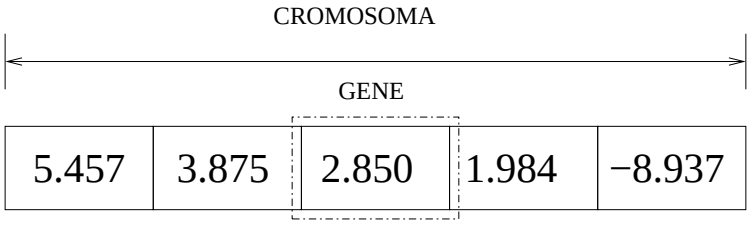



Figura 3.2: Ejemplo de un esquema de CR de una solución

CR hizo su aparición en el año de 1989 en aplicaciones específicas de los AGs tales como: la extracción de información en sistemas químicos [51], el uso de metaoperadores para encontrar los parámetros más adecuados de un AG estándar [12] y, posteriormente,

en problemas de optimización numérica en espacios continuos [13, 21, 33, 56, 97].

Un AG inicia su proceso con una población de cromosomas generada aleatoriamente y avanza hacia un mejor cromosoma, mediante la aplicación de los operadores genéticos. Durante las iteraciones sucesivas, llamadas **generaciones**, los cromosomas base producen nuevas poblaciones que se forma mediante un mecanismo de selección y operadores genéticos específicos como cruza y mutación. Dado un cromosoma en particular, la función devuelve un valor numérico único que es proporcional a la utilidad o la adaptación de la solución.

Aunque hay muchas variantes de los AGs, en general, todos operan sobre una población de cromosomas o individuos, que representan las posibles soluciones al problema y ejecutan los 3 pasos siguientes:

1. Evaluación de la aptitud de los individuos.
2. Formación de descendientes genéticos (población intermedia) a través de un mecanismo de selección.
3. Recombinación a través de operadores de cruza y mutación.

El algoritmo 1 muestra la estructura base de un AG. $P(t)$ denota la población en la generación $t$.

**Algoritmo 1**: Estructura de un AG

```
1  Datos: P población, t generaciones
2  Resultado: conjunto de soluciones óptimas
3  begin
4      t ← 0;
5      Genera población inicial P(t);
6      Evalúa P(t) para calcular aptitud de cada individuo;
8      while no se cumpla el criterio de terminación do
9          t ← t + 1;
10         Seleccionar P(t) individuos de P(t-1) con base en la aptitud;
11         Aplicar operadores genéticos (cruza y mutación);
12         Calcular aptitud de cada individuo, evaluando P(t);
13 end
```

**Mecanismo de selección**. Consideremos una población $P$ con cromosomas $C_1, C_2, ..., C_n$. El mecanismo de selección produce una población intermedia, $P^{'}$, con copias de los cromosomas en $P$. El número de copias recibidas para cada cromosoma depende de su aptitud. Cromosomas con mayor aptitud tienen, por lo general, una mayor probabilidad de ser seleccionados.

**Operador de cruza**. Es un método que comparte información entre cromosomas, combinando características de dos o más individuos que actúan como padres para formar

descendientes, con el propósito de que los cromosomas seleccionados puedan generar mejores soluciones. El operador de cruza no se aplica a todos los pares de cromosomas de la población intermedia, sino solamente a aquellos que fueron elegidos por el proceso de selección y su uso depende de la probabilidad definida por el usuario.

Suponiendo que tenemos dos cromosomas $C_1 = (c_{11}, ..., c_{1n})$ y $C_2 = (c_{21}, ..., c_{2n})$, que han sido seleccionados para aplicar el operador de cruza, el nuevo descendiente será resultado de combinar cierta información definida por métodos específicos. Para ejemplificar dicho funcionamiento se describen a continuación algunos operadores usados comúnmente en los AGs con codificación real.

*Cruza simple* [56,97]. Una posición $i \in \{1, 2, ..., n-1\}$ es aleatoriamente seleccionada, y se construyen dos nuevos individuos como sigue:

$$H_1 = (c_{11}, c_{12}, ..., c_{1i}, c_{2i+1}, ...., c_{2n}),$$

$$H_2 = (c_{21}, c_{22}, ..., c_{2i}, c_{1i+1}, ...., c_{1n})$$

*Cruza aritmética* [56]. Dos cromosomas descendientes $H_k = (h_1^k, ..., h_i^k, ..., h_n^k)$ $k = \{1, 2\}$, son generados, donde $h_i^1 = \lambda c_i^1 + (1-\lambda)c_i^2$ y $h_i^2 = \lambda c_i^2 + (1-\lambda)c_i^1$. $\lambda$ es una constante (cruza aritmética uniforme) o variable que cambia en cada iteración (cruza aritmética no uniforme).

*Cruza BLX* $-\alpha$ [21] Un descendiente es generado como: $H = (h_1, ..., h_i, ..., h_n)$, donde $h_i$ es un número aleatorio elegido en el intervalo $[c_{min} - I \cdot \alpha, c_{max} + I \cdot \alpha]$, $c_{max} = max(c_i^1, c_i^2)$, $c_{min} = min(c_i^1, c_i^2)$, $I = c_{max} - c_{min}$.

**Operador de mutación**. El operador de mutación altera arbitrariamente uno o más componentes (genes) de un cromosoma, los cuales son seleccionados a fin de aumentar la variabilidad estructural de la población. El papel de la mutación en los AGs es el de restaurar la pérdida o material genético inexplorado en la población para evitar una convergencia prematura a soluciones óptimas locales, tratando de asegurar que la probabilidad de llegar a cualquier punto en el espacio de búsqueda nunca sea cero. Cada posición de cada cromosoma en la población sufre un cambio al azar de acuerdo a una probabilidad definida por un porcentaje definido por el usuario.

Supongamos que se tiene un cromosoma $C = (c_1, ..., c_i, ..., c_n)$ y un gen $c_i$ dentro del intervalo $[a_i, b_i]$. Entonces el gen $c_i^{'}$ se obtiene después de aplicar el operador de mutación como lo muestran los siguientes ejemplos:

*Mutación aleatoria* [56]. $c_i^{'}$ es un número aleatorio en el intervalo de la variable $[a_i, b_i]$.

*Mutación no uniforme* [56]. Si este operador se aplica en una generación $t$ y $g_{max}$ es el número máximo de iteraciones, entonces

$$c_i^{'} = \begin{bmatrix} c_i + \Delta(t, b_i - c_i) & \text{si } \tau = 0, \\ c_i - \Delta(t, c_i - a_i) & \text{si } \tau = 1 \end{bmatrix} \tag{3.1}$$

con $\tau$ igual a un número de cero a uno y

$$\Delta(t, y) = y \left( 1 - \tau^{\left(1 - \frac{t}{gmax}\right)^b} \right) \tag{3.2}$$

donde $\tau$ es un número aleatorio en el intervalo $[0, 1]$ y $b$ es un parámetro definido por el usuario, el cual determina el grado de dependencia entre el número de iteraciones. La ecuación (3.2) proporciona un valor en el rango $[0, y]$ tal que la probabilidad de devolver un número cercano a cero aumenta a medida que avanza el algoritmo. El tamaño del intervalo de generación de genes podrá ser inferior con el paso de las generaciones. Esta propiedad hace que este operador realice una búsqueda uniforme en el espacio inicial cuando $t$ es pequeño, y una búsqueda local en una fase posterior.

Cabe señalar que después de la cruza y la mutación, una estrategia de selección adicional, llamada estrategia elitista, debe ser adoptada [42], con la finalidad de que el cromosoma con mejor resultado sobreviva intacto de una generación a otra. Esto es necesario ya que es posible que el mejor cromosoma desaparezca, a causa de los operadores de cruza o mutación.

Utilizar un esquema de CR tiene como ventaja el uso de dominios de gran tamaño para las variables, lo cual es difícil de lograr en las implementaciones con sistema binario. Otra ventaja es la capacidad para explotar la "gradualidad" de las funciones con variables continuas. El concepto de gradualidad se refiere al hecho de que pequeños cambios en las variables corresponden a ligeros cambios en la función. Esta línea desarrolló a su vez la capacidad de ajuste local en las soluciones.

Operadores genéticos con CR como la mutación no uniforme de Michalewicz [56] permiten el refinamiento de soluciones, el cual se produce de una forma más rápida que en mutaciones con representación binaria donde el ajuste es más difícil debido al efecto de los "riscos de Hamming" [8]. El uso de una representación en números reales no tiene diferencias entre el genotipo (codificación) y el fenotipo (espacio de búsqueda), por lo tanto no se requiere de procesos de codificación y decodificación, lo cual permite un aumento en la velocidad de ejecución del AG.

### 3.1.2. Optimización mediante cúmulos de partículas

Existen sistemas de optimización inspirados en el comportamiento social de las colonias de insectos y otras sociedades colectivas. Este tipo de técnicas se denominan genéricamente inteligencia de cúmulos (IC). Los sistemas de IC a menudo se consideran dentro de la familia de los AEs, debido a que utilizan una población de individuos, llamados "agentes"que evolucionan a través del tiempo para encontrar mejores soluciones del problema.

La optimización mediante cúmulos de partículas (mejor conocida como *Particle Swarm Optimization* o PSO) y la optimización mediante colonia de hormigas (*Ant Colony Optimization* o ACO) son dos formas de IC. La característica principal de estos sistemas es que en lugar de usar operadores genéticos para adaptarse al ambiente, cada agente evoluciona

por sí mismo de acuerdo a su experiencia pasada y su interacción local con otros agentes. A la larga, la interacción local entre agentes causa un comportamiento global inteligente que desemboca en la localización de una solución aceptable.

PSO es un algoritmo de optimización estocástico propuesto originalmente por Kennedy y Eberhart en el año de 1995 [17]. Éste se basa en una población o "cúmulo" que, a diferencia de la mayoría de los AEs no implementa un mecanismo de selección de aptitud. Los individuos persisten a lo largo del tiempo adaptándose a los cambios del ambiente. Particularmente, las partículas modifican su posición en el espacio de búsqueda bajo la influencia de la experiencia de sus vecinos, así como la suya propia.

PSO simula el comportamiento social del vuelo de las aves o de un banco de peces, para encontrar rápidamente una solución a un problema de optimización, ya que aprovecha el comportamiento colectivo que las aves presentan al volar para encontrar comida. Siguiendo la analogía con el vuelo de las aves, en PSO, un cúmulo de partículas vuela a través de un espacio de búsqueda hiper-dimensional. Debido a esto, PSO se considera un algoritmo de comportamiento distribuido que desempeña una búsqueda multidimensional.

Cada individuo del cúmulo es llamado partícula y se representa mediante un vector $x_i = (x_{i1}, x_{i2}, ..., x_{ij})^T$ para $i = 1, ..., s$ y $j = 1, ..., n$, donde $s$ es el número de partículas y $n$ el número de dimensiones. La posición de cada partícula está influenciada por la mejor posición visitada por $s$ y la mejor posición visitada por sus vecinos. De acuerdo con esto, la posición de cada partícula, $x_i$, se ajusta en cada iteración (t) del algoritmo utilizando la ecuación (3.3)

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{3.3}$$

donde, $v_i$ es un vector de la forma $v_i = (v_{i1}, v_{i2}, ..., v_{ij})^T$ y representa la velocidad de movimiento, $i$ representa a la partícula actual y $j$ representa la dimensión del problema. Por lo tanto, $v_i$ está dada por la ecuación (3.4)

$$v_i(t+1) = \omega v_i(t) + c_1 r_1(t)(y_i(t) - x_i(t)) + c_2 r_2(t)(\hat{y}_i(t) - x_i(t)) \tag{3.4}$$

donde:

- $\omega$ es el factor de inercia [82].
- $c_1$ y $c_2$ son dos constantes, llamados coeficientes de aceleración.
- $r_1$ y $r_2$ son dos vectores aleatorios de la forma $r_1 = (r_{11}, r_{12}, ..., r_{1j})$ y $r_2 = (r_{21}, r_{22}, ..., r_{2j})$. Su valor está dado por una distribución uniforme con límite inferior 0 y límite superior 1.
- $y_i(t)$ es un vector de la forma $y_i(t) = (y_{i1}, y_{i2}, ..., y_{ij})(t)$ y representa la mejor posición de la partícula *(líder)* $x_i$ encontrada por sí misma en el paso $t$.
- $\hat{y}_i(t)$ es un vector de la forma $\hat{y}_i(t) = (\hat{y}_{i1}, \hat{y}_{i2}, ..., \hat{y}_{ij})(t)$, que representa la mejor posición obtenida por los vecinos de la partícula $x_i$ en el paso $t$.

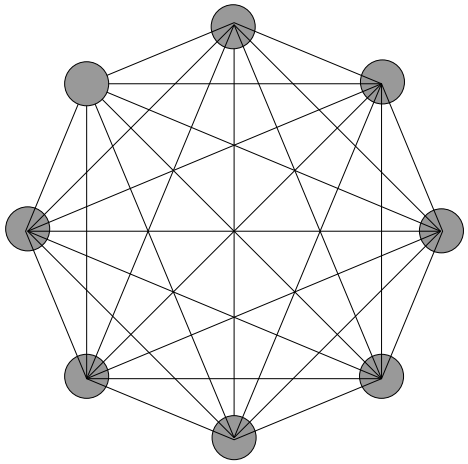


Figura 3.3: Topología de PSO version gbest

Es importante destacar que la velocidad de cada partícula $x_i$ se determina a partir de tres factores importantes:

- **La velocidad de inercia**, $v_i(t)$, actúa como un plazo de impulso para prevenir oscilaciones excesivas en la dirección de búsqueda.
- **El componente cognitivo**, $c_1 r_1(t)(y_i(t) - x_i(t))$, representa la mejor posición que una partícula ha encontrado por sí misma donde $c_1$ representa la fuerza de atracción de esta posición. Análogamente con el vuelo de las aves, el componente cognitivo representa la tendencia natural de los individuos a regresar al ambiente donde experimentaron su mejor desempeño.
- **El componente social**, $c_2 r_2(t)(\hat{y}_i(t) - x_i(t))$, representa la mejor posición que todo el vecindario ha logrado obtener. Esta posición se obtiene comparando la posición de sus vecinos con la propia donde $c_2$ indica la fuerza de atracción de la mejor posición alcanzada por el cúmulo. Análogamente con el vuelo de las aves, este componente representa la tendencia de los individuos a seguir al individuo más apto.

Para establecer el componente social $\hat{y}_i$ de cada partícula es necesario definir una topología de interconexión del vecindario. Un ejemplo es la topología completamente conectada, donde todas las partículas del cúmulo se comunican entre sí. A este algoritmo se le denomina **mejor global (gbest)**. Ver figura 3.3 y algoritmo 2.

**Algoritmo 2**: Algoritmo de PSO versión gbest.

**Datos:** $C$ cúmulo de partículas , $t$ iteraciones
**Resultado:** conjunto de soluciones óptimas
**begin**
  $t \leftarrow 0$ ;
  Genera cúmulo de partículas inicial ;
  **for** *cada partícula* $i$ **do**
    Inicializar $x_i$ y $v_i$ aleatoriamente ;
    Evaluar $x_i$ en la función de aptitud ;
    Inicializar $y_i \leftarrow x_i$ ;
  Seleccionar $\hat{y}_i$ ;
  **while** *no se cumpla el criterio de terminación* **do**
    **for** *cada partícula* $i$ **do**
      **for** *cada dimension* $j$ **do**
        Calcular la velocidad de $v_{ij}$ usando:
        $v_{ij}(t+1) \leftarrow \omega v_{ij}(t) + c_1 r_{1j}(t)(y_{ij}(t) - x_{ij}(t)) + c_2 r_{2j}(t)(\hat{y}_{ij}(t) - x_{ij}(t))$;
        Actualizar la posición de $x_{ij}$ usando:
        $x_{ij}(t+1) \leftarrow x_{ij}(t) + v_{ij}(t+1)$ ;
      Evaluar la nueva posición de $x_i$ ;
      **if** $x_i \leqslant y_i$ **then**
        $y_i \leftarrow x_i$;
      **if** $x_i \leqslant \hat{y}_i$ **then**
        $\hat{y}_i \leftarrow x_i$ ;
    $t \leftarrow t + 1$;
**end**

### 3.1.3. Evolución diferencial

Evolución Diferencial (ED) surgió en el año de 1995 como fruto de los intentos de Kenneth Price y Rainer Storn por resolver el problema del "ajuste polinomial de Chebychev" [87]. Con el fin de mejorar las soluciones existentes, se emplearon vectores diferencia como mecanismo de perturbación y de esta idea surgió la ED [88].

ED es un modelo para parámetros con codificación real, que hace énfasis en el operador de mutación, además de emplear un operador de cruza o recombinación *a posteriori* de la mutación. La idea principal de ED es utilizar la *diferencia entre vectores* para perturbar a un tercer vector de un conjunto de soluciones (población). En ED un individuo se representa como una *n-tupla* llamada **vector objetivo** $\vec{x} = (x_1, x_2, ..., x_n)$, donde $x_i \epsilon \mathbb{R} (i = 1, 2, ..., n)$.

Los pasos que conforman ED se describen a continuación:

**Inicialización**. Una población $\mathbf{P}_{x,0}$ de $N$ vectores $D-$dimensionales $\vec{x}_{i,0} = [x_{1,i,0}, \ldots, x_{D,i,0}]$, $i = 1, \ldots, N$ es generado aleatoriamente dentro de los límites inferior y superior $\vec{b}_L =$

$[b_{1,L}, \ldots, b_{D,L}]$ y $\vec{b}_U = [b_{1,U}, \ldots, b_{D,U}]$

**Mutación diferencial**. Con respecto a cada vector $\vec{x}_{i,g}$ en la población actual, un vector mutante $\vec{v}_{i,g}$ es generado agregando un **vector diferencia escalado** y seleccionado aleatoriamente, a un **vector base** seleccionado aleatoriamente de la población actual. Esto es, en la $g-$ésima generación, el $i-$ésimo vector mutante $\vec{v}_{i,g}$ con respecto al $i-$ésimos vector objetivo $\vec{x}_{i,g}$ en la $g-$ésima población actual es generado por la ecuación (3.5).

$$\vec{v}_{i,g} = \vec{x}_{r1,g} + F \cdot (\vec{x}_{r2,g} - \vec{x}_{r3,g}), \tag{3.5}$$

donde $F$ es un factor de escala de la mutación ($F \in (0, 1+)$) y $i \neq r1 \neq r2 \neq r3$. La figura 3.4 muestra el comportamiento del operador de mutación.

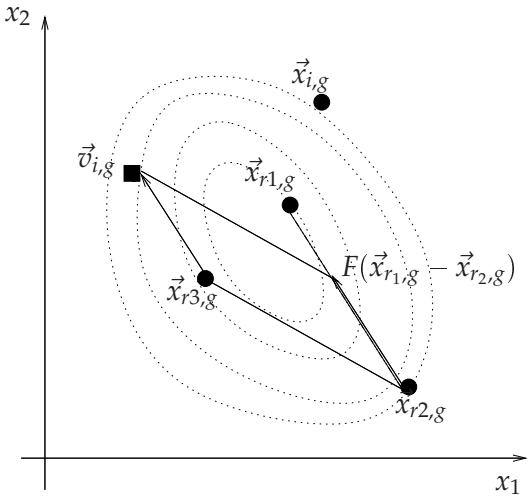



Figura 3.4: Esquema del operador de mutación en ED

**Recombinación discreta**. Con respecto a cada vector objetivo $\vec{x}_{i,g}$ en la población actual, un vector de prueba $\vec{u}_{i,g}$ es generado por la cruza del vector objetivo $\vec{x}_{i,g}$ con el correspondiente vector mutante $\vec{v}_{i,g}$ bajo un porcentaje de cruza previamente especificada por el parámetro probabilístico $Cr \in [0, 1]$. Esto es, en la $g-$ésima generación, el $i-$ésimo vector de prueba $\vec{u}_{i,g}$ con respecto al $i-$ésimo vector objetivo $\vec{x}_{i,g}$ en la población actual es generado por la ecuación (3.6).

$$u_{j,i,g} = \begin{cases} v_{j,i,g} & \text{si } \text{rand}_j[0,1] \leqslant Cr \text{ o } j = j_{\text{rand}} \\ x_{j,i,g} & \text{de lo contrario} \end{cases} \tag{3.6}$$

La figura 3.5 muestra el comportamiento del operador de cruza, después de haber aplicado mutación.

Inicialmente se propusieron dos procedimientos de recombinación denominados exponencial y binomial. Estos controlan la frecuencia con la cual los parámetros del vector de mutación $\vec{v}_i = \vec{x}_{r3} + F(\vec{x}_{r1} - \vec{x}_{r2})$ son seleccionados para formar parte de la solución descendiente $\vec{u}_i$. En la figura 3.6 se ilustran ambos procesos, donde $U_i(0, 1)$ representa una variable aleatoria uniformemente distribuida en el intervalo [0,1] y el vector $\vec{z}$ es el producto de la recombinación de los vectores $\vec{x}$ y $\vec{y}$. La cruza exponencial es similar a la cruza de un

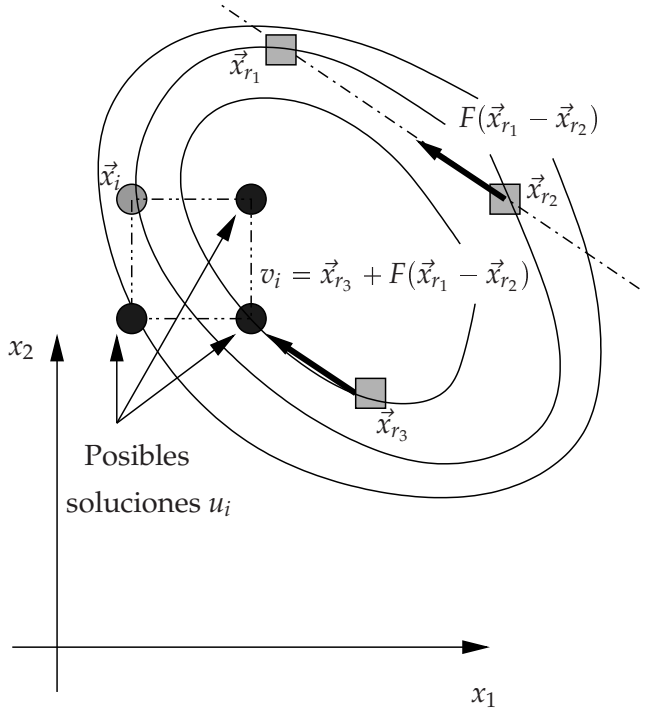



Figura 3.5: Esquema del operador de recombinación en ED

punto [27], mientras que la cruza binomial es similar a la cruza uniforme en los algoritmos genéticos [90].

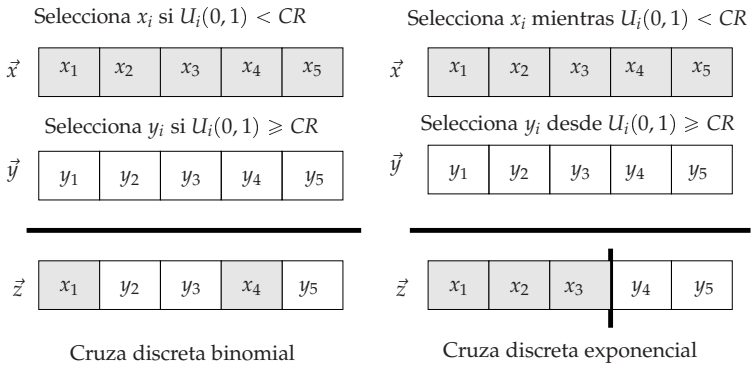



Figura 3.6: Recombinación discreta binomial y exponencial

**Selección y reemplazo**. Si el **vector de prueba** $\vec{u}_{i,g}$ tiene un mejor valor de la función objetivo que el de su correspondiente vector objetivo $\vec{x}_{i,g}$, éste remplaza al **vector objetivo** en la $(g+1)^{\text{ésima}}$ generación. De lo contrario, el vector objetivo permanece en la $(g+1)^{\text{ésima}}$ generación. Ver ecuación (3.7)

$$\vec{x}_{i,g+1} = \begin{cases} \vec{u}_{i,g} & \text{si } f(\vec{u}_{i,g}) \leqslant f(\vec{x}_{i,g}), \\ \vec{x}_{i,g} & \text{de lo contrario} \end{cases} \tag{3.7}$$

El pseudocódigo de la ED se muestra en el algoritmo 3.

**Algoritmo 3**: Algoritmo de la ED

**Datos:** $P_g$ población inicial de vectores , $G$ generaciones
**Resultado:** conjunto de soluciones óptimas
**begin**
  $G \leftarrow 0$;
  Inicializar $P_g \leftarrow \{\vec{x}_1, ..., \vec{x}_{np}\}$;
  **while** *no se cumpla el criterio de terminación* **do**
    **for** $i \leftarrow \{1, ..., np\}$ **do**
      $r_1, r_2, r_3 \in 1, ..., np$ aleatoriamente seleccionados,
      donde $r_1 \neq r_2 \neq r_3 \neq i$ ;
      $j_{rand} \in \{1, ..., n\}$ aleatoriamente seleccionado;
      **for** $j \leftarrow \{1, ..., n\}$ **do**
        **if** $U_j[0, 1] < CR \mid\mid j == j_{rand}$ **then**
          $u_{i,j} \leftarrow x_{r3,j,G} + F(x_{r1,j,G} - x_{r2,j,G})$ ;
        **else**
          $u_{i,j} \leftarrow x_{i,j,G}$;
        **if** $f(\vec{u}_i) \leqslant f(\vec{x}_{i,G})$ **then**
          $\vec{x}_{i,G+1} \leftarrow \vec{u}_i$ ;
    $G \leftarrow G + 1$;
**end**

**Modelos de ED**. Desde sus primeras versiones, la ED no ha presentado un único diseño. En la literatura se han propuesto diferentes estrategias para los operadores de mutación y recombinación. Por tal motivo, se estableció una nomenclatura que denomina las diferentes versiones de ED [70].

Existen diferentes variantes de ED. Para clasificarlas se utiliza la siguiente notación, **DE/x/y/z**, donde:

$x$ especifica la forma en que se eligen los vectores que van a participar en la mutación.

$y$ indica el número de diferencias entre vectores que se utilizan para la mutación.

$z$ denota el esquema de cruza que se va a utilizar.

Algunos ejemplos son los siguientes:

**DE/rand/1/bin**. Indica que se realiza una selección aleatoria de los vectores que conforman el vector mutación, así como una sola diferencia de vectores y un proceso de recombinación binomial.

**DE/best/1/bin**. Indica que el mejor individuo de la población participa en el proceso de mutación. Se realiza una sola diferencia de vectores y un proceso de recombinación binomial.

**DE/rand/2/exp**. Selecciona aleatoriamente 4 vectores que componen el vector mutación y son recombinados con un proceso exponencial.

**DE/best/2/exp**. Indica que el mejor individuo de la población participa en el proceso de mutación. 4 vectores componen el vector mutación y son recombinados con un proceso exponencial.

Además, la ecuación (3.5) puede variar de la siguiente forma:

- DE/rand/1: $v_{i,G} = \vec{x}_{r_3,G} + F(\vec{x}_{r_1,G} - \vec{x}_{r_2,G})$
- DE/best/1: $v_{i,G} = \vec{x}_{best,G} + F(\vec{x}_{r_1,G} - \vec{x}_{r_2,G})$
- DE/best_current/1: $v_{i,G} = \vec{x}_{i,G} + F(\vec{x}_{best,G} - \vec{x}_{i,G}) + F(\vec{x}_{r_1,G} - \vec{x}_{r_2,G})$
- DE/rand/2: $v_{i,G} = \vec{x}_{r_1,G} + F(\vec{x}_{r_2,G} - \vec{x}_{r_3,G} + \vec{x}_{r_4,G} - \vec{x}_{r_5,G})$
- DE/best/2: $v_{i,G} = \vec{x}_{best,G} + F(\vec{x}_{r_1,G} - \vec{x}_{r_2,G} + \vec{x}_{r_3,G} - \vec{x}_{r_4,G})$

## 3.2. Manejo de restricciones en los AEs

Los AEs no cuentan con un mecanismo explícito para manejo de restricciones, por lo que en los últimos años se ha realizado una amplia investigación para incorporar mecanismos a los AEs que les permitan resolver problemas con restricciones. A continuación se describen algunas de las técnicas más conocidas.

### 3.2.1. Funciones de penalización

Las funciones de penalización es uno de los mecanismos que se utiliza comúnmente para manejar restricciones en los AEs. En este caso la idea es transformar un problema con restricciones en uno sin restricciones al agregar a la función objetivo los factores de penalización tanto para restricciones de igualdad como para las de desigualdad. De esta forma, si las restricciones son violadas, entonces el valor de la función objetivo se ve afectado por un costo adicional que es proporcional a la magnitud de la violación de las restricciones.

En programación matemática, se consideran dos tipos de funciones de penalización [10]:

- Penalización exterior. Se permiten soluciones infactibles al inicio de la búsqueda y, posteriormente se guía la búsqueda hacia las zonas factibles.
- Penalización interior. Únicamente se permiten soluciones factibles. La penalización es elegida de tal forma que cuando los puntos están lejos de los límites de la zona factible ésta es pequeña y tiende a infinito cuando los puntos se acercan a dichos límites. Lo anterior, con la finalidad de preservar la búsqueda únicamente en el interior de la zona factible.

Debido a la dificultad de generar soluciones factibles en muchos problemas del mundo real, en los AEs se ha optado principalmente por el uso de esquemas de penalización exterior.

Hay al menos tres formas de penalizar a un individuo de acuerdo a su violación de las restricciones [75]:

- Puede penalizársele simplemente por no ser factible, sin usar ninguna información sobre qué tan cerca se encuentra de la región factible.
- Puede usarse la "cantidad" de infactibilidad de un individuo para determinar su penalización correspondiente.
- Puede usarse el esfuerzo de "reparar" al individuo (o sea, el costo de hacerlo factible) como parte de la penalización.

Richardson et al. [75] definieron algunas de las reglas básicas para diseñar una función de penalización:

1. Las penalizaciones que son funciones de la distancia a la zona factible son mejores que aquellas que son sólo funciones del número de restricciones violadas.
2. Para un problema con pocas variables y pocas restricciones, una penalización que sea sólo función del número de restricciones violadas no producirá ninguna solución factible.
3. Pueden construirse buenos factores de penalización a partir de 2 factores: el costo de cumplimiento máximo y el costo de cumplimiento esperado. El primero de ellos se refiere al costo de hacer factible a una solución infactible.
4. Las penalizaciones deben estar cerca del costo de cumplimiento esperado, pero no deben caer frecuentemente por debajo de él. Entre más preciso sea el factor de penalización, mejores resultarán las soluciones producidas.

Las funciones de penalización son de la siguiente forma:

$$\phi(\vec{x}) = f(\vec{x}) \pm \left[\sum_{i=1}^{m} r_i G_i + \sum_{i=1}^{p} c_j L_j\right] \tag{3.8}$$

donde $\phi(\vec{x})$ es la función objetivo extendida a optimizar, $G_i$ y $L_j$ son funciones de las restricciones $g_i(\vec{x})$ y $h_i(\vec{x})$ respectivamente, $r_i$ y $c_j$ son constantes positivas llamadas "factores de penalización".

La forma más común de $G_i$ y de $L_j$ esta dada por las ecuaciones (3.9) y (3.10):

$$G_i = max\{0, g_i(\vec{x})\}^{\beta} \tag{3.9}$$

$$L_j = |h_j(\vec{x})|^{\gamma} \tag{3.10}$$

donde, $\beta$ y $\gamma$ son uno o dos, normalmente. Además, una restricción de igualdad se puede transformar en una de desigualdad de la siguiente forma:

$$|h_j(\vec{x})| - \epsilon \leqslant 0 \tag{3.11}$$

donde $\epsilon$ establece la tolerancia permitida.

El uso de esta técnica tiene como principal ventaja el hecho de que el resultado depende de los factores de penalización y éstos son difíciles de determinar. Se debe cuidar que los factores de penalización esté equilibrados con respecto a los valores de la función objetivo y además se debe determinar qué tanto se quiere penalizar a una solución infactible. Se han propuesto diversas variantes de la función de penalización. A continuación se describen las más populares en la literatura especializada.

#### 3.2.1.1. Pena de muerte

Consiste en desechar a los individuos infactibles y volverlos a generar. Una variante de esta técnica consiste en asignar una aptitud de cero a un individuo infactible y tomar el valor de la función objetivo para los individuos factibles. Esta técnica no es muy recomendable ya que habrá un estancamiento en la búsqueda cuando en la población inicial no exista ningún individuo factible. Michalewicz [57] ha concluido que esta técnica sólo se puede usar en espacios de búsqueda convexos y en los casos en los que la zona factible constituya una parte razonablemente grande del espacio de búsqueda. Además, esta técnica sólo puede lidiar con restricciones de desigualdad.

Esta técnica es muy eficiente porque no tenemos que recalcular las restricciones o la función objetivo ni tenemos que reparar a los individuos no factibles. Sin embargo, tiene varios inconvenientes, como se indicó anteriormente.

#### 3.2.1.2. Penalizaciones estáticas

Un ejemplo es la técnica propuesta por Homaifar, Lai y Qi [38]. Consiste en definir factores de penalización cuyo valor no cambia durante el proceso evolutivo. La idea principal es definir varios niveles de violación y elegir un coeficiente de violación para cada uno de ellos, de manera que el coeficiente de penalización se incremente conforme alcanzamos niveles más altos de violación de las restricciones. Un individuo se evalúa usando la ecuación (3.12):

$$fitness_i = f_i(\vec{x}) + \sum_{j=1}^{n} R_{k,j} g_j(\vec{x}) \tag{3.12}$$

donde $R_{i,j}$ son los coeficientes de penalización utilizados y $k = 1, 2, ..., l$, siendo $l$ los niveles de violación definidos por el usuario.
El principal problema de esta técnica es que requiere la definición de $n(2l + 1)$ parámetros. De tal forma que si tenemos un problema moderadamente pequeño, con seis restricciones y dos niveles, tendríamos que definir treinta parámetros, lo cual es evidentemente excesivo.

#### 3.2.1.3. Penalizaciones dinámicas

Joines y Houck [41] propusieron una técnica en la que los factores de penalización cambian con respecto al tiempo. Los individuos de la generación $t$ se evalúan de acuerdo a la ecuación (3.13).

$$fitness_i(\vec{x}) = f_i(\vec{x}) + (Cxt)^{\alpha} \sum_{j=1}^{n} |g_i(\vec{x})|^{\beta} \tag{3.13}$$

donde $C, \alpha$ y $\beta$ son constantes definidas por el usuario. Los valores sugeridos por los autores para las constantes son: $C = 0.5, \alpha = 1$ y $\beta = 1$ ó 2.

La técnica es muy susceptible a los valores de los parámetros y suele converger prematuramente cuando éstos no son seleccionados adecuadamente. Sin embargo, parece funcionar muy bien cuando la función objetivo es cuadrática.

#### 3.2.1.4. Penalizaciones adaptativas

Bean y Hadj-Alouane [40] desarrollaron una técnica para adaptar penalizaciones con base en un proceso de retroalimentación del ambiente durante la corrida de un algoritmo genético. Cada individuo es evaluado usando la ecuación (3.14).

$$fitness_i(\vec{x}) = f_i(\vec{x}) + \lambda(t) \sum_{j=1}^{n} (g_j\vec{x})^2 \tag{3.14}$$

donde $\lambda(t)$ se actualiza cada $t$ generaciones usando las siguientes reglas:

$$\lambda(t) = \begin{cases} \frac{1}{\beta_1}\lambda(t) & \text{si se cumple el caso 1} \\ \beta_2\lambda(t) & \text{si se cumple el caso 2} \\ \lambda(t) & \text{de lo contrario.} \end{cases} \tag{3.15}$$

donde $\beta_1, \beta_2 > 1$ y con valores diferentes (para evitar ciclos). El caso 1 ocurre cuando el mejor individuo en las últimas $k$ generaciones fue siempre factible. El caso 2 ocurre cuando dicho individuo no fue nunca factible. Esta técnica lo que hace entonces es disminuir la penalización cuando el mejor individuo resulta consistentemente factible y aumentar la penalización cuando resulta infactible. Si en los cambios el mejor individuo es a veces factible y a veces no, entonces la penalización no se cambia. Obviamente, es necesario definir el valor inicial $\lambda_0$.

El principal problema de esta técnica es cómo elegir el factor inicial de penalización y el intervalo generacional (o sea, $k$) de manera que el monitoreo de factibilidad produzca resultados razonables.

Una $k$ muy grande o muy pequeña puede hacer que la función de penalización nunca cambie, en cuyo caso la técnica se reduce a una penalización estática tradicional. Finalmente, tampoco está claro cómo definir buenos valores de $\beta_1$ y $\beta_2$.

### 3.2.2. Otras técnicas

Además de las funciones de penalización, algunos investigadores han propuesto otras técnicas para manejo de restricciones. A continuación se presenta una breve revisión de los métodos más importantes reportados en la literatura especializada [4].

#### 3.2.2.1. Operadores y representaciones especiales

Algunos problemas pueden ser particularmente difíciles al ser tratados con las representaciones normales de los algoritmos evolutivos. Es esos casos puede ser más fructífero idear una representación especial para el problema y, de ser necesario, un conjunto de operadores especiales aplicables a dicha representación.

Un ejemplo de operadores especiales para espacios convexos es la primera versión de GENOCOP [58], el cual comienza con al menos una solución factible, y le aplica operadores que generan hijos que son combinaciones lineales de los individuos originales. Tales operadores son útiles porque el algoritmo está diseñado para trabajar únicamente con problemas con restricciones lineales. Posteriormente se desarrollaron otras versiones de GENOCOP, que además pueden manejar restricciones no lineales, pero tales propuestas no pertenecen a la clasificación de operadores especiales y por ello no se describen aquí.

Otra propuesta que cuenta con operadores especiales es el algoritmo genético consistente con las restricciones de Kowalczyk [46], la cual utiliza operadores especiales, así como un método especial de iniciación de la población para asegurar que los individuos sean consistentes con las restricciones. Sin embargo, tales operadores y el método de iniciación son muy costosos computacionalmente. Schoenauer y Michalewicz [79] han utilizado operadores que tratan de explorar la frontera entre las zonas factible y no factible. La motivación para desarrollar este tipo de operadores, es que en muchos problemas de optimización no lineal el óptimo global está ubicado en la frontera de la zona factible. El problema es que estos operadores son altamente especializados, al grado de tener que ser diseñados específicamente para un problema en particular.

Un tipo de técnicas importantes que se basan en representaciones especiales son los decodificadores, los cuales son una transformación que relaciona una solución factible con una solución codificada. La transformación debe ser construida de tal forma que cualquier posible solución producida por el algoritmo pueda ser transformada en una solución factible. La técnica más importante hasta la fecha que utiliza decodificadores, es la de los mapas homomorfos de Koziel y Michalewicz [47,48], que mapea la región factible de un problema a una figura regular (un hipercubo) en la cual es mucho más fácil moverse.

Una ventaja sobresaliente de esta técnica es que puede manejar cualquier tipo de restricciones, con zonas factibles convexas y no convexas, e incluso disjuntas. Esta propuesta es importante, por los resultados publicados de un grupo de funciones de prueba (propuesto por Michalewicz y Schoenauer, [59]), que fueron los mejores producidos por cualquier algoritmo evolutivo hasta ese momento, y que por tal motivo se han convertido en un punto de comparación para los algoritmos evolutivos con mecanismos para manejo de restricciones.

#### 3.2.2.2. Separación de restricciones y objetivos

Algunos enfoques tratan de forma separada a los objetivos y a las restricciones. Por ejemplo, Paredis [67] propuso un algoritmo co-evolutivo que utiliza dos subpoblaciones, una representa las soluciones del problema, y la otra representa a las restricciones. Cada

individuo de la segunda subpoblación es evaluado según la cantidad de individuos en la primera población que violan la restricción que representa.

Varios autores han propuesto métodos en los que se trata como superiores a los puntos factibles. Por ejemplo, Powell y Skolnick [69] propusieron un método donde los puntos factibles siempre tienen una aptitud entre -$\infty$ y 1, y los puntos infactibles siempre tienen una aptitud entre 1 y $+\infty$.

Deb también tiene un método similar [15], en el que los puntos infactibles siempre tienen una peor aptitud que los puntos factibles, porque se agrega la magnitud de las violaciones a la aptitud del peor individuo factible. Para hacer más simple esta asignación de aptitud, Deb utiliza torneos binarios, en los que las reglas son las siguientes:

1. Si un individuo es factible y el otro infactible, siempre gana el individuo factible.
2. Si ambos individuos son factibles, gana el que tenga mejor valor de la función de aptitud.
3. Si ambos individuos son infactibles, gana el individuo que tenga una magnitud más baja en la violación de restricciones.

Hinterding y Michalewicz [34] propusieron un método que complementa a los padres antes de hacer la recombinación. La selección de ambos padres se hace con criterios distintos: el primer padre se selecciona tomando en cuenta su aptitud y su factibilidad, mientras que el segundo se selecciona de forma que tenga el mínimo de restricciones satisfechas en común con el primero. Esta idea intenta producir descendientes que sean mejores que los padres, en el sentido de restricciones violadas. Hay más métodos que tratan en forma diferente a los objetivos y a las restricciones. Schoenauer y Xanthakis [80] extendieron un método llamado memoria conductista, para hacerlo capaz de manejar restricciones. Este método trabaja en varias fases, y en cada una se trata de satisfacer una de las restricciones, hasta que en la fase final se trata de optimizar la función objetivo, suponiendo que se tienen suficientes soluciones factibles en la población. Este método puede ser bastante sensible al orden en el que se satisfacen las restricciones.

También hay varios enfoques que utilizan técnicas de "optimización multiobjetivo". Dado que cada una de las restricciones puede tratarse como un objetivo en el que hay que minimizar la violación, podemos agregar el objetivo original al conjunto de restricciones, siendo todas estas funciones los objetivos por optimizar.

Ray et al. [73] propusieron un método donde se jerarquiza a los individuos con base en su valor de las funciones objetivo y a la violación de restricciones. Posteriormente se utiliza un mecanismo de complemento de restricciones como el de Hinterding y Michalewicz [34].

#### 3.2.2.3. Jerarquización estocástica

Este método de manejo de restricciones ha sido uno de los más exitosos y se ha mantenido como un método de referencia contra el que suelen compararse las nuevas propuestas. Fue propuesto por Thomas Runarsson y Xin Yao [76]. El objetivo principal de esta técnica

es balancear la influencia de la función objetivo y el grado de violación de las restricciones al momento de asignar el valor de aptitud a un individuo.

Esta técnica no requiere la definición de factores de penalización. Este mecanismo jerarquiza $\lambda$ posibles soluciones, utilizando un procedimiento similar al método de ordenamiento de la burbuja, de la siguiente forma: Debido a que determinar los valores óptimos para los factores de penalización es sumamente difícil, se define una probabilidad $P$ que determina el balance entre tener prioridad con respecto al valor de la función objetivo o tener prioridad con respecto a la violación de las restricciones. Se realizan comparaciones entre pares de posibles soluciones. La probabilidad de comparar las posibles soluciones con respecto al valor de la función objetivo es 1 si ambas soluciones son factibles; en otro caso, es $P$ . Los autores determinaron que se logran los mejores resultados, en varios problemas, cuando $0.4 < P < 0.5$.

Este procedimiento se muestra en el algoritmo 4, donde $U(0,1)$ es un generador de números aleatorios uniformemente distribuidos y $N$ es el número de veces que se recorre la población. En su forma original, la jerarquización estocástica emplea una estrategia evolutiva (EE) multi-miembro como motor de búsqueda, la cual realiza mutaciones generadas con una distribución normal de probabilidades. Además, utiliza un coeficiente para regular el tamaño de paso por cada variable de decisión en cada individuo. En caso de que alguna variable quede fuera del rango permitido después del proceso de mutación, ésta toma el valor del límite correspondiente. No se emplea un operador de recombinación.

**Algoritmo 4**: Jerarquización estocástica

```
1  Datos: Población de individuos que se desea jerarquizar.
2  Resultado: Población de individuos ordenados de acuerdo a jerarquía
3  begin
4      for i ← 1 a N do
5          for j ← 1 a λ − 1 do
6              if ψ(x⃗_j) == ψ(x⃗_{j+1}) == 0 || U(0,1) < Pf then
7                  if f(x⃗_j) > f(x⃗_{j+1}) then
8                      Intercambiar (x⃗_j, x⃗_{j+1}) ;
9              else
10                 if ψ(x⃗_j) > ψ(x⃗_{j+1}) then
11                     Intercambiar (x⃗_j, x⃗_{j+1}) ;
12         if No se realizó intercambio then
13             Terminar();
14 end
```

Los autores presentan una versión mejorada de su algoritmo en [77]. Pero es importante mencionar que el mecanismo para el manejo de restricciones permanece igual. La diferencia está en el proceso de mutación y en que se agrega una recombinación discreta similar al de la ED, en caso de que las variables generadas después del proceso de mutación estén fuera

del rango permitido. Los autores reportan resultados realizando 350,000 evaluaciones de la función objetivo, con la utilización de las trece funciones de prueba que aparecen en el apéndice A, en la p.g. 123, de esta tesis.

## 3.3. Resumen del capítulo

En la primera parte de este capítulo se introdujo el concepto de un buscador global para la solución de problemas definidos en espacios continuos. Se presentó una serie de métodos que trabajan con variables reales y se describieron algunas técnicas para el manejo de las restricciones.

La gran diversidad de técnicas descritas anteriormente son representativas del estado del arte y cada una tiene sus ventajas y desventajas para la solución de problemas de optimización. Los métodos que han presentado mejores soluciones en diversos conjuntos de problemas son los algoritmo de la ED y jerarquías estocásticas, los cuales utilizamos para nuestra propuesta.

La combinación de las técnicas planteadas en este capítulo y los mecanismos de búsqueda local crean algoritmos híbridos que generan mejores resultados en la solución de problemas de optimización. El siguiente capítulo describe mecanismos de búsqueda local que trabajan para problemas definidos en espacios continuos.

Capítulo 4

# Buscadores locales para espacios continuos

El término **local** utilizado frecuentemente en las metaheurísticas de búsqueda, se asocia al uso de **estructuras de entorno**, reflejando el concepto de proximidad o vecindad entre las soluciones alternativas de un problema.

Las soluciones creadas, en un determinado entorno, por un operador o mecanismo de generación se denominan **soluciones vecinas**. Estos mecanismos realizan un estudio local del espacio de búsqueda, el cual consiste en analizar el entorno de una solución para decidir cómo continuar el recorrido de la búsqueda.

En la figura 4.1 se muestra un ejemplo del proceso que sigue un algoritmo de búsqueda local, donde $\sigma_i$ son los puntos solución encontrados en cada iteración del algoritmo y $N(\sigma_i)$ es el vecindario asociado a cada solución con $i = 0, ..., 4$.

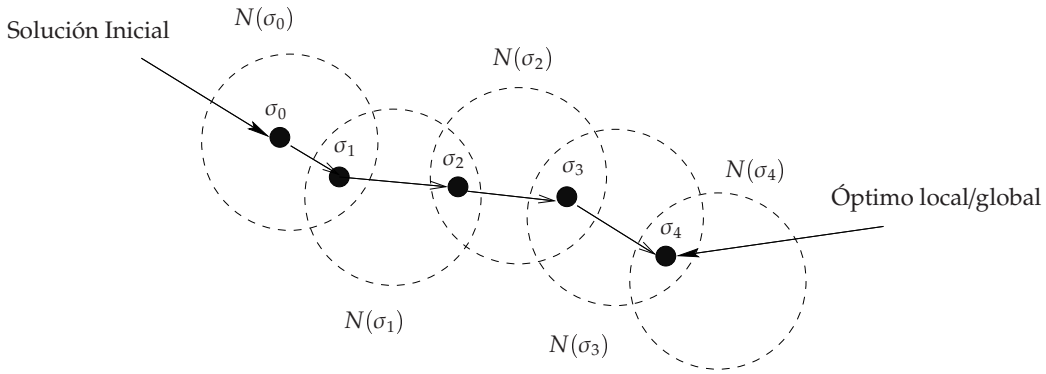



Figura 4.1: Ejemplo del mecanismo de búsqueda local

El algoritmo 5 es un procedimiento iterativo de búsqueda local que utiliza una solución actual o base, busca en su vecindario una mejor solución y, en caso de encontrarla, reemplaza la solución actual por ésta (ver figura 4.2).

Un mecanismo de búsqueda local se usa para hacer eficiente el refinamiento de las soluciones. Una vez que se ha ubicado una zona factible se recomienda utilizar este mecanismo para que agilice la convergencia al óptimo global.

**Algoritmo 5**: Algoritmo de búsqueda local

```
1 Datos: solución inicial s
2 Resultado: mejor solución dentro de la vecindad de s
3 begin
4     generar un vecindario N(s) alrededor de la solución base;
5     while s no sea un óptimo local o se cumpla con algún criterio do
6         s' ∈ N(s);
7         if s' es mejor que s then
8             s ← s';                                   ▷ con f(s) < f(s')
9 end
```

La búsqueda local en problemas con funciones definidas en espacios continuos tiene que enfrentar dificultades como: establecer la dirección de búsqueda, realizar movimientos pequeños que generen soluciones con una alta precisión y evitar caer en óptimos locales que a veces están muy alejados del óptimo global del problema (ver figura 4.3). En [30], se mencionan como posibles soluciones a estos problemas las siguientes:

- Permitir movimientos de empeoramiento de la solución actual. p. ej. recocido simulado y búsqueda tabú.
- Modificar la estructura de entornos. p. ej. búsqueda descendente basada en entornos variables [31].
- Inicializar la búsqueda desde otra solución, p. ej. búsqueda multiarranque [65].

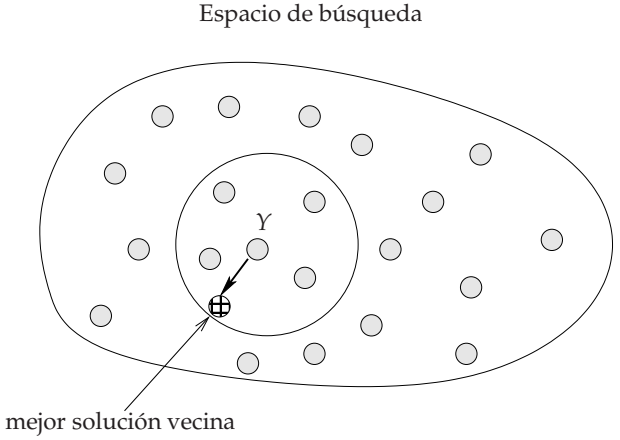



Figura 4.2: Búsqueda en el vecindario

Los algoritmos de búsqueda local en espacios continuos hacen uso de parámetros estratégicos (p. ej. control del tamaño de paso) para guiar la búsqueda. Generalmente, estos parámetros se adaptan con el propósito de incrementar la probabilidad de producir soluciones más eficientes. Todo procedimiento de búsqueda local requiere un número sustancial

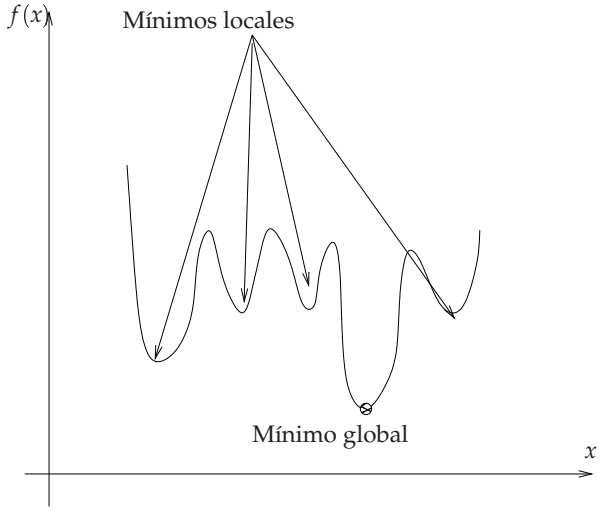



Figura 4.3: Problema de mínimos locales

de evaluaciones de la función objetivo.

Los mecanismos de búsqueda local pueden ser métodos numéricos tradicionales de optimización, los cuales generalmente realizan búsqueda local utilizando la información que proporciona la **derivada** de una función. Otras opciones son buscadores locales basados en **heurísticas** o en operadores que definan **estructuras de vecindarios**.

Para los fines de esta tesis, a continuación se describen una serie de mecanismos de búsqueda local para espacios continuos. Estos incluyen tanto procedimientos que hacen uso de la primera y segunda derivada, como los que se guían estocásticamente y los que utilizan operadores que trabajan en un vecindario.

## 4.1. Métodos de búsqueda directa

Los métodos de búsqueda directa requieren únicamente valores de la función objetivo y no necesitan información de la derivada de la función. Estos métodos son útiles cuando resulta muy difícil, muy laborioso o simplemente imposible obtener expresiones analíticas para la derivada de la función a optimizarse.

Este tipo de métodos resuelven problemas tanto unidimensionales como multidimensionales de optimización. En las funciones de una sola variable, los algoritmos tienen dos direcciones de búsqueda para modificar un punto. Sin embargo, en las funciones de $N$ variables, estos algoritmos tienen $2^N$ formas diferentes de definir direcciones de búsqueda, por lo que se requieren procedimientos que manipulen un conjunto de direcciones de búsqueda para poder lidiar con la no linealidad de las funciones.

Las desventajas de estos métodos son:

- No garantizan la convergencia al óptimo global.

- En la mayoría de los casos dependen de un punto inicial de búsqueda.
- En funciones multi-modales pueden quedar estancados en óptimos locales.

### 4.1.1. Método de patrones de búsqueda de Hooke-Jeves

El método de patrones de búsqueda fue propuesto en 1961 por Robert Hooke y T.A. Jeeves. La idea básica de este método es combinar diferentes direcciones univariables para generar una nueva dirección de búsqueda y encontrar un nuevo punto, creando un conjunto de direcciones de búsqueda de manera iterativa [39].

Los valores sucesivos de $\vec{x}$ pueden ser interpretados como puntos en un espacio $K-$dimensional. El procedimiento de ir de un punto dado a otro, se llama "movimiento" [39]. Un movimiento es exitoso si el valor de $f(\vec{x})$ disminuye; de lo contrario, es un fracaso. Este método realiza dos tipos de movimientos:

- **Movimiento exploratorio**. Este tipo de movimiento adquiere conocimientos acerca del comportamiento de la función. El conocimiento es independiente del éxito o fracaso de los movimientos de exploración, sin tener en cuenta cualquier evaluación cuantitativa de los valores de la función. Los movimientos exploratorios examinan la vecindad del punto actual para encontrar el mejor punto alrededor del mismo. De tal forma, los movimientos exploratorios examinan el comportamiento local de la función.

**Algoritmo 6**: Movimiento exploratorio

**Datos:** solución actual (punto base) $x_c$
**Resultado:** resultado del movimiento exploratorio
**begin**
  variable $x_i^c$ es perturbada por $\Delta_i$;
  hacer $i \leftarrow 1$, $x \leftarrow x^c$;
  **while** $i \neq N$ **do**
    Encontrar:
    $f \leftarrow f(x)$
    $f^+ \leftarrow f(x_i + \Delta_i)$ y
    $f^- \leftarrow f(x_i - \Delta_i)$;
    Encontrar $f_{\text{mín}} \leftarrow \text{mín}(f, f^+, f^-)$;
    Hacer $x \leftarrow f_{\text{mín}}$;
    $i \leftarrow i + 1$;
  $x$ es el resultado;
  **if** $x \neq x^c$ **then**
    reportar ÉXITO;
  **else**
    reportar FRACASO;
**end**

En el movimiento exploratorio, el punto actual es perturbado en la dirección positiva y en la dirección negativa a lo largo de cada variable, a razón de una variable a la vez y el mejor punto obtenido es almacenado. Al final de la perturbación de cada variable se cambia el punto actual por el mejor punto encontrado. Si el punto encontrado al final de todas las perturbaciones de las variables, es diferente al punto original, el movimiento exploratorio se considera exitoso; de lo contrario, se considera fallido (ver algoritmo 6).

- **Movimiento de patrones**. Se obtiene un nuevo punto saltando del mejor punto actual $x^c$ a lo largo de una dirección que conecte el mejor punto previo $x^{(k-1)}$ y el punto base actual de la manera siguiente:

$$x_p^{(k+1)} = x^{(k)} + (x^{(k)} - x^{(k-1)})$$

El método de Hooke-Jeeves consiste de una aplicación iterativa de un movimiento exploratorio en la vecindad del punto actual y de un salto subsecuente usando el movimiento de patrones. Si el movimiento de patrones no toma la solución a una mejor región, entonces no se acepta y se reduce el alcance de la búsqueda exploratoria. Esto se repite hasta alcanzar convergencia (ver algoritmo 7).

**Algoritmo 7**: Algoritmo de Hooke Jeeves

```
1  Datos: Punto inicial x^0, Δ_i(i ← 1,2,...,N), α (factor de reducción de paso), ϵ
   parámetro de terminación
2  Resultado: mejor solución
3  begin
4      hacer k ← 0;
5      while ‖Δ‖ < ϵ do
6          realizar un movimiento exploratorio con x^k como punto base;
7          hacer que x sea la salida del movimiento exploratorio;
8          if movimiento exploratorio es exitoso then
9              hacer x^(k+1) ← x;
10             repeat
11                 hacer k ← k + 1;
12                 efectuar el movimiento de patrones:
13                 x_p^(k+1) ← x^(k) + (x^(k) − x^(k−1));
14                 realizar otro movimiento exploratorio usando x_p^(k+1) como el punto
                   base.
15                 hacer que el resultado sea x^(k+1);
16             until f(x^(k)) ⩽ f(x^(k+1))
17 end
```

Debido a que la búsqueda depende en gran medida de los movimientos a lo largo de las direcciones coordenadas $(x_1, x_2, ...x_n)$ durante el movimiento exploratorio, el algoritmo puede converger prematuramente a una solución errónea, especialmente en el caso de

funciones con interacciones altamente no lineales entre las variables. El algoritmo también puede quedar atrapado en el ciclo de generación de movimientos aleatorios. Otra característica de este método es que termina sólo tras buscar exhaustivamente la vecindad del punto al que se ha convergido. Esto requiere un número elevado de evaluaciones de la función objetivo para poder converger a una solución que tenga una precisión razonable.

### 4.1.2. Método simplex de Nelder-Mead

El método simplex de búsqueda directa fue originalmente propuesto por Spendley, Hext y Himsworth en 1962 [85]. Posteriormente fue mejorado por John Nelder y Roger Mead en 1965 [64] y sus apellidos se usan desde entonces para nombrar el método. Es una técnica para minimizar una función objetivo en un espacio multi-dimensional. Este método utiliza el concepto de un simplex, el cual es un politopo [1] de $n + 1$ vértices en $n$ dimensiones. Por ejemplo, el segmento de línea que une dos puntos es un 1-simplex, el triángulo definido por tres puntos es un 2-simplex, y el tetraedro definido por cuatro puntos es un 3-simplex.

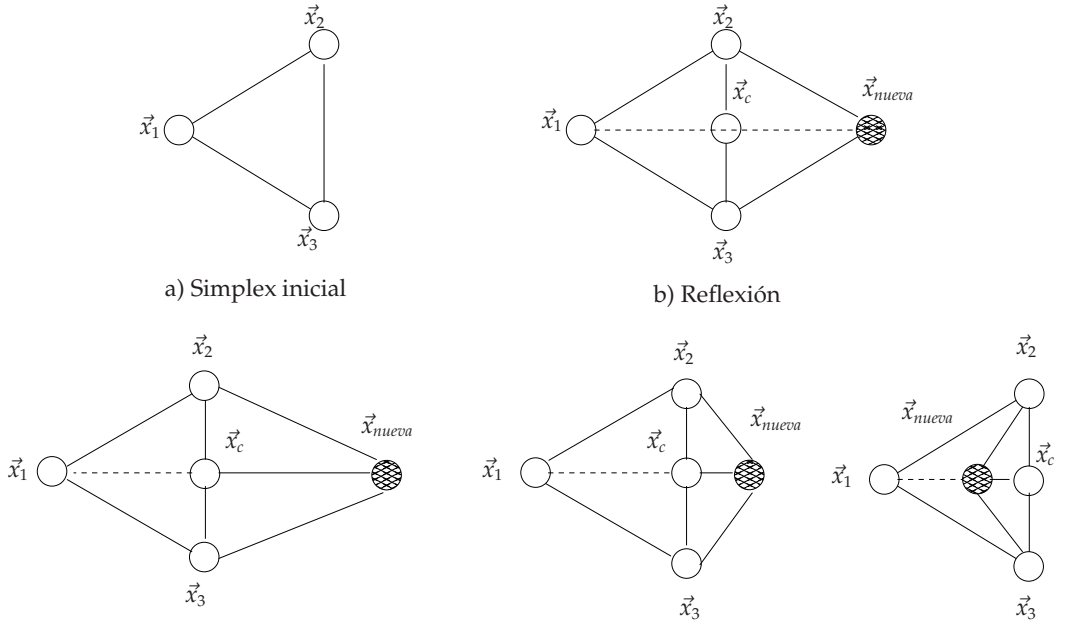



Figura 4.4: Ejemplo hipotético en dos dimensiones de la generación de nuevas soluciones mediante los operadores utilizados por el método de Nelder-Mead.

Un simplex puede ser construido a partir de la ecuación (4.1), donde $\vec{x}_0$ es un vértice base para generar los demás $\vec{x}_i$.

[1] En geometría un politopo es la generalización a cualquier dimensión de un polígono bidimensional, o un poliedro tridimensional.

$$\vec{x_i} = \vec{x_0} + \delta_1 u_i + \sum_{j=1, j\neq i}^{n} \delta_2 u_j \;\; i = 1, 2, ..., n \tag{4.1}$$

donde:

$i = 1, 2, ..., n$, $\delta_1 = \frac{\alpha}{n\sqrt{2}}(\sqrt{n+1} + n - 1)$, $\delta_2 = \frac{\alpha}{n\sqrt{2}}(\sqrt{n+1} - 1)$ y $u_j$ es el vector unitario a lo largo del $j-$ésimo eje coordenado. En el caso particular de $\alpha = 1$ se tendrá un simplex con vértices equidistantes, conocido como simplex regular.

Presuponiendo un problema de minimización, el método define $f(\vec{x_i})$ como el valor de la función para cada punto $\vec{x_i}$. Los puntos vértices $\vec{x_h}$, $\vec{x_l}$ corresponden al mayor y menor valor de la función objetivo, respectivamente. El punto $\vec{x_c}$, es el centroide del simplex (su cálculo se realiza a partir de la ecuación (4.2)). En cada iteración del algoritmo, el peor punto ($\vec{x_h}$) es reemplazado por un nuevo punto.

$$\vec{x_c} = \frac{1}{n} \sum_{i=1, i\neq h}^{n+1} \vec{x_i} \tag{4.2}$$

Este método hace uso de tres operaciones, *reflexión, contracción y expansión*. A continuación se describen dichas operaciones:

- **Reflexión**. Se aplica sobre el punto $\vec{x_h}$ hacia la cara opuesta para obtener una nueva solución con un valor $f(\vec{x})$ menor (ver figura 4.4(b)). El punto de reflexión $\vec{x_r}$ se calcula de la siguiente manera:

$$\vec{x_r} = (1 + \alpha)\vec{x_c} - \alpha\vec{x_h} \tag{4.3}$$

  donde $\alpha$ es una constante positiva, llamada coeficiente de reflexión. Si $f(\vec{x_r})$ es un valor entre $f(\vec{x_h})$ y $f(\vec{x_l})$, entonces $\vec{x_h}$ es reemplazado por $\vec{x_r}$ y se reinicia el procedimiento con el nuevo simplex.

- **Expansión**. Este operador se aplica sólo si el proceso de reflexión obtuvo un nuevo punto tal que, $f(\vec{x_r}) < f(\vec{x_l})$ ( o sea, si se produjo un nuevo mínimo) (ver figura 4.4(c)). Su cálculo se define por la ecuación (4.4).

$$\vec{x}_{nueva} = \gamma\vec{x_r} + (1 - \gamma)\vec{x_c} \tag{4.4}$$

  El coeficiente de expansión $\gamma > 1$ define la razón de la distancia entre $\vec{x}_{nueva}$ y $\vec{x_c}$. Si $f(\vec{x}_{nuevo}) < f(\vec{x_l})$ se reemplaza $\vec{x_h}$ por $\vec{x_r}$ antes de reiniciar el procedimiento.

- **Contracción**. Si en el proceso de reflexión se genera un punto $\vec{x_r}$ tal que $f(\vec{x_r}) > f(\vec{x_h})$, entonces se considera que la reflexión falló, por lo tanto se realiza una *contracción interna*, (ver figura 4.4(d)). El nuevo punto se genera de la siguiente manera:

$$\vec{x}_{nueva} = \beta\vec{x_h} + (1 - \beta)\vec{x_c} \tag{4.5}$$

  El coeficiente de contracción $\beta$ está definido entre 0 y 1. Por otro lado, si el proceso de la reflexión genera un punto $\vec{x_r}$ tal que $f(\vec{x_r}) > f(\vec{x_i})$, para toda $i$ excepto $i = h$,

y $f(\vec{x}_r) < f(\vec{x}_h)$, entonces se realiza una contracción menor al caso anterior llamada contracción externa (ver figura 4.4(e)), de la siguiente forma:

$$\vec{x}_{nueva} = \beta\vec{x}_h - (1+\beta)\vec{x}_c \tag{4.6}$$

Los coeficientes $\alpha$, $\beta$ y $\gamma$ son los factores que determinan el cambio en el volumen del simplex. Los parámetros que a Nelder y Mead les dieron buenos resultados y han sido adoptados como estándares son:

$$\alpha = 1,\ \beta = \tfrac{1}{2},\ \gamma = 2 \text{ y } \delta = \tfrac{1}{2}$$

El algoritmo 8 muestra el funcionamiento completo del método.

**Algoritmo 8**: Método de Nelder-Mead

**Datos:** $\gamma > 1$ (factor de expansión), $\beta \in (0,1)$ (factor de contracción) y $\epsilon$ tolerancia
**Resultado:** mejor solución encontrada
**begin**
  **while** *no se cumpla criterio de terminación:* $Q < \epsilon$ **do**
    encontrar $x_h$ (el peor punto), $x_l$ (el mejor punto) y $x_g$ (el segundo peor punto);
    calcular el centroide: $x_c \leftarrow \frac{1}{n} \sum_{i=1, i\neq h}^{n+1} x_i$;
    calcular el punto reflejado: $x_r \leftarrow 2x_c - x_h$;
    $x_{nueva} \leftarrow x_r$;
    **if** $f(x_r) < f(x_l)$ **then**
      realizar expansión: $x_{nueva} \leftarrow (1+\gamma)x_c - x_h$;
    **else**
      **if** $f(x_r) \geqslant f(x_h)$ **then**
        realizar contracción: $x_{nueva} \leftarrow (1-\beta)x_c + \beta x_h$;
      **else**
        **if** $f(x_g) < f(x_r) < f(x_h)$ **then**
          realizar contracción: $x_{nueva} \leftarrow (1+\beta)x_c - x_h$;
    calcular $f(x_{nueva})$;
    $x_h \leftarrow x_{nueva}$;
    calcular $Q \leftarrow \left\{ \sum_{i=1}^{n+1} \frac{(f(x_i) - f(x_c))^2}{n+1} \right\}^{\frac{1}{2}}$;
**end**

## 4.2. Métodos de búsqueda basados en el gradiente

Los métodos de búsqueda basados en el gradiente utilizan información de la derivada de la función, como estrategia para moverse entre una solución y otra. Estos métodos hacen

uso de las técnicas clásicas de cálculo diferencial, las cuales pueden encontrar el máximo o el mínimo de una función de varias variables. Estos métodos presuponen que la función es dos veces diferenciable con respecto a cada variable de decisión y que todas las derivadas son continuas. Sin embargo, es importante hacer énfasis en que estos procedimientos se pueden aplicar sólo si tanto la función objetivo como las restricciones son diferenciables.

El gradiente de una función es un vector de $n$ componentes dado por la siguiente ecuación:

$$\nabla f = \begin{Bmatrix} \delta f/\delta x_1 \\ \delta f/\delta x_2 \\ . \\ . \\ . \\ \delta f/\delta x_n \end{Bmatrix} \tag{4.7}$$

El gradiente tiene una propiedad muy importante. Si nos movemos a lo largo de la dirección del gradiente desde cualquier punto en un espacio $n-$dimensional, el valor de la función se incrementa con la mayor velocidad posible [72].

Por otro lado, los métodos de gradiente requieren una cantidad considerable de experimentación para determinar los tamaños de paso que permitan llegar al óptimo global de una función.

### 4.2.1. Método de ascenso o descenso acelerado

La dirección del gradiente es denominada dirección del ascenso empinado. Sin embargo, la dirección del ascenso empinado es una propiedad local y no una global. Esto se ilustra con la figura 4.5, donde los vectores que van de $1-1^{'}, 2-2^{'}, ...5-5^{'}$ son las direcciones de ascenso indicadas por el gradiente de la función, lo que quiere decir que el valor de la función se incrementa en cada paso.

Una característica importante es el cálculo del tamaño de paso para cada dirección, ya que si efectuamos movimientos infinitamente pequeños, el número de evaluaciones requeridos para llegar al óptimo global será muy elevado.

El vector de gradiente representa la dirección del *ascenso empinado*, por lo que el negativo del vector de gradiente denota la dirección de *descenso empinado*. Todos los métodos de descenso hacen uso del vector de gradiente (ya sea de forma directa o indirecta) para encontrar las direcciones de búsqueda.

Para la evaluación del gradiente se requiere el cálculo de las derivadas parciales de la función a optimizarse $\delta f/\delta x_i, i = 1, 2, ..., n)$.

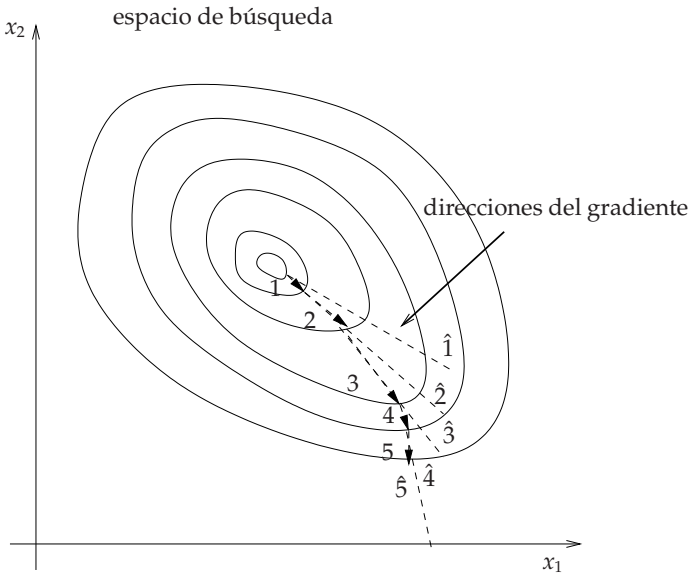



Figura 4.5: Ejemplo de las direcciones de búsqueda al aplicar gradiente

Desafortunadamente, hay tres situaciones en las cuales la evaluación del gradiente presenta problemas:

- La función es diferenciable en todos los puntos, pero el cálculo de las componentes del gradiente $\delta f/\delta x_i$, es sumamente difícil o imposible.
- Las expresiones para las derivadas parciales $\delta f/\delta x_i$ pueden obtenerse, pero requieren mucho tiempo de cómputo para su evaluación.
- El gradiente $\nabla f$ no está definido en todos los puntos.

Para solucionar los primeros dos casos, se hace uso de las diferencias finitas, las cuales aproximan derivadas. Esta técnica se emplea a menudo en análisis numérico, especialmente en ecuaciones diferenciales numéricas ordinarias, ecuaciones en diferencias y ecuación en derivadas parciales [72]. Para el cálculo de las derivadas de primer orden se tiene la ecuación (4.8) y las de segundo orden están dadas por la ecuación (4.9).

$$\left.\frac{\delta f(x)}{\delta x_i}\right|_{x^t} = \frac{f(x_i^t + \Delta x_i^t) - f(x_i^t - \Delta x_i^t)}{2\Delta x_i^t} \tag{4.8}$$

$$\left.\frac{\delta^2 f(x)}{\delta^2 x_i^2}\right|_{x^t} = \frac{f(x_i^t + \Delta x_i^t) - 2f(x_i^t) + f(x_i^t - \Delta x_i^t)}{(\Delta x_i^t)^2} \tag{4.9}$$

Si $\Delta x_i$ toma un valor muy pequeño, la diferencia entre los diferentes puntos en los cuales se evalúa la función será también muy pequeña y podrían predominar los errores de redondeo. Por otra parte, si $\Delta x_i$ es muy grande, el error de truncamiento podría predominar en el cálculo del gradiente.

Para la solución del tercer problema se recomienda usar un método de búsqueda directa [6].

## 4.3. Métodos basados en heurísticas

Según George Polya [71], la base de una heurística reside en la experiencia de resolver problemas y en ver cómo se hace. Por eso se dice que hay búsquedas ciegas, búsquedas heurísticas y búsquedas racionales.

Reeves [74] define una heurística como una técnica que busca soluciones buenas (casi óptimas) a un costo computacional razonable, aunque sin garantizar factibilidad u optimalidad de las mismas. En algunos casos ni siquiera puede determinarse qué tan cerca del óptimo se encuentra una solución producida por una heurística.

Las heurísticas generalmente son usadas para obtener soluciones subóptimas de problemas cuya optimización no es factible en tiempos polinomiales. Estos métodos incluyen la elaboración de estrategias y técnicas que faciliten la búsqueda de varias alternativas para solucionar los problemas.

Dado que las heurísticas utilizan procedimientos estocásticos, es indispensable conocer los casos en los que son aplicables y los límites de su uso. En ingeniería deben considerarse como ayudas o apoyos para hacer estimaciones rápidas y diseños preliminares.

Ciertas heurísticas también pueden ser usadas como procedimientos de búsqueda local en espacios continuos. Dichos procedimientos deben trabajar en vecindarios o entornos como es el caso de la búsqueda tabú.

### 4.3.1. Búsqueda tabú

La Búsqueda Tabú propuesta por Glover en 1989 es un procedimiento de búsqueda por entornos cuya característica distintiva es el uso de memoria adaptativa y estrategias especiales de resolución de problemas [26].

La memoria adaptativa permite: restringir el entorno de búsqueda e introducir mecanismos de reinicialización de la búsqueda mediante intensificación sobre zonas del espacio de búsqueda ya visitadas, o diversificación sobre posibles zonas del espacio de búsqueda poco visitadas.

La búsqueda tabú utiliza un procedimiento de búsqueda local o por vecindades para moverse iterativamente desde una solución $x$ hacia una solución $x'$ en la vecindad de $x$, hasta satisfacer algún criterio de parada [26]. Se basa fundamentalmente en la utilización de un historial de búsqueda, que permite ejecutar su estrategia de análisis y exploración de diferentes regiones del espacio de búsqueda. Este historial o memoria se implementa como una "lista" tabú. En cada iteración se elige la mejor solución entre las permitidas y se añade a la lista tabú, donde se mantienen las soluciones recientes que se excluyen de las siguientes iteraciones.

Los aspectos principales de esta técnica son:

- Permite movimientos de empeoramiento para escapar de óptimos locales.

**Algoritmo 9**: Algoritmo de búsqueda tabú

```
1  Datos: soluciones iniciales F
2  Resultado: mejor solución encontrada x
3  begin
4      selecciona x ∈ F (F representa soluciones factibles);
5      while No se cumpla el criterio de paro do
6          x* ← x (x* es la mejor solución encontrada hasta el momento);
7          c ← 0 (contador de iteraciones);
8          T conjunto de movimientos "tabú";
9          (N(x) es la función vecindario);
10         T ← ∅;
11         if (N(x) − T) es igual a ∅ then
12             T ← ∅;
13             genera vecindario;
14         else
15             c ← c + 1; selecciona n_c en N(x) − T de cada:
16             n_c(x) ← f(n(x) : n ∈ N(x) − T)
17             f(x) es la función de evaluación definida por el usuario
18         x ← n_c(x);
19         if f(x) < f(x*) then
20             x* ← x;
21         actualiza T ;
22 end
```

- Evita recorridos cíclicos, incorporando un mecanismo de generación de vecinos que evita la exploración de zonas del espacio de búsqueda que ya han sido visitadas (generación de entornos tabú restringidos).
- Emplea mecanismos de reinicialización para mejorar la capacidad del algoritmo para la exploración-explotación del espacio de búsqueda (intensificación y diversificación).

Para realizar los puntos anteriores se utilizan dos estructuras de memoria adaptativas distintas: memoria de corto plazo o lista tabú y memoria de largo plazo.

La memoria de corto plazo guarda información que permite guiar la búsqueda de forma inmediata, desde el comienzo del procedimiento (generación de entornos tabú restringidos).

La memoria de largo plazo guarda información que permite guiar la búsqueda *a posteriori*, después de una primera etapa en la que se han realizado una o varias ejecuciones del algoritmo aplicando la memoria a corto plazo. La información guardada en esta memoria se usa para comenzar con la búsqueda desde otra solución inicial.

Un cierto movimiento puede cambiar su estatus prohibido o "tabú" cuando se satisface

un cierto criterio de aspiración. Debido al uso de memoria, la búsqueda tabú es capaz de escapar de óptimos locales, a diferencia de otros buscadores locales convencionales.

En conclusión, la búsqueda tabú consta de tres componentes principales:

- Una memoria de corto plazo que evita que el algoritmo se cicle.
- Una memoria intermedia que permite intensificar la búsqueda.
- Una memoria de largo plazo que permite diversificar la búsqueda.

El algoritmo 9 muestra el procedimiento completo de la búsqueda tabú.

## 4.4. Operadores genéticos basados en vecindarios

Existen varios operadores de cruza para algoritmos evolutivos, con los cuales los descendientes se generan a partir de una cierta *distribución de probabilidades* de acuerdo a la posición relativa de los padres. Usualmente, estos operadores emplean más de dos padres [28]. Estas cruzas se dividen en dos tipos [3]:

- Cruzas que generan descendientes **cercanos a los padres**
- Cruzas que generan descendientes **cercanos al centro geométrico de los padres**

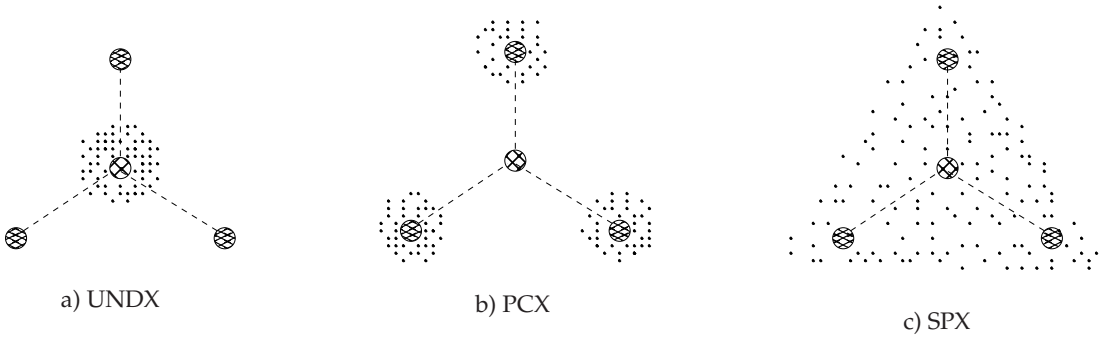



Figura 4.6: Ejemplo de distribución de descendientes en tres tipos de cruzas basadas en vecindarios: a) cruza con distribución normal unimodal (UNDX), b) recombinación céntrica de los padres (PCX) y c) cruza simplex (SPX)

### 4.4.1. Cruza con distribución normal unimodal (UNDX)

Este operador de cruza trabaja con $n$ vectores padre iniciales y una distribución normal de probabilidad, siguiendo los pasos del algoritmo 10.

**Algoritmo 10**: Operador de cruza UNDX

**Datos:** puntos padre iniciales
**Resultado:** descendientes al aplicar operador
**begin**
  selecciona $(\mu - 1)$ padres aleatoriamente
  calcula el promedio $\vec{g}$ de esos padres
  calcula $(\mu - 1)$ vectores de dirección formados por $(\vec{d}^{(i)} \leftarrow \vec{x}^{(i)} - \vec{g})$
  se calcula el vector $\vec{e}^{i} \leftarrow \frac{\vec{d}^{i}}{|\vec{d}^{i}|}$
  se elige aleatoriamente otro padre $\vec{x}^{(\mu)}$
  se calcula la longitud $D$ del vector $(\vec{x}^{(\mu)} - \vec{g})$ ortogonal a todos los $\vec{e}^{i}$
  de lo anterior se calcula el descendiente como:
  $\vec{y} \leftarrow \vec{g} + \sum_{i=1}^{\mu-1} \omega_i |\vec{d}^{i}| \vec{e}^{i} + \sum_{i=\mu}^{n} \nu D \vec{e}^{i}$
  donde $\omega_i$ y $\nu_i$ son variables normalmente distribuidas con media cero y varianza $\sigma_\varsigma^2$ y $\sigma_\nu^2$, respectivamente .
  Se sugiere que $\omega_i \leftarrow \frac{1}{\sqrt{\mu-2}}$ y $\nu_i \leftarrow \frac{0.35}{\sqrt{n-\mu-2}}$ y $\mu \leftarrow 3$ a 7
**end**

Kita y Yamamura [44] sugieren $\sigma_\varsigma = 1/\sqrt{\mu - 2}, \sigma_n = 0.35/\sqrt{n - \mu - 2}$, respectivamente e indican que $\mu = 3$ a 7 muestra un buen desempeño. Es interesante tener en cuenta que cada descendiente se crea alrededor del vector promedio $\vec{g}$. La probabilidad de crear un descendiente lejos del vector $\vec{g}$ es muy baja, y por el contrario, la probabilidad máxima se asigna a vectores cerca de $\vec{g}$. La figura 4.6 a) muestra tres padres y varios descendientes creados por el operador UNDX.

### 4.4.2. Recombinación céntrica de los padres (PCX)

El operador **PCX** asigna una alta probabilidad al hecho de que los descendientes estén cerca de los individuos base (padres) seleccionados. Su funcionamiento en el algoritmo 11 es a partir de la modificación del operador UNDX.

Este operador tiene una mayor probabilidad de crear descendientes cerca de los padres. La figura 4.6 b) muestra la distribución de soluciones descendientes con tres padres. Puesto que los padres individuales tienen cierta "Aptitud" en el operador de selección, se puede suponer que las soluciones cerca de estos padres pueden ser buenos candidatos, en particular en problemas de búsqueda en espacios continuos. Por el contrario, puede ser muy difícil encontrar soluciones buenas cerca del centro de gravedad de los padres.

**Algoritmo 11**: Operador de cruza PCX

**Datos:** puntos padre iniciales
**Resultado:** descendientes al aplicar operador
**begin**
  calcula el vector promedio $\vec{g}$ de $\mu$ padres
  se asigna la misma probabilidad de que el descendiente esté cerca de alguno de sus padres
  se calcula el vector dirección $\vec{d}^{(p)} \leftarrow \vec{x}^{(p)} - \vec{g}$
  se calculan las distancias $D_i$ perpendiculares a $\vec{d}^p$
  se calcula la media $\overline{D}$
  el descendiente se calcula mediante:
  $\vec{y} \leftarrow \vec{d}^p + \omega_\varsigma \vec{d}^p \sum_{i=1, i\neq p}^{\mu} \omega_\nu \overline{D} \vec{e}^i$
  donde $\vec{e}^i$ son las bases ortonormales sobre los subespacios perpendiculares a $\vec{d}^{(p)}$ y $\omega_\varsigma$ y $\omega_\nu$ son variables con distribución normal y con varianza $\sigma_\varsigma^2$ y $\sigma_\nu^2$, respectivamente.
**end**

### 4.4.3. Cruza simplex (SPX)

Un operador de cruza muy conocido en algoritmos genéticos para codificación real es el de BLX$-\alpha$ de Eshelman y Schaffer [21] que se muestra en la figura 4.7. BLX-$\alpha$ selecciona uniforme e independientemente nuevos individuos con valores que se encuentran entre $[d_i - \alpha d_i, d_i + \alpha d_i]$ para cada eje diagonal. Eshelman, Mathias y Schaffer extendieron BLX-$\alpha$ proponiendo box-BLX-$\alpha$-$\beta$, pool-wise box-BLX-$\alpha$-$\beta$ y directional-BLX-$\alpha$-$\beta$-$\gamma$ [20]. Existen estudios que justifican en qué casos es más adecuado usar cada una de estas versiones. Por ejemplo, si la distribución de la población se encuentra sobre la región que forma una diagonal en el espacio del problema, tiene más sentido tomar muestras en una región que es paralela a la diagonal [89].

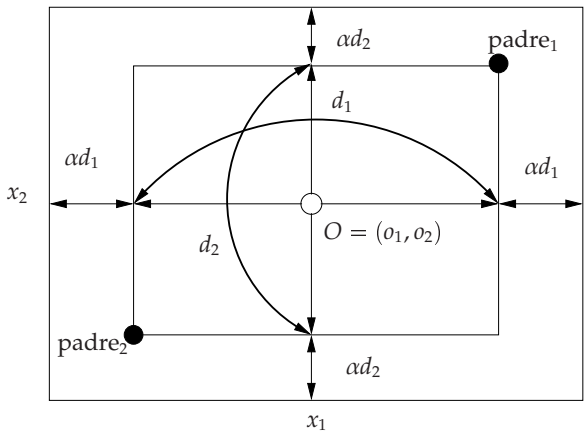



Figura 4.7: Operador BLX-$\alpha$ con $n = 2$

Por otro lado, el operador **SPX** crea descendientes distribuidos uniformemente alrededor del centroide de sus padres, restringiéndolos a un área predeterminada por un $k-$simplex. El simplex crossover (SPX) tiene características tanto de box y directional-BLX con propiedades del *simplex* y $m-$ vectores (padres) con $m \geqslant 2$ .

En $R^{(n)}, n+1$ puntos que son independientes unos con otros forman un simplex. Por simplicidad, primero consideremos una cruza SPX con tres padres en un espacio bidimensional de búsqueda como se muestra en la figura 4.8, donde $X^{(1)}, X^{(2)}$ y $X^{(3)}$ son tres vectores solución base o padres. Entonces, esos vectores forman un simplex. Tal como se realiza en el BLX$-\alpha$, este simplex se expande en cada dirección $(X^{(j)} - O)$ por $(1+\epsilon)(\epsilon \geqslant 0)$ veces, donde $O$ es el centro de masa de los tres padres, calculado como:

$$O = \frac{1}{3}\sum_{j=1}^{3} X^{(j)} \tag{4.10}$$

y

$$Y^{(j)} = (1+\varepsilon)(X^{(j)} - O). \tag{4.11}$$

Posteriormente, se generan tres descendientes seleccionando uniformemente valores para expandir el simplex.

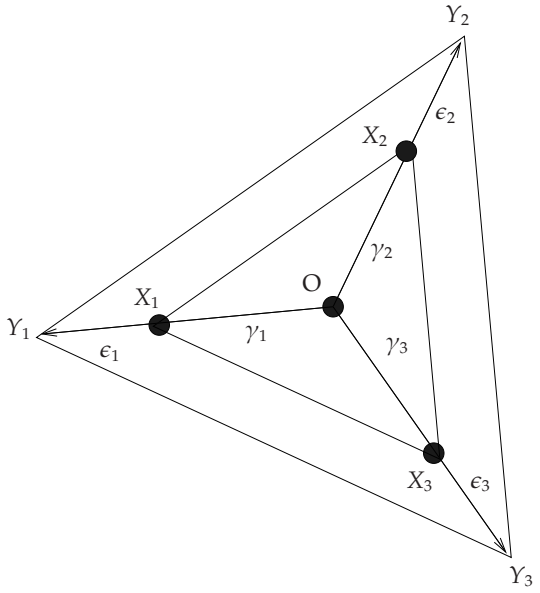



Figura 4.8: Representación gráfica del operador simplex

En general, el SPX se especifica como SPX$-n-m-\varepsilon$, donde $n$ es el número de direcciones del espacio de búsqueda, $m$ es el número de padres y $\varepsilon$ es un parámetro de control que define la tasa de expansión. Por lo tanto, el SPX que se muestra en la figura 4.8 se especifica como SPX$-2-3-\varepsilon$. Podemos ahora definir SPX$-n-m-\varepsilon$ para un caso más general donde $m$ se encuentra en el rango $[2, n+1]$: sea $X = (x_1, \ldots, x_n)$ un vector $n-$dimensional de números reales representando una posible solución (cromosoma). En

SPX-$n$-$m$-$\varepsilon$, $(m \leqslant n+1)$ individuos $X^{(j)} = (x_1^{(j)}, \ldots, x_n^{(j)}), j = 1, \ldots, m$ son seleccionados aleatoriamente de la población para ser cruzados $X_1, \ldots, X_N$.

Posteriormente, se divide $R^n$ en $h$ conjuntos de $R^{m-1}$ y en $R^q$ espacios por combinación aleatoria de $n$ dimensiones de coordenadas sin duplicados como:

$$R^n = \underbrace{R^{m-1} \times \ldots \times R^{m-1}}_{h} \times R^q, \tag{4.12}$$

donde, $h$ es la parte entera de $(n/(m-1))$ y $q$ el residuo de $(n/(m-1))$. En cada $R^{(m-1)}$ se crea un *simplex* usando $m$ vectores padres que pertenecen a $R^{m-1}$. Después, se generan $m$ vectores de descendientes seleccionando valores uniformemente en cada $R^{m-1}$. Finalmente se reemplazan los valores del vector generados con los valores del vector de $X^{(j)}$ en las posiciones correspondientes a la dimensión de $R^n$, y así se obtiene el vector $m$ de descendencia $X'^{(j)} = (x_1'^{(j)}, \ldots, x_n'^{(j)}), i = 1, \ldots, m$. Aquí dejamos $q$ valores de parámetros de $x_1'^{(j)}$ que pertenecen a $R^q$ sin cambios, v.g., $x_k'^{(j)} = x_k^{(j)} | \forall k \in R^q, j = 1, \cdots, m$.

Ahora, consideremos el otro extremo, donde $m$ es 2, v.g., SPX-$n$-2-$\varepsilon$. En este caso $R^{m-1}$ es $R^1$. Como un simplex en un espacio uni-dimensional es un *segmento*, un parámetro de descendencia en un eje $x_i$ se obtiene muestreando uniformemente el valor del segmento, como se muestra en la figura 4.9. Este muestreo es idéntico al muestreo de BLX-$\alpha$ en la figura 4.7, v.g., BLX-$\alpha \equiv$SPX$-n-2-\varepsilon$. Además, puede verse que el valor de $\varepsilon$ corresponde a $2 \times \alpha$ del BLX-$\alpha$.

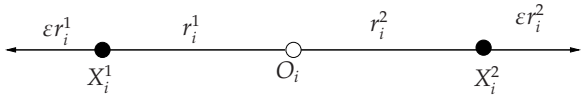



Figura 4.9: Ejemplo de SPX$-n-2-\varepsilon$

De esta forma, esta variante de SPX es realmente una extensión de BLX-$\alpha$ y tiene las siguientes características:

- Como un simplex es básicamente independiente de los sistemas de coordenadas, el operador SPX puede heredar esta independencia.
- Como los valores de los vectores de descendencia son muestreados uniformemente con $m$ vectores padres, éstos heredan características de los padres y el muestreo puede reflejar una cierta ligadura entre los padres.
- Como SPX mantiene un buen balance entre exploración y explotación en la generación de descendencia, trabaja bien sobre funciones que tienen una alta multimodalidad.
- Como el muestreo uniforme en un simplex puede ser realizado con procedimientos sencillos, SPX es un operador simple y no consume demasiado tiempo de cómputo.

### 4.4.4. Cruza PBX

Esta técnica fue propuesta por Syswerda en [91]. Este operador de cruza extiende la funcionalidad del operador de cruza $BLX - \alpha$ presentado por Eshelman y Schaffer (1993). Se llama parent-centric $BLX - \alpha$ ($PBX - \alpha$) y se describe a continuación.

Dados dos vectores $X = (x_1, ...., x_n)$ y $Y = (y_1, ...., y_n)$ $(x_i, y_i \in [a_i, b_i]$ , $i = 1, ..., n)$ los cuales son dos cromosomas con codificación real que han sido seleccionados para aplicar el operador de cruza sobre éstos. $PBX - \alpha$ genera (aleatoriamente) uno de los posibles descendientes: $Z_1 = (z_1^1, ..., z_n^1)$ o $Z_2 = (z_1^2, ..., z_n^2)$, donde $z_i^1$ es un número elegido aleatoriamente en el intervalo $[l_i^1, u_i^1]$ con:

$l_i^1 = max\{a_i, x_i - I \cdot \alpha\}$ y $u_i^1 = min\{b_i, x_i + I \cdot \alpha\}$

$z_i^2$ se elige en el intervalo $[l_i^2, u_i^2]$ con la ecuación (4.13)

$$l_i^2 = max\{a_i, y_i - I \cdot \alpha\} \quad \text{y} \quad u_i^2 = min\{b_i, y_i + I \cdot \alpha\} \tag{4.13}$$

donde $I = |x_i - y_i|$.

### 4.4.5. Operador de mutación BGA

El operador de mutación BGA es empleado para mantener diversidad en la población [63, 84]. En caso de trabajar con codificación real, uno de los temas de mayor importancia incluye el control de la proporción o la intensidad en la cual los genes en codificación real mutan, v.g., el tamaño de paso [2]. Se han sugerido distintas técnicas para el control del tamaño de paso durante la ejecución de un algoritmo genético con codificación real [32,83]. Un ejemplo es la mutación no uniforme, la cual es considerada como uno de de los operadores de mutación más adecuados para codificación real [33]. Su idea principal es decrementar el tamaño de paso durante el progreso de la ejecución. En este sentido, el algoritmo realiza una búsqueda uniforme en el espacio inicial y localmente en una etapa posterior, favoreciendo la mejora local. Un método alternativo es el propuesto en [49] donde un conjunto discreto de tasas de mutación se pone a disposición del algoritmo, el cual se puede autoadaptar para usar cualquiera de estos conjuntos.

Supongamos que $C = (c_1, \ldots, c_i, \ldots, c_n)$ es un cromosoma y $c_i \in [a_i, b_i]$ es un gen a ser mutado. El gen, $c_i'$, resultante de la aplicación de este operador es:

$$c_i' = c_i \pm rang_i \cdot \sum_{k=0}^{15} \alpha_k 2^{-k}, \tag{4.14}$$

donde $rang_i$ define el rango de mutación y normalmente se establece en $0.1 \cdot (b_i - a_i)$.El signo + o − es seleccionado con una probabilidad de 0.5 y $\alpha_i \in 0, 1$ se genera aleatoriamente con $p(\alpha_i = 1) = \frac{1}{16}$. Usando este operador se generan valores en el intervalo $[c_i - rang_i, c_i + rang_i]$, con una probabilidad muy alta de generar un vecindario de $c_i$. La proximidad mínima posible de este operador tiene una precisión de $rang_i \cdots 2^{-15}$.

## 4.5. Resumen del capítulo

En este capítulo se estudiaron 4 grupos de mecanismos de búsqueda local, los cuales se basan: en búsqueda directa, en el gradiente, en heurísticas y en operadores genéticos basados en vecindarios, siendo este último en el que enfocamos nuestra propuesta.

Un buscador local debe considerar estrategias de solución para la definición de un subespacio de búsqueda, para la estrategia de reemplazo de los vectores, para la frecuencia y tiempo de uso, etc. Estos aspectos hacen que la investigación entorno a los algoritmos de búsqueda local sea de gran interés en la actualidad.

En el siguiente capítulo estudiaremos algoritmos del estado del arte que combinan algunos de los mecanismos mencionados en este capítulo, con técnicas de búsqueda global, los cuales han demostrado acelerar la convergencia a mejores soluciones en el espacio de búsqueda de un problema.

Capítulo 5

# Algoritmos meméticos en la literatura

Los algoritmos meméticos (AMs) son métodos que combinan técnicas heurísticas de exploración y procedimientos de búsqueda local. Ambos métodos trabajan en forma complementaria para encontrar soluciones factibles a problemas de optimización.

Por un lado el mecanismo de búsqueda global localiza regiones prometedoras en el espacio de soluciones del problema, mientras que el mecanismo de búsqueda local mejora dichas regiones refinando los valores del vector de las variables de cada solución.

El diseño de AMs debe considerar un balance entre estos dos mecanismos, involucrar procedimientos estratégicos de reemplazo y métodos de conservación de la diversidad con la finalidad de agilizar la velocidad de convergencia a óptimos globales de diversos problemas.

En los capítulos 3 y 4 se mencionaron métodos eficaces de búsqueda global y local para espacios continuos. En este capítulo se describen AMs competitivos, que hacen uso de los métodos vistos, para la resolución de problemas de optimización mono-objetivo con y sin restricciones. El estudio de estos AMs tuvo la finalidad de comparar métodos, analizar ventajas de sus procedimientos, además de haber sido utilizados como base para la propuesta de AM introducida en esta tesis.

## 5.1. AMs para optimización mono-objetivo sin restricciones

En los problemas de optimización mono-objetivo sin restricciones, los AMs no requieren utilizar un mecanismo para el manejo de restricciones, por lo que únicamente se enfocan en mantener un balance adecuado entre la búsqueda global y local del algoritmo.

A continuación se describen AMs competitivos de la literatura especializada.

### 5.1.1. RCMA-XHC: Algoritmo genético y operador de cruza adaptativo como buscador local

Lozano et al. [50] propusieron un AM llamado RCMA-XHC, el cual está integrado por un mecanismo de búsqueda global basado en un algoritmo genético (AG) con codificación real. Como mecanismo de búsqueda local, usan un operador de cruza del tipo "escalamiento de colina", *hillclimber*.

**Algoritmo 12**: Algoritmo XHC

**Datos:** $p_1, p_2, n_{off}, n_{it}$
**Resultado:** $p_1^{'}, p_2^{'}$
**begin**
  $p_1^{'} \leftarrow p_1$ y $p_2^{'} \leftarrow p_2$;
  **repeat**
    Realizar cruza en $p_1^{'}$ y $p_2^{'}$ para generar $n_{off}$ descendientes $o_1, ..., o_{n_{off}}$;
    Evaluar $o_1, ..., o_{n_{off}}$;
    Encontrar el descendiente con el mejor valor de aptitud, $o_{best}$;
    Reemplaza el peor entre $p_1^{'}$ y $p_2^{'}$ con $o_{best}$;
  **until** $n_t$ *veces*
  retorna $p_1^{'}$ y $p_2^{'}$;
**end**

RCMA-XHC asigna de forma adaptativa diferentes probabilidades para el uso del buscador local, lo que genera un balance entre la búsqueda global y local de acuerdo a las particularidades de cada problema.

Este AM se compone de los siguientes procedimientos:

- **Mecanismo de selección para aplicar cruza.** En un algoritmo genético el operador de cruza intercambia segmentos de cadenas que representan los cromosomas de soluciones que actúan como padres para generar descendientes (intercambio de información cromosómica). Un mecanismo de selección para las soluciones padre determina las condiciones para aplicar dicho operador.

  El mecanismo propuesto en RCMA-XHC es un apareamiento selectivo negativo, el cual se refiere a la reproducción entre individuos con diferente fenotipo. [1] La semejanza o disimilitud entre fenotipos está definida por la distancia euclidiana determinada por los valores de sus cromosomas. La finalidad de este método es incrementar la diversidad genética en la población.

- **Operadores genéticos.** En [50], se utiliza como operadores de cruza y mutación a PBX [91] y BGA [63, 84], respectivamente, ambos descritos en el capítulo 4, en la p.g. 56.

- **Mecanismo adaptativo para aplicar buscador local.** RCMA-XHC presenta un esquema de autoadaptación para asignar un valor de probabilidad (LS) a cada cromosoma $c_{nuevo}$ que se generó por cruza y mutación (ver ecuación (5.1)).

$$pls = \begin{cases} 1 & \text{si } f(c_{nuevo}) \text{ es mejor que } f(c_{peor}) \\ 0.0625 & \text{en otro caso} \end{cases} \tag{5.1}$$

[1] En genética moderna, se denomina *fenotipo* a los rasgos específicos de un individuo. En cómputo evolutivo se denomina fenotipo a la decodificación del cromosoma; es decir, a los valores obtenidos al pasar de una representación (p. ej. binaria) a la usada por la función objetivo.

donde $f$ es la función de aptitud y $c_{peor}$ es la peor solución en la población actual. $c_{nuevo}$ se considera un cromosoma descendiente, el cual es mejor que el peor cromosoma de las soluciones actuales. Este último se somete al procedimiento de búsqueda local, mediante el operador de cruza escalando la colina. De lo contrario se asigna un valor de probabilidad muy bajo ($pls = 0.0625$).

- **Método de cruza escalando la colina XHC.** El objetivo de este operador es obtener el mejor nivel de precisión para guiar a la población hacia las regiones de búsqueda más prometedoras para producir un refinamiento eficiente sobre las soluciones.

  El procedimiento mantiene un par de padres para ejecutar repetidamente la cruza sobre éstos y el ciclo termina hasta que un número de descendientes $n_{off}$ es alcanzado. Después se selecciona al mejor descendiente para reemplazar al peor padre (sólo si es mejor). El proceso itera $n_{it}$ veces y regresa los dos padres actuales finales. Este modelo de cruza XHC requiere valores para $n_{off}$ y $n_{it}$, y un par de padres iniciales $(p_1, p_2)$. Ver algoritmo 12.

- **Estrategia de reemplazo.** La condición de sustitución de soluciones en cada generación, se realiza sólo si el nuevo individuo es mejor que el anterior; a esta condición se le conoce como "estrategia de reemplazo estándar". Se hace notar que el eliminar al peor individuo de la población genera una alta presión de selección.

El algoritmo 13 muestra el procedimiento completo de RCMA-XHC.

El AM RCMA-XHC a pesar de ser adaptativo, requiere la definición de varios parámetros, lo cual implica realizar un análisis sobre el impacto de sus valores en el desempeño del algoritmo.

### 5.1.2. LDSE: Evolución diferencial y método de Nelder-Mead

En [52], se muestra un esquema general hibridizando un algoritmo evolutivo y el método de descenso empinado simplex. Este procedimiento se conoce también como evolución $m$-simplex, donde $m \in 1, 2, ..., n$ y $n$ es el número de soluciones padre. Con este método se producen individuos que sobreviven por la regla de selección natural.

La evolución $m$-simplex mantiene un conjunto de población $\vec{X}(t)$ de $N$ puntos (individuos) $X_i(t), i = 1, 2, ..., N$. El progreso evolutivo conduce esos puntos a la vecindad del mínimo global. El algoritmo se guía mediante el reemplazo de todos los puntos malos en la población con nuevos puntos mejores por generación. Específicamente, para cada individuo $\vec{X}_i(t)(i = 1, 2, ..., N)$ en la población actual, $m+1$ puntos (individuos) son seleccionados aleatoriamente para formar un $m$-simplex. Entre los $m + 1$ puntos se tiene que encontrar el mejor individuo $X_b$ y el peor individuo $X_w$. El $i$-ésimo individuo $X_i(t)$ tiene dos oportunidades de mejorar. La primera oportunidad se otorga por el operador de reflexión. Si el punto reflejado es mejor que $X_i(t)$, entonces $X_i(t)$ es reemplazado por el punto reflejado, por lo tanto se dice que es promovido. En otro caso, una segunda oportunidad se brinda, mediante el operador de contracción con el objetivo de mejorar $X_i(t)$.

**Algoritmo 13**: Algoritmo RCMA-XHC

```
1  Datos: población inicial
2  Resultado: soluciones factibles
3  begin
4      Inicializa población;
5      while no se cumpla el criterio de terminación do
6          Usar la estrategia de apareamiento selectivo negativo (selección de dos
           padres);
7          Aplicar los operadores PBX y BGA para crear un nuevo descendiente O_new;
8          Evaluar O_new;
9          Invocar el mecanismo adaptativo pls para O_new
10         if u(0,1) < pls then
11             Encontrar el mejor cromosoma en la población, c_best;
12             Mejorar mediante método de cruza "escalando la colina" con
               O_new, c_best, n_off, n_it
13             c^1_xhc y c^2_xhc son retornados (c^1_xhc es el mejor);
14             Reemplazar c_best con c^1_xhc, sólo si éste es mejor;
15             Utilizar la estrategia de reemplazo estándar para insertar c^2_xhc en la
               población;
16         else
17             Emplear estrategia de reemplazo estándar para insertar c_new en la
               población;
18 end
```

La reflexión del peor punto $X_w$, considerando el centroide de los otros $m$ puntos $\vec{X}$, debe cumplir con la regla $X_{ref} = \vec{X} + \alpha \cdot (\vec{X} - \vec{X}_w)$, y la contracción debe cumplir con la regla $X_{cont} = \vec{X} + \beta \cdot (X_w - \vec{X})$ tal como en el método Nelder-Mead visto en el capítulo 4. Sin embargo, si el $i$-ésimo individuo $X_i(t)$ pierde las dos oportunidades y no puede recibir beneficios promedio, esto es, su valor de función es peor que el valor promedio de la población actual, éste va a realizar su último esfuerzo para avanzar hacia el mejor individuo $X_b$ o alejarse del peor individuo $X_w$. El último esfuerzo podría degenerar al $i$-ésimo individuo, además de poder incrementar la diversidad de la población. El procedimiento anteriormente descrito se muestra en el algoritmo 14.

Los factores de escalamiento $\alpha$ y $\beta$, usados en este AM, usualmente se encuentran en los intervalos $[0.5, 2]$ y $[-0.5, -0.1] \cup [0.1, 0.5]$ respectivamente. El tamaño de la población $N$ debe ser mucho más grande que la dimensión del problema $n$ y los puntos seleccionados $m + 1$ no son necesariamente diferentes entre sí.

A la evolución $m$-simplex se le conoce como evolución simplex de baja dimensionalidad (LDSE por sus siglas en inglés) ya que la dimensión del simplex es mucho menor que la dimensión del problema, esto es, $m$ es mucho menor que $n$. De forma similar, es llamado evolución simplex de dimensionalidad completa (FDSE) si $m = n$. Sin embargo, en [52]

**Algoritmo 14**: Algoritmo LDSE

**Datos:** Población de tamaño n, límite superior e inferior $lb, ub$, factores de escalamiento $\alpha$ y $\beta$
**Resultado:** Soluciones factibles
**begin**
  Establece $t \leftarrow 0$;
  Inicializa población $\vec{X}(0) \leftarrow \{X_1(0), X_2(0), ..., X_N(0)\}$
  donde $X_i(0) \leftarrow (x_{i,1}(0), x_{i,2}(0), ..., x_{i,n}(0))$;
  **repeat**
    Evaluar $\vec{X}(t)$ y calcular $f(X_i(t))$ para cada individuo;
    Reproduce y actualiza $\vec{X}(t)$;
    Por cada $i \in \{1, 2, ..., N\}$, seleccionar aleatoriamente $m+1$ individuos de la población actual: $X_{r_i}, i \leftarrow 1, 2, ..., m+1$, determinar cuál es el mejor y el peor de los puntos $X_{mejor}, X_{peor}$ y calcular $\vec{X} \leftarrow \frac{1}{m} \sum_{r_i \neq w} X_{r_i}$;
    Dado un punto de reflexión como $X_{ref} \leftarrow \vec{X} + \alpha \cdot (\vec{X} - X_w)$;
    **if** $f(X_{ref}) < f(X_i(t))$ **then**
      $X_i(t+1) \leftarrow X_{ref}$;
    **else**
      Dada $X_{cont} \leftarrow \vec{X} + \beta \cdot (X_w - \vec{X})$; **if** $f(X_{cont}) < f(X_i(t))$ **then**
        $X_i(t+1) \leftarrow X_{cont}$;
      **else**
        **if** $f(X_i(t)) \geqslant \frac{1}{N} \sum_i f(X_i(t))$ **then**
          **if** $f(X_{r_b}(t)) < f(X_i(t))$ **then**
            $X_i(t+1) \leftarrow X_i(t) + 0.618 \cdot (X_b(t) - X_i(t))$,
          **else**
            $X_i(t+1) \leftarrow X_i(t) + 0.382 \cdot (X_i(t) - X_w(t))$,
  **until** *hasta que el criterio de paro sea satisfecho*
**end**

los resultados numéricos muestran que LDSE es superior a FDSE para problemas con alta dimensionalidad.

El objetivo de analizar AMs que usan el método de Nelder y Mead es para marcar las diferencias con la propuesta de tesis y el operador que ésta usa.

### 5.1.3. DEahcSPX: Algoritmo de evolución diferencial, técnica escalando la colina y operador de cruza SPX

Noman e Iba proponen en [66] un AM denominado "DEahcSPX". Este algoritmo combina evolución diferencial [88] con una técnica escalando la colina y el operador de cruza SPX [92] (descrito en el capítulo 4).

Este AM (ver algoritmo 15) se aplica a problemas de optimización de cualquier dimensión. DEahcSPX tiene la particularidad de aplicar primero un proceso iterativo que invoca al buscador local, con la finalidad de refinar la mejor solución de la población actual ($P^G$).

Una vez que no encuentra mejora se emplea el mecanismo de búsqueda global basado en ED en su versión *rand*/1/*bin*, el cual recorre la población actual para generar nuevos descendientes.

**Algoritmo 15**: Algoritmo DEahcSPX

```
1  Datos: población inicial P^G
2  Resultado: soluciones factibles
3  begin
4      Inicializa población P^G ;
5      Evalua P^G ;
6      while no se cumpla con criterio de terminación do
7          B igual El índice de la mejor solución;
8          P^G.[B] ← AHCXLS(P^G, n_p);
9          for cada individuo I en P^G do
10             Reproduce un descendiente J de I a partir de ED;
11             P^{G+1} ← P^{G+1} ∪ y la selección de (I, J);
12         Establece G ← G + 1;
13 end
```

**Algoritmo 16**: Mecanismo escalando la colina AHCXLS

```
1  Datos: Mejor vector solución I y n_p número de padres
2  Resultado: Mejora a I
3  begin
4      Establece P[1] ← I;
5      for i ← 2 a n_p iteraciones do
6          P[i] ← selecciona individuo aleatorio de la población;
7      C ← aplicar Cruza SPX(P) ;
8      if C es mejor a a P[1] then
9          P[1] ← C;
10     else
11         regresa P[1];
12 end
```

El mecanismo escalando la colina AHCXLS (ver algoritmo 16) itera un número indeterminado de veces para aplicar el operador de cruza SPX. La condición de paro de este proceso es cuando no se tiene mejora de la solución base que recibe ($I$).

AHCXLS adapta el buscador local SPX (visto en el capítulo 4), sin agregar parámetros al algoritmo, con la finalidad de encontrar una solución mejor a la que recibe. Mientras que el buscador local SPX crea un $m-$simplex, donde $m$ es 3 o 4 dependiendo de la dimensionalidad del problema.

El operador de cruza SPX recibe las soluciones que genera el $m-$simplex. A partir de estos puntos iniciales se calcula el centroide del simplex. Se establece un factor de expansión o contracción $\epsilon$ para definir el vecindario límite. Posteriormente, se generan $n-1$ escalares aleatorios. Por último, mediante la ecuación del paso 11 en el algoritmo 17 se generan los descendientes.

El análisis completo del comportamiento de este operador se presenta en el capítulo 6 y 7, ya que se usa dicho operador como base para el algoritmo propuesto en este trabajo de tesis.

**Algoritmo 17**: Operador de cruza SPX

1 **Datos:** $n_p$ soluciones padre
2 **Resultado:** Refinar, generando solución descendiente
3 **begin**
4 Elegir $n_p$ soluciones padre de la población $x_i^G,\ i \leftarrow 1, ..., n_p$ de acuerdo al modelo generacional usado y calcular el centro de masa $O$;
5 $O \leftarrow \frac{1}{n}\sum_{i=1}^{n_p} x_i^G$;
6 Generar $r_i$ números aleatorios
7 $r_i \leftarrow u^{\frac{1}{i+1}}, (i \leftarrow 1, ..., n_p - 1)$
8 donde $u$ es un número aleatorio $\in [0, 1]$;
9 Calcular $y_i$ y $C_i$
10 $y_i \leftarrow O + \epsilon(x_i^G - O), (i \leftarrow 1, ..., n_p)$
11
$$C_i \leftarrow \begin{cases} 0, & (i \leftarrow 1) \\ r_{i-1}(y_{i-1} - y_i + C_{i-1}), & (i \leftarrow 2, ..., n_p) \end{cases}$$
donde $\epsilon = 1.0$ es el radio de expansión y un parámetro de control de SPX;
12 Generar un descendiente $C$
13 $C \leftarrow y_{n_p} + C_{n_p}$;
14 **end**

A continuación se mencionan algunas de las características que contempla el AM:

- Utiliza 10 funciones de prueba mono-objetivo con dimensión $N = 10, 50, 100$ y 200
- Sugieren una población de entre $5N$ a $10N$
- El algoritmo de ED utiliza los valores de $F = 0.9$ *y* $Cr = 0.9$
- Se sugiere usar el operador de búsqueda local SPX con un número de padres igual a 3, o 4 para problemas de alta dimensionalidad.

Entre los estudios que se realizan en [66], se tienen: la sensibilidad al tamaño de la población en el algoritmo y la escalabilidad de los problemas que resuelve. Además, se presenta una comparación entre DEahcSPX y diversos esquemas competitivos de la literatura especializada.

## 5.2. AMs para optimización mono-objetivo con restricciones

Las funciones de restricción, en los problemas de optimización, redefinen el espacio de búsqueda factible, y puede modificarse la ubicación de la solución óptima global. Una solución a esta limitante es utilizar mecanismos de manejo de restricciones, los cuales establecen un equilibrio entre la optimización de la función objetivo y la minimización de la violación de las restricciones, como lo muestran los métodos vistos en el capítulo 3.

En este apartado, se analizan AMs que resuelven problemas mono-objetivo con restricciones, mientras que en el capítulo 7 se realiza una comparación entre estos métodos y la propuesta de esta tesis, con la finalidad de verificar qué tan competitivo y superior es el AM propuesto al resolver un conjunto de funciones, el cual contempla diversos problemas en espacios de búsqueda restringidos (ver apéndice A).

### 5.2.1. MCODE cooperativo: AM con ED y cooperación co-evolutiva.

La metodología llamada cooperación co-evolutiva fue propuesta por primera vez por Potter y DeJong en [68] y se inspira en la relación cooperativa de las especies en la naturaleza. Shi [43] extiende este trabajo y propone ED con cooperación co-evolutiva, donde una combinación de individuos desde cada subpoblación sigue una solución completa del sistema. El método asigna un valor de aptitud a todo el sistema, cuyo valor varía de acuerdo a cada subpoblación.

Cuando existe una alta interdependencia ente las variables de una función a optimizarse, el comportamiento de un algoritmo genético co-evolutivo no es siempre satisfactorio. Debido a esto, Potter y De Jong constituyeron una combinación aleatoria entre los individuos a partir de evaluar la aptitud de todas las especies. Esta estrategia resuelve parcialmente el problema, sin embargo cuando se tiene un número considerable de variables todo se complica.

Los trabajos previos de cooperación evolutiva habían sido empleados únicamente en problemas de optimización mono-objetivo sin restricciones, por lo que Liu et al. en [29] proponen un AM para problemas con restricciones, llamado MCODE. Este algoritmo trabaja con dos poblaciones cooperativas, las cuales son construidas para evolucionar de forma independiente mediante el algoritmo de ED en su versión *best*/1/*bin* (ver capítulo 3). La primera población tiene como propósito minimizar la función objetivo sin considerar las funciones de restricción y la segunda población se encarga de minimizar la violación de las funciones de restricción, sin considerar la función objetivo.

En este algoritmo la población total es dividida de acuerdo a la función objetivo y la violación de las restricciones. Por lo tanto, e incluso cuando se tienen miles de variables de decisión, el número de subgrupos es dos.

Además, MCODE cuenta con un mecanismo de interacción y migración entre las dos poblaciones. Por un lado, las poblaciones evolucionan de forma separada y después de varias iteraciones migran las soluciones factibles del primer grupo al segundo grupo de las soluciones infactibles.

Las dos poblaciones evolucionan de acuerdo a una función de aptitud asignada a ciertas generaciones, el sistema los combina y después divide nuevamente en dos grupos para reanudar otra evolución, lo cual crea un ciclo.

Incorporar un buscador local en un algoritmo puede mejorar los resultados notablemente. Por esta razón MCODE utiliza un operador de mutación gaussiana, el cual se aplica si la mejor solución no tiene cambio después de varias iteraciones, es decir, cuando el mejor individuo de la población no evoluciona después de varias generaciones.

La mutación gaussiana consiste en añadir a cada elemento de un vector solución, un valor aleatorio que se encuentre dentro de una distribución gaussiana para crear un nuevo vector descendiente.

En algunas ocasiones una solución infactible se encuentra mucho más cerca del valor óptimo que otras soluciones ubicadas en el dominio de los factibles. Por lo tanto, se ocupa un umbral que calcule los límites de estos dos grupos. Esto se logra estableciendo $N$ como el número de restricciones y $\epsilon_i$ como la violación de cada restricción.

Si se cumple que $\sum_{i=1}^{N} |\epsilon_i| \leqslant \sigma$, entonces el individuo pertenece al primer grupo; de lo contrario, pertenece al segundo. Para lograr que la optimización global satisfaga las restricciones $\sigma$ debe decrementarse de acuerdo al proceso de evolución o al radio de soluciones factibles.

El diagrama de flujo mostrado en la figura 5.1 y el algoritmo 18 muestran el funcionamiento de la propuesta.

Existen dos operaciones importantes en MCODE cooperativo. En primer lugar es necesario establecer la mejor relación, no sólo en la función de minimización de la función objetivo, sino también al minimizar la violación de las funciones de restricción. Por otro lado, si el proceso de ED convierte a un individuo factible en uno infactible, la operación correspondiente se cancela.

La ventaja de MCODE cooperativo es que no requiere establecer valores de coeficientes de penalización.

**Algoritmo 18**: Algoritmo MCODE cooperativo

**Datos:** Población inicial
**Resultado:** Solución óptima
**begin**
  Inicializa población y parámetro $\sigma$;
  **while** *No se cumpla el criterio de convergencia* **do**
    **if** *si no existen soluciones factibles e infactibles* **then**
      Regresar al paso 4
    **else**
      Dividir la población en dos subconjuntos $A$ *y* $B$ (factibles e infactibles);
    **while** *La mejor solución no se estanque* **do**
      Aplicar ED a la población $A$ *y* $B$ por un número determinado de generaciones;
      Combinar las poblaciones generadas de $A$ *y* $B$, hacer interacción y migración;
      **if** *las soluciones aun no convergen* **then**
        actualizar las poblaciones $A,\ B$ *y* $\sigma$;
        El ajuste de $\sigma$ conforma la siguiente regla: si el radio de las soluciones factibles excede un valor predeterminado $\epsilon_1, (N_{factible}/Ntotal \geqslant \epsilon_1)$ entonces $\sigma \leftarrow \sigma \times c_1$, si el radio de las soluciones infactibles excede un determinado valor $\epsilon_2, (N_{infalible}/N_{total} \geqslant \epsilon_2)$ entonces $\sigma \leftarrow \sigma \times c_2$, de lo contrario $\sigma \leftarrow \sigma$. En el proceso anterior,
        $c_1 \geqslant 1\ \ y\ \ c_2 < 1\ \ y\ \ c_1 \in (1,2), c_2 \in (0,1)$ son recomendados.
      **else**
        romper ciclo;
    Aplicar mutación gaussiana a los individuos. Actualizar el mejor valor;
  Regresar la mejor solución del conjunto.
**end**

### 5.2.2. LEDE: Exploración local basada en evolución diferencial.

La propuesta conocida como LEDE de Ali y Kajee-Bagdadi [1] es un algoritmo que combina una modificación de ED e incorpora una técnica de exploración local periódica. Dicha técnica es una versión simplificada del método de búsqueda de patrones (BP). Funciones de penalización, tales como la superioridad de puntos factibles (SFP) [69] y la penalización de parámetro libre (PFP) [15] son utilizadas para el manejo de restricciones en el AM.

Este algoritmo modifica el operador de mutación de ED de la siguiente forma:

$$\hat{x}_{i,k} = x_{b,k} + F_k(x_{\beta,k} - x_{\alpha,k}) \tag{5.2}$$

donde $x_{b,k}$ es el mejor punto en la población, $x_{\beta,k}$ y $x_{\alpha,k}$ son puntos uniformemente seleccionados de la población.

En la ecuación (5.2), $F_k$ es un parámetro de escalamiento dependiente de la iteración del algoritmo, es decir se le asignan valores de acuerdo a $F = Unif\{[-1, -0.4] \cup [0.4, 1.0]\}$. En cada $k-$ésima iteración un número aleatorio $r_k \in (0, 1)$ es generado. Si $r_k \leqslant 0.5$ entonces $F_k$ se establece de manera uniforme en el intervalo $[0.4, 1]$. De lo contrario, $F_k$ se establece en el intervalo $[-1, -0.4]$. Si el punto mutado no se encuentra dentro del intervalo de los valores de las variables ($\hat{x}_{i,k} \notin [l, u]$) entonces se usa un valor diferente para $F_k$.

LEDE emplea dos esquemas para el manejo de restricciones SFP [69] y PFP [15]. Ambos evalúan la aptitud de la población dependiendo de la iteración del algoritmo.

**El esquema SFP** define como función de penalización la siguiente ecuación:

$$\hat{f}(y_{i,k}) = f(y_{i,k}) + R \times G(y_{i,k}) + \Theta_k(y_{i,k}), \quad y_{i,k} \in T_k \tag{5.3}$$

donde:

$\hat{f}(y_{i,k})$ es el valor de aptitud de cada individuo.

$f(y_{i,k})$ el valor de la función objetivo evaluada en un determinado punto $y_{i,k}$.
Es decir el $i-$ésimo elemento de la población en la $k-$ésima iteración.

$R$ es un parámetro de penalización

$G(y_{i,k}) = \sum_{j=1}^{m} max[0, g_j(x)]^q$, se refiere a la suma de la evaluación de las funciones de restricción en puntos infactibles. El algoritmo define q = 1;

Y el término $\Theta_k(Y_{i,k})$ se define mediante la ecuación (5.4)

$$\Theta_k(Y_{i,k}) = \begin{cases} 0 & \text{si } T_k \cap \Omega = \varnothing \text{ o } y_{i,k} \in \Omega \\ \alpha & \text{si } T_k \cap \Omega \neq \varnothing \text{ y } y_{i,k} \notin \Omega \end{cases} \tag{5.4}$$

en donde $\Omega$ es el conjunto de los puntos factibles.

$T_k$ el conjunto total de las soluciones de la población actual.

El valor de $\alpha$ se calcula de acuerdo a:

$$\alpha = max\left[0, \max_{y \in T_k \cap \Omega} \in T_k f(y) - min_{z \in T_k \ (T_k \cap \Omega)} \ [f(z) + R \times G(z)]\right] \tag{5.5}$$

El término $\Theta_k$ es usado para penalizar los puntos infactibles sólo cuando la población $T_k$ contiene puntos factibles. Por lo tanto, cuando la población contiene puntos factibles, serán tan malos como los puntos infactibles.

**El esquema PFP** es una modificación [60] del esquema PFP. La función de penalización se define de acuerdo a la siguiente ecuación:

$$\hat{f}(y_{i,k}) = f(y_{i,k}) + G(y_{i,k}) + \Theta_k(y_{i,k}), \quad y_{i,k} \in T_k \tag{5.6}$$

donde

$$\Theta_k(y_{i,k}) = \begin{cases} 0 & \text{si } y_{i,k} \in \Omega, \\ -f(y_{i,k}) & \text{si } T_k \cap \Omega = \varnothing, \\ -f(y_{i,k}) + max_{y \in T_k \cup \Omega} \ f(y) & \text{si } T_k \cap \Omega \neq \varnothing \text{ y } y_{i,k} \notin \Omega. \end{cases} \tag{5.7}$$

Cuando la población contiene puntos factibles e infactibles se asegura que ambos conjuntos de puntos sean tan malos entre ellos. Esto se logra añadiendo el valor de la función objetivo del peor punto factible y la violación de las restricciones de cada punto infactible.

Entonces el cálculo de la aptitud con un esquema PFP está dado por:

$$\hat{f}(y) = G(y) + max_{x \in T_k \cup \Omega} \ f(x) \tag{5.8}$$

El cálculo de la aptitud de PFP es sencillo si todos los puntos son factibles. De lo contrario, se asocia una regla de selección en ED, dada por la siguiente ecuación.

$$\Theta_k(x_{i,k+1}) = \begin{cases} y_{i,k} & \text{si } \hat{f}(y_{i,k}) < \hat{f}(x_{i,k}), \\ x_{i,k} & \text{en otro caso} \end{cases} \tag{5.9}$$

donde $x_{i,k}$ y $y_{i,k}$ son $S_k$ (población que usa ED) y $T_k$ (población que minimiza las restricciones), respectivamente.

La exploración local en LEDE es una modificación al método de BP [45] para optimización local. A éste se le nombra patrón local de búsqueda (LPS por sus siglas en inglés) y es invocado al final de la $k-$ésima iteración del mecanismo global.

LPS explora la vecindad del mejor punto $x_{(k)}$ en la población, $S_k$ de acuerdo a los pasos descritos en el algoritmo 19.

El algoritmo 20 muestra el funcionamiento completo de LEDE. $S_k$ es la población inicial, $f(x_{i,k})$ la función objetivo, $G(x_{i,k})$ las funciones de restricción de cada punto. La función de penalización $\Theta_k(x_{i,k})$, la evaluación de la aptitud es $\hat{f}(x_{i,k})$ para cada punto. La siguiente generación $T_k$ es generada por la mutación y cruza. Los puntos de $S_k$ y $T_k$ son comparados punto a punto para generar $S_{k+1}$. Este proceso de comparación invoca al buscador local LPS.

**Algoritmo 19**: Algoritmo de búsqueda local basado en una modificación de búsqueda de patrones

**begin**
  Seleccionar aleatoriamente 5 puntos distintos en la población, el 10 % de $S_k$;
  Calcula el promedio de la distancia, $AvgDis$, entre los $m$ puntos y el conjunto $\Delta_k \leftarrow 0.1 \times AvgDis$
  Establecer el centro $x_k \leftarrow x_{r,k}$ donde $x_{r,k}$ es seleccionado uniformemente de los 5 puntos.
  **for** $i \leftarrow 1, 2, ..., 2n$ **do**
    calcular $y^i \leftarrow x_k + \Delta_k d_i$
  **if** *Un punto* $y^i \in P_k$ *es* $\hat{f}(y^i) < \hat{f}(x_{(k)})$ **then**
    reemplaza $x_{r,k}$ en $S_k$ con $y^i$, es decir $x_{r,k} \leftarrow y^i$ y terminar LPS
  **if** $\hat{f}(y^i) \geqslant \hat{f}(x_{(k)})$ *para toda* $y^i \in P_k$ **then**
    conservar $x_{r,k} \in S_k$ y terminar LPS
**end**

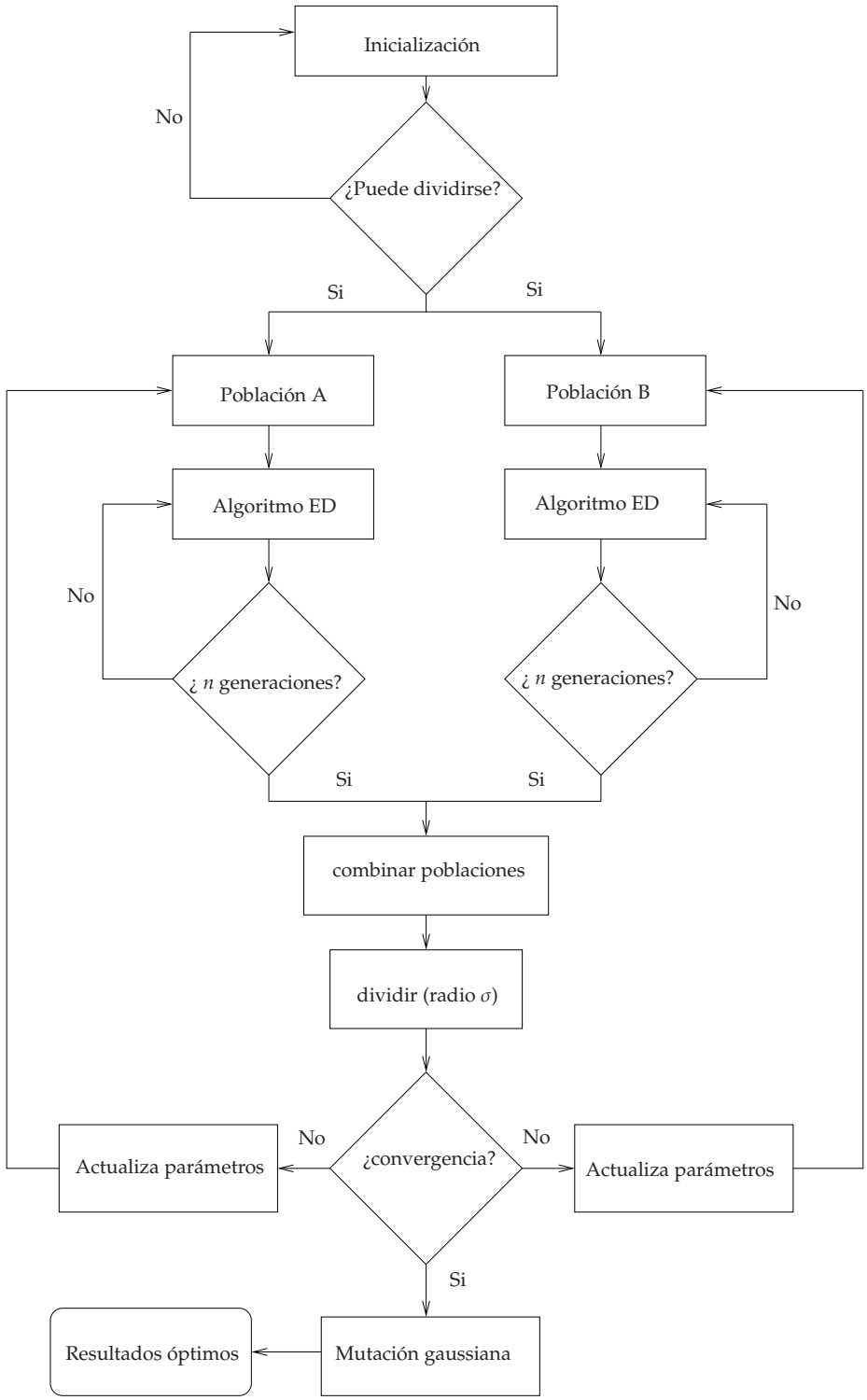



Figura 5.1: Diagrama de flujo del algoritmo MCODE

**Algoritmo 20**: Algoritmo LEDE

**Datos:** Población inicial y parámetros $N$, $C_r$
**Resultado:** Solución óptima
**begin**
  Inicializa $N$ y $C_r$ ($R$ si se utiliza SFP) y la población inicial como $S_k \leftarrow \{x_{1,k}, x_{2,k}, ..., x_{N,k}\}$, $k \leftarrow 1$;
  Para toda $i$ calcula $f(x_{i,k})$ y $G(x_{i,k})$;
  $k \leftarrow 1$;
  Calcula $\Theta_k(x_{i,k})$ y $\hat{f}(x_{i,k})$, usando la ecuación (5.3) y (5.4) para SFP o (5.6) y (5.7) para PFP
  **while** *No se conozca el criterio de paro* **do**
    **if** *No se conoce el criterio de paro* **then**
      Generar un punto de prueba $y_{i,k} \in T_k$, generado por:
      **Mutación** usando la ecuación $\hat{x}_{i,k} \leftarrow x_{b,k} + F_k(x_{\beta,k} - x_{\alpha,k})$
      **Cruza** usando la ecuación

$$(Y^j_{i,k}) \leftarrow \begin{cases} \hat{x}^j_{i,k} & \text{si } r^j \leqslant c_r o j \leftarrow I_i, \\ x^j_{i,k} & \text{si } r^j > c_r y j \neq I_i, \end{cases}$$

      donde $I_i$ es un entero aleatorio en el conjunto $I_i \in I \leftarrow \{1, 2, ..., n\}$, la potencia $j$ representa el $j-$ésimo elemento en de los puntos y $r^j \in (0, 1)$ es una aleatorio elegido para cada j. Evaluar $f(y_{i,k})$ y $G(y_{i,k})$;
    Calcula $\Theta_k(x_{i,k})$ y $\hat{f}(x_{i,k})$, usando las ecuaciones (5.3) y (5.4) para SFP o (5.6) y (5.7) para PFP
    Actualizar $S_k$, es decir obtener $S_{k+1}$ usando la regla de selección (5.9)
    **if** $k \; mod \; k_t \leftarrow 0$ **then**
      usar buscador local LPS para actualizar $S_{k+1}$ si se requiere;
    Hacer $k \leftarrow k + 1$
**end**

Capítulo 6

# Propuesta de algoritmo memético para problemas con restricciones

La propuesta de Algoritmo Memético para Espacios Restringidos (AMER), contempla cuatro aspectos principales, los cuales se describen en las secciones de este capítulo. La sección 6.1 explica una técnica de muestreo para generación de puntos uniformemente distribuidos en un espacio de búsqueda $n-$dimensional. Posteriormente, se describen los mecanismos de búsqueda global y búsqueda local en las secciones 2 y 3, respectivamente. En la sección 6.4 se detalla la adaptación de un algoritmo para el manejo de las restricciones y, finalmente se muestra la implementación completa de AMER en la sección 6.5.

## 6.1. Muestreo de puntos con distribución uniforme

Existen diversas técnicas de muestreo para la generación de vectores solución en un espacio $n-$dimensional. Una forma sencilla para la generación de estos vectores es el muestreo aleatorio. Desafortunadamente, éste presenta ruido [9] y las soluciones no se distribuyen de manera uniforme por todo el espacio de búsqueda. Otra alternativa son las técnicas de muestreo con distribución uniforme, las cuales definen puntos equidistantes en un espacio definido por la dimensionalidad del problema.

Un muestreo uniformemente distribuido responde a la pregunta sobre cómo variar una variable $Y$ que depende de otras $x_1, x_2, \ldots, x_k$.

La propuesta realizada en esta tesis utiliza el muestreo por hipercubos latinos (LHS, por sus siglas en inglés). Esta técnica fue desarrollada por Mc. Kay et al. en 1979 [53].

Presuponiendo un problema $k$ dimensional, donde se requiere $n$ diferentes puntos, entonces LHS sigue los pasos que a continuación se describen:

1. El rango de cada variable es dividido en $n$ subintervalos no sobrepuestos, es decir que no coincidan, los cuales tendrán la misma probabilidad.
2. Se genera aleatoriamente un valor dentro de cada subintervalo (esto respecto a la probabilidad de densidad de cada subintervalo).
3. Los $n$ valores obtenidos para $x_1$ forman parejas con los $n$ valores de $x_2$ de manera aleatoria.
4. Estas $n$ parejas son combinadas de manera aleatoria con los $n$ valores de $x_3$ para formar $n$ tripletas.

5. Esto se repite hasta formar $n$ vectores $k-$dimensionales.

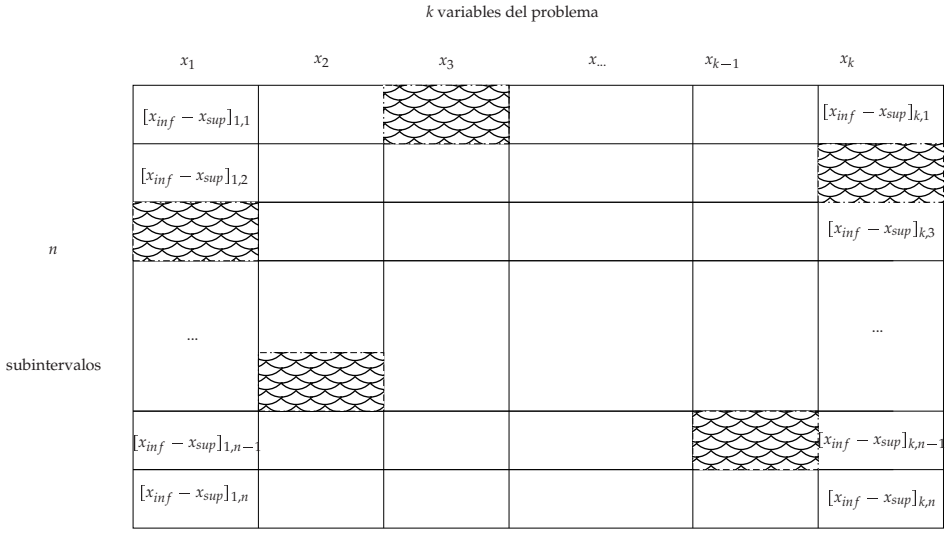



Figura 6.1: Un ejemplo de muestreo por hipercubos latinos

La figura 6.1 muestra el funcionamiento del muestreo LHS para un problema con $k$ variables, donde se requiere la generación de $n$ vectores. En este caso, se divide el intervalo de cada variable en $n$ subintervalos y aleatoriamente se van combinando (sin repetición) valores aleatorios dentro del rango de cada subintervalo. Los recuadros sombreados de la figura representan posibles valores para la generación de un vector. En la figura 6.2 puede verse un ejemplo de muestreo de 100 puntos en un espacio 3-dimensional

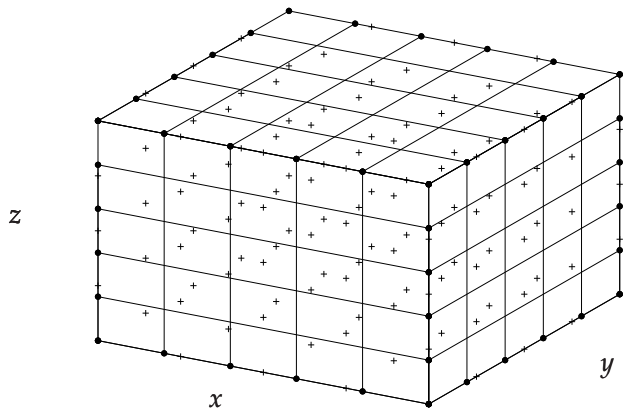



Figura 6.2: Caso hipotético de un muestreo por hipercubos latinos en un espacio 3-dimensional

## 6.2. Evolución diferencial como mecanismo de búsqueda global

El algoritmo de ED es uno de los mejores métodos heurísticos de optimización global que se conocen. Además de ser muy efectivo, suele converger rápidamente al óptimo del problema y, por lo general, su comportamiento es robusto. Adicionalmente, requiere pocos parámetros de control para su funcionamiento y su implementación es sencilla.

Como se estudió en el capítulo 3, en la p.g. 29, existen diversas versiones de ED. Dichas versiones son analizadas en un estudio comparativo presentado en [94] donde se concluye que el modelo **DE/rand/1/bin** muestra tener buen rendimiento con un menor número de evaluaciones de la función objetivo. Por lo anterior, se decidió trabajar con esta versión para nuestra propuesta. El algoritmo 3 en la sección 3.1.3, explica el funcionamiento completo de la versión **DE/rand/1/bin** de ED.

ED como mecanismo de búsqueda global en nuestra propuesta explora todo el espacio de búsqueda definido por la función objetivo, lo que significa que habrá soluciones factibles e infactibles. Esta característica permite que el algoritmo mantenga diversidad en las soluciones, además de considerar aquellos vectores que se encuentren en los límites de la región factible, como se observa en la figura 6.3.

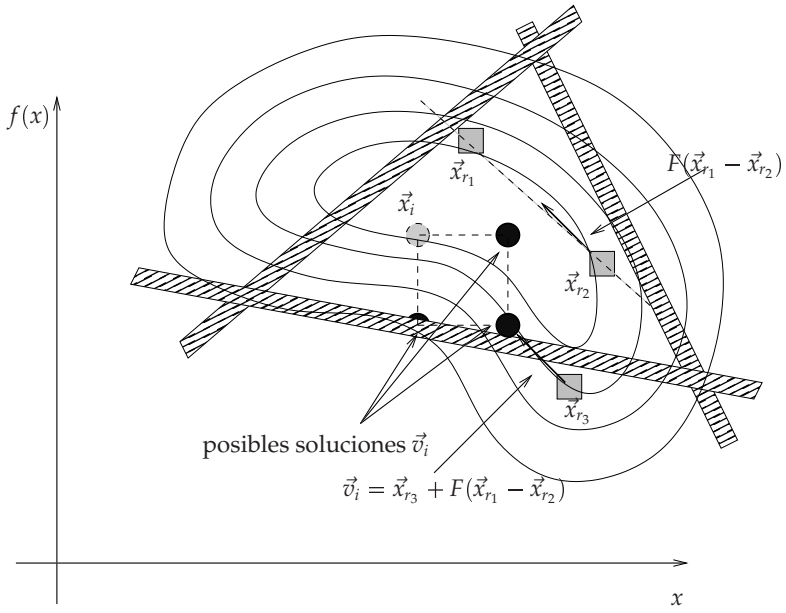



Figura 6.3: Ejemplo del funcionamiento del algoritmo de ED en un espacio restringido

**Estrategia de reemplazo para ED y SPX**

La única modificación realizada al algoritmo de ED del algoritmo 3 es en el operador de selección, ya que al tratarse de problemas en espacios restringidos se considera la comparación entre dos vectores, de la población anterior y la actual, de la siguiente manera:

1. Si las dos soluciones son infactibles se tiene preferencia por aquella que viole en

menor grado la función de penalización.

2. Si las dos tienen igual valor en la función de penalización se escoge aquella que tenga menor valor en la función objetivo.

3. Si las dos soluciones son factibles se elige aquella con menor valor en la función objetivo.

Esta comparación es también usada como mecanismo de reemplazo en el buscador local.

El algoritmo 21 muestra la modificación realizada a ED para el manejo de las restricciones.

## 6.3. Jerarquías estocásticas para el manejo de restricciones

El algoritmo de jerarquías estocásticas estudiado en la sección 3.2.2 es uno de los métodos de manejo de restricciones más competitivos en la literatura y de ahí que se haya decidido incorporar en nuestra propuesta. El algoritmo de jerarquías estocáticas se emplea tal cual como se presentó en el algoritmo 4 de esta tesis (ver sección 3.2.2.3).

El esquema de la figura 6.4 muestra un espacio con posibles soluciones representadas por $x_1, x_2, ..., x_{11}$. Estas soluciones son tanto factibles como infactibles, y en el proceso de jerarquización estocástica se ordenan estas soluciones efectuándose un balance estocástico entre la función objetivo y la función de penalización. Puede observarse en la lista ordenada que la mejor solución es el primer elemento $x_{10}$ ya que se trata de una solución factible, además de estar más cerca del mínimo global del problema. En contraste, $x_9$ es la peor solución, al ser infactible y estar más alejada de la solución óptima global.

A partir de este ordenamiento se utilizaron los primeros y últimos 3 elementos de la lista que serán los padres utilizados por el operador SPX. Esto con la finalidad de refinar a la mejor solución y a la peor solución encontradas en cada iteración del algoritmo.

## 6.4. Operador de cruza simplex

El operador de cruza simplex estudiado en el capítulo 4.4.3 mostró tener características que aceleran, en gran medida, la convergencia de la ED en problemas sin restricciones, por lo que la propuesta introducida en esta tesis adapta dicho operador para problemas con restricciones.

Retomando los pasos que usa este operador de cruza, se explicará a detalle cada uno de ellos y la adaptación a la propuesta.

**Paso 1.** Seleccionar $n_p$ vectores padre de la población $\vec{x}_i^G$, donde $i = 1, ..., n_p$

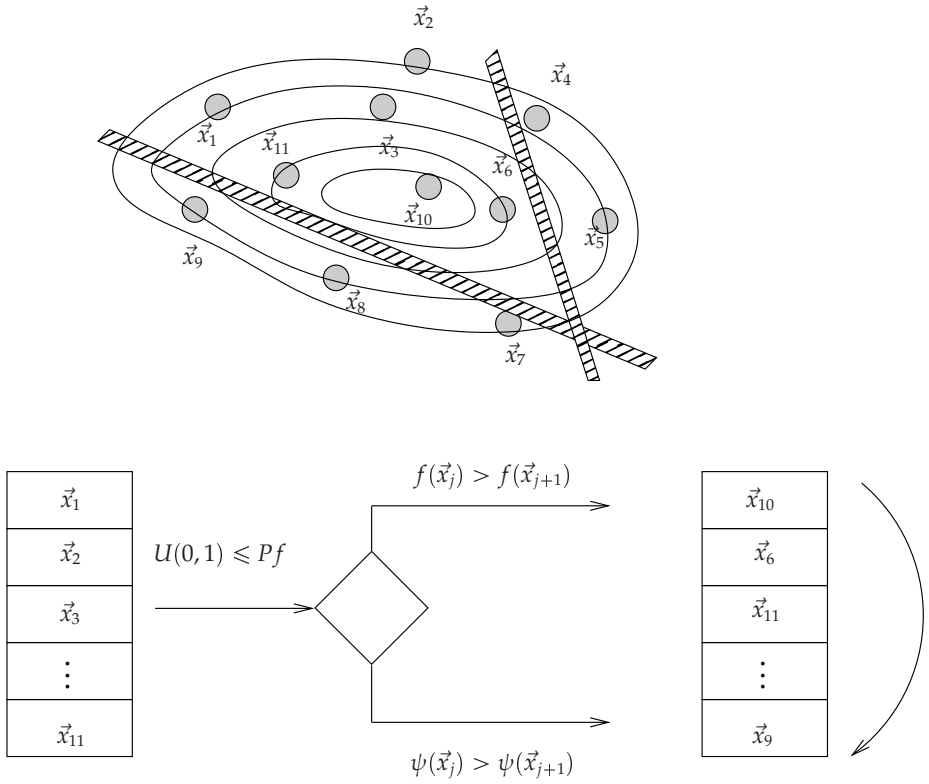
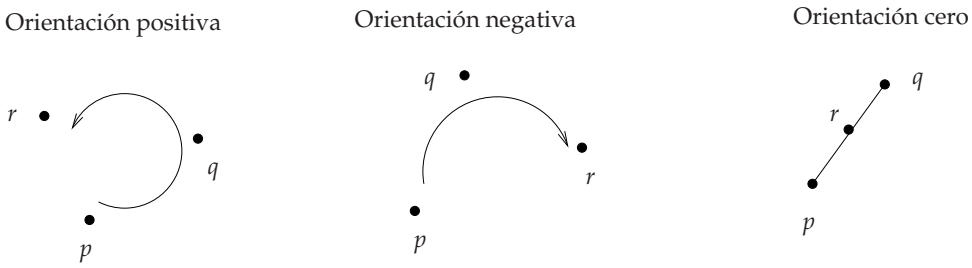



Figura 6.4: Adaptación del mecanismo de manejo de restricciones

Los vectores seleccionados definen el vecindario de búsqueda mediante un $k-$simplex. En AMER se fijó $n_p$ con un valor igual a 3, lo que significa que se trata de un 2-simplex o triángulo en un espacio $n-$dimensional. Para verificar que se forma un simplex, es decir que los puntos son no colineales, debe calcularse el determinante de estos 3 puntos [5], de acuerdo a lo siguiente:

- Existe una relación, llamada orientación, entre $(d+1)-$tuplas ordenadas de puntos en un espacio $n-$dimensional. La orientación extiende la noción de relaciones binarias en un espacio 1-dimensional.
- Dada una tripleta ordenada de puntos $< p; q; r >$ en el plano, decimos que tienen una orientación positiva si definen un triángulo en sentido contrario a las manecillas del reloj.
- Tienen una orientación negativa si forman un triángulo en sentido de las manecillas del reloj.
- Se tiene una orientación cero si son colineales (incluyendo el caso donde dos o más puntos son idénticos). La orientación depende del orden en que los puntos son dados.
- En general, en un espacio $d-$dimensional el área de un simplex (un simplex o $d-$simplex es el análogo en $d$ dimensiones de un triángulo) acotado por $d+1$ puntos puede obtenerse tomando este determinante y dividiéndolo por $d!$




$$orient(p,q,r) = det \begin{pmatrix} 1 & p_x & p_y \\ 1 & q_x & q_y \\ 1 & r_x & r_y \end{pmatrix}$$

Figura 6.5: Casos de orientación para puntos en un espacio 2-dimensional

- Dada la capacidad de calcular el área de cualquier triángulo (o simplex en dimensiones más grandes), es posible calcular el volumen de cualquier polígono (o poliedro).

Para el caso específico de nuestra propuesta únicamente se requiere identificar si el determinante de los puntos es diferente de 0. (ver figura 6.5)

**Paso 2.** Calcular el centro de masa $\vec{O}$ del $k-$simplex

Ya que $n_p$ es igual a 3 el centro de masa del 2-simplex queda definido por la ecuación (6.3)

$$\vec{O} = \frac{1}{3}\sum_{i=1}^{3} \vec{x_i^G} \tag{6.3}$$

donde $\vec{x_i}$ corresponde a cada vector solución de la población $G$.

**Paso 3.** Aplicar factor de expansión $\epsilon$

El factor de expansión en [66] es igual a 1, lo que indica que no se expande ni contrae el $k-$simplex. La diferencia con AMER es que este factor varía de acuerdo al número de iteración, con el objetivo de expandir o contraer el 2-simplex. En las primeras iteraciones este factor en el algoritmo tiene un valor $\epsilon = 1.5$ y en las últimas iteraciones (20 % del total de las generaciones) es $\epsilon = 0.75$. Lo anterior se debe a que el algoritmo explora sobre un espacio de búsqueda más pequeño que al inicio del proceso.

**Paso 4.** Generar $r_i$ números aleatorios de acuerdo a la ecuación (6.4):

$$r_i = u^{\frac{1}{i+1}}, (i = 1, ...., n_{p-1}), \tag{6.4}$$

donde $u \in [0, 1]$ es un número aleatorio en el rango de 0 a 1.

En el caso de la propuesta de esta tesis, estos factores son $r_1 = u^{\frac{1}{2}}$ y $r_2 = u^{\frac{1}{3}}$.

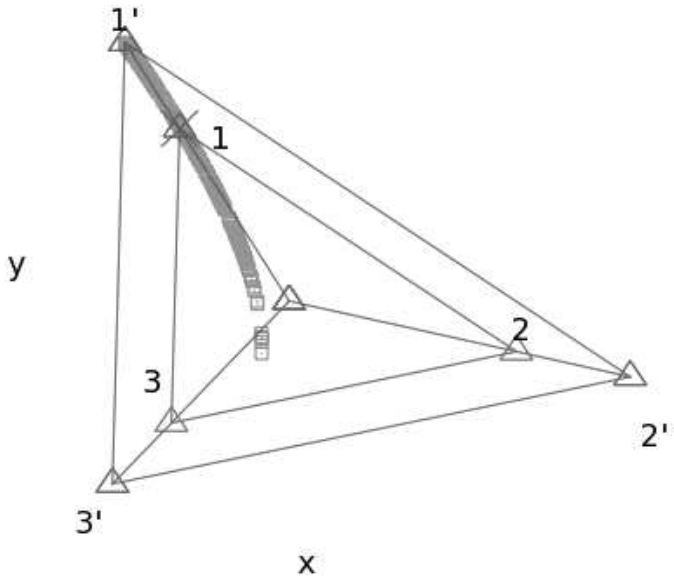



Figura 6.6: Generación de puntos mediante el operador SPX de búsqueda local en un espacio bidimensional

**Paso 5.** Calcular los valores para $y_i$ y $C_i$ de acuerdo a las ecuaciones (6.5) y (6.6).

$$y_i = O + \epsilon(x_i^G - O), (i = 1, ..., 3) \tag{6.5}$$

$$\vec{C_i} = \begin{cases} 0, & (i = 1) \\ \vec{r}_{i-1}(\vec{y}_{i-1} - \vec{y}_i + \vec{C}_{i-1}), & (i = 2, 3) \end{cases} \tag{6.6}$$

**Paso 6.** Generar descendiente.

Finalmente, se genera el vector descendiente $\vec{C}$ a partir de la ecuación (6.7). El reemplazo de la nueva solución se realiza sólo si se cumple con el mecanismo de la estrategia de reemplazo definida anteriormente.

$$\vec{C} = \vec{y}_3 + \vec{C}_3 \tag{6.7}$$

En AMER este operador se ejecuta 2 veces en cada generación.

En las figuras 6.6 y 6.7 puede visualizarse el comportamiento del operador SPX, ya que se generan puntos dentro de un vecindario definido por el 2-simplex en un espacio bidimensional y tridimensional, respectivamente.

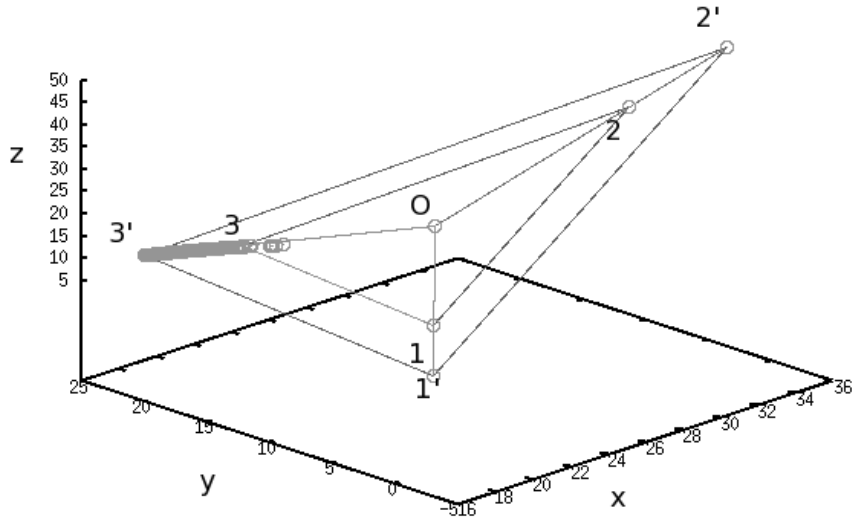



Figura 6.7: Geneación de puntos mediante el operador SPX de búsqueda local en un espacio tridimensional

## 6.5. Algoritmo propuesto

El criterio de terminación establece que el algoritmo memético ha convergido en la generación G, siempre y cuando, la desviación estándar de la función objetivo sea menor que un valor suficientemente pequeño $\sigma$, como sigue:

$$\sum_{i=1}^{tam_{pob}} (f(\vec{x_i}) - \vec{f})^2 \leqslant \sigma \tag{6.8}$$

En la figura 6.8 y en el algoritmo 22 se muestra el proceso completo del algoritmo memético que aquí se propone.

**Algoritmo 21**: Modificación al algoritmo de ED

**Datos:** $D,\ G_{max},\ NP \geqslant 4, F \in (0,1+),\ CR \in [0,1]$, y los límites iniciales $\vec{x}^{(lo)},\ \vec{x}^{(hi)}$
**Resultado:** Solución óptima
**begin**
  Inicializar:

$$\left\{\begin{array}{ll} \forall_i \leqslant NP \vee \forall_j \leqslant D: & x_{j,i,G\leftarrow 0} \leftarrow x_j^{(lo)} + rand_j[0,1] \cdot (x_j^{(hi)} - x_j^{(lo)}) \\ i \leftarrow \{1,2,...,NP\}, & j \leftarrow \{1,2,...,D\}, G \leftarrow 0, rand_j[0,1] \in [0,1] \end{array}\right\} \tag{6.1}$$

  **while** $G < G_{max}$ **do**

$$\forall_i \leqslant NP \left\{\begin{array}{ll} \text{Mutación y recombinación:} & \\ \text{seleccionar aleatoriamente} & r_1, r_2, r_3 \in \{1,2,....,NP\}, \text{donde}: r_1 \neq r_2 \neq r_3 \neq i \\ j_{rand} \in \{1,2,...,D\} & \text{Seleccionar aleatoriamente uno de cada } i \\ \forall_j \leqslant D, u_{j,i,G+1} \leftarrow & \left\{\begin{array}{l} x_{j,r_3,G} + F \cdot (x_{j,r_1,G} - x_{j,r_2,G}) \\ \text{si } (rand_j[0,1) < CR \vee j \leftarrow j_{rand}) \\ x_{j,i,G} \quad \text{de lo contrario} \end{array}\right. \\ \text{Seleccionar:} & \\ \vec{x}_{i,G+1} \leftarrow & \left\{\begin{array}{l} \text{si ambas soluciones son infactibles y} \\ \phi(\vec{u}_{i,G+1}) < \phi(\vec{x}_{i,G}) \\ \text{entonces} \quad \vec{u}_{i,G+1} \\ \text{si ambas soluciones son infactibles y} \\ \phi(\vec{u}_{i,G+1}) \leftarrow \phi(\vec{x}_{i,G}) \\ \text{entonces} \quad \vec{u}_{i,G+1} \quad \text{si} \quad f(\vec{u}_{i,G+1}) \leqslant f(\vec{x}_{i,G}) \\ \text{si} \quad \vec{u}_{i,G+1} \quad \text{es factible y} \quad \vec{x}_{i,G+1} \quad \text{no lo es} \\ \text{entonces} \quad \vec{u}_{i,G+1} \\ \vec{x}_{i,G} \quad \text{de lo contrario} \end{array}\right. \end{array}\right. \tag{6.2}$$

    $G \leftarrow G + 1;$
**end**

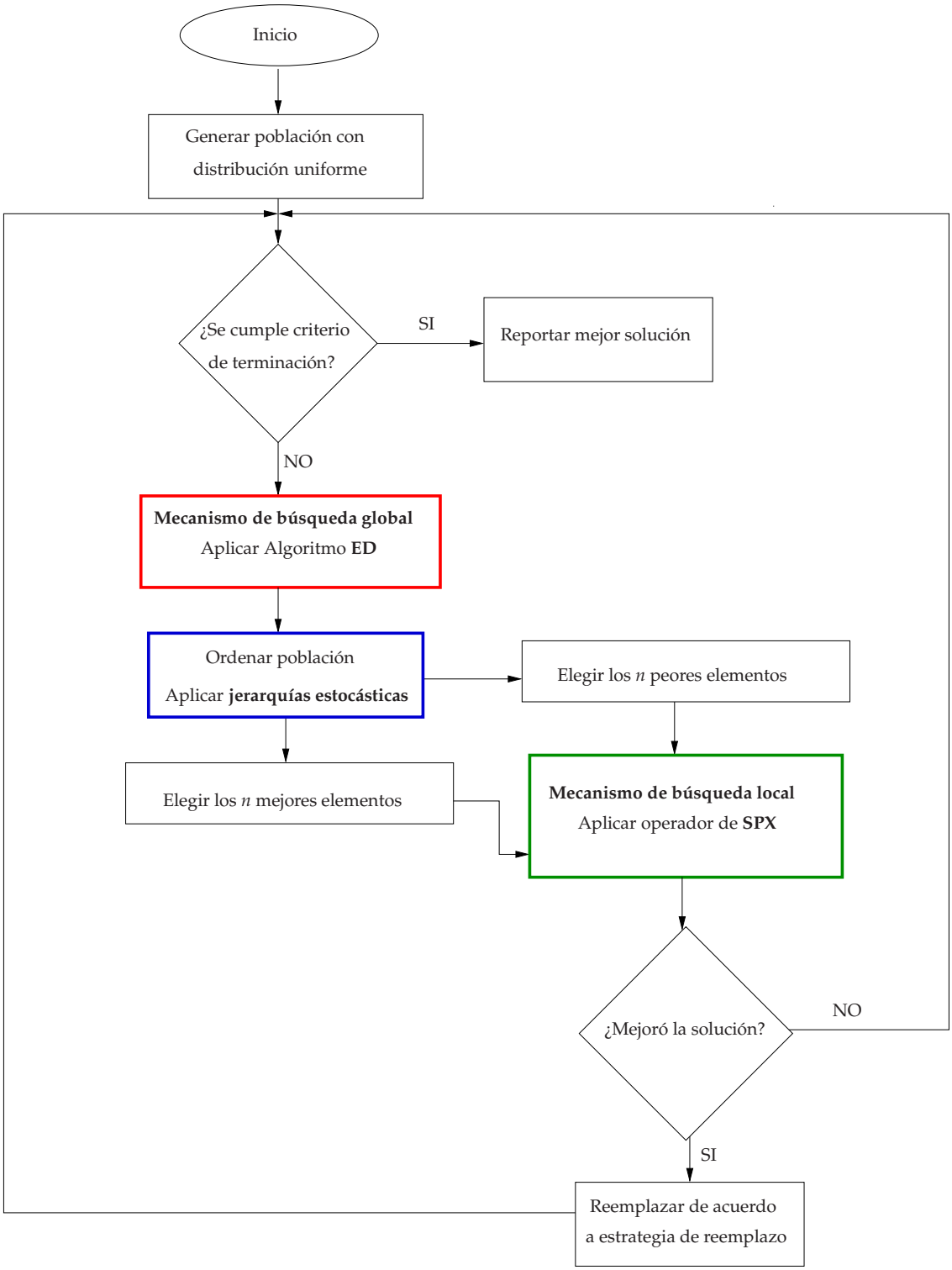



Figura 6.8: Diagrama de Flujo de la propuesta de esta tesis.

**Algoritmo 22**: Propuesta de algoritmo memético

**Datos:** Población Inicial, $Pf$ (parámetro de control para el balance en el algoritmo de jerarquías estocásticas), $Cr$ (parámetro de recombinación de ED), $F$ (parámetro de mutación en ED), $n$ número de soluciones para generar vecindario
**Resultado:** Mejor solución al problema
**begin**
  Generar población con distribución uniforme, mediante LHS ;
  **while** *No se cumpla con criterio de terminación* **do**
    // Procedimiento de evolución diferencial; ▷ **mecanismo de búsqueda global**
    **for** $i \leftarrow \{1, ..., np\}$ **do**
      $r_1, r_2, r_3 \in 1, ..., np$ aleatoriamente seleccionados,
      donde $r_1 \neq r_2 \neq r_3 \neq i$ ;
      $j_{rand} \in \{1, ..., n\}$ aleatoriamente seleccionado;
      **for** $j \leftarrow \{1, ..., n\}$ **do**
        **if** $U_j[0,1] < CR \mid\mid j == j_{rand}$ **then**
          $u_{i,j} \leftarrow x_{r3,j,G} + F(x_{r1,j,G} - x_{r2,j,G})$ ;
        **else**
          $u_{i,j} \leftarrow x_{i,j,G}$;
        **if** $f(\vec{u}_i) \leqslant f(\vec{x}_{i,G})$ **then**
          $\vec{x}_{i,G+1} \leftarrow \vec{u}_i$ ;
      Aplicar mecanismo de selección;
                                ▷ mecanismo para el manejo de las restricciones
    Ordenar la población mediante algoritmo de jerarquización estocásticas;
    // Operador de cruza SPX; ▷ **mecanismo de búsqueda local**
    Asignar $mejor_{sol}[3]$, los 3 mejores elementos del conjunto ordenado;
    Asignar $peor_{sol}[3]$, los 3 peores elementos del conjunto ordenado;
    Aplicar mecanismo de búsqueda local (operador **SPX**) a los elementos de $mejor_{sol}[n]$ y $peor_{sol}[n]$
    $G \leftarrow G + 1$;
**end**
**Criterio de terminación**.

Capítulo 7

# Validación del método propuesto

En este capítulo se describen los experimentos implementados para obtener estadísticas cuyos resultados nos permitieron demostrar la eficacia en el funcionamiento de nuestra propuesta.

En primer lugar, establecimos una metodología para el diseño experimental. Presentamos los resultados y gráficas estadísticas de nuestra propuesta. Comparamos nuestro método contra los algoritmos mencionados en la sección 5.2. Por último, hicimos evidente el efecto que tiene el uso de un buscador local en la solución de problemas en espacios restringidos.

Un mecanismo o algoritmo a estudiar puede ser representado por el modelo de la figura 7.1, donde se visualiza al proceso como una combinación de variables $\vec{x}$, $\vec{z}$ y una salida $y$ que tiene una o más respuestas observables. Algunas de estas variables $x_1, x_2, \ldots x_p$ son controlables (o pueden serlo para los fines de la prueba) [61].

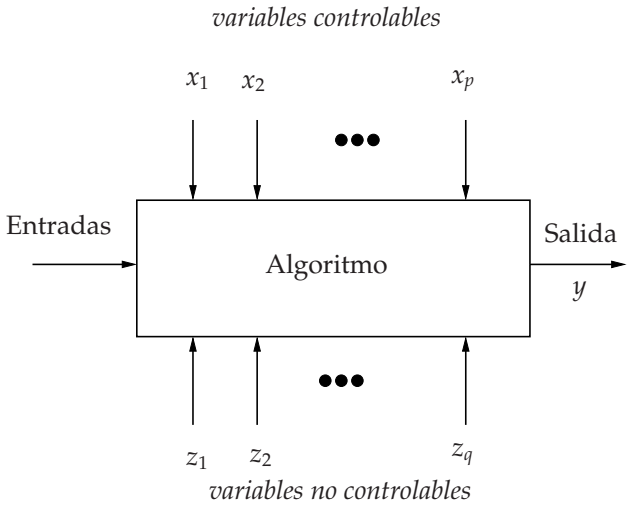



Figura 7.1: Modelo general de un proceso.

Para la elaboración de experimentos es necesario usar pruebas o ensayos que permitan obtener conocimiento acerca de un proceso, mecanismo o sistema en particular. Diseñar experimentos como una serie de pruebas que produzcan cambios deliberados en las variables de entrada de un proceso o sistema hace posible observar e identificar las causas de los cambios en la respuesta de salida [11].

El diseño de experimentos espera cumplir con los siguientes objetivos:

- Determinar cuáles variables tienen mayor influencia en la respuesta $\vec{y}$
- Determinar el mejor valor de $\vec{x}$ que influye en $\vec{y}$, de modo que $\vec{y}$ tenga un valor deseado.
- Determinar el mejor valor de $\vec{x}$ que influye en $\vec{y}$, de modo que la variabilidad de $\vec{y}$ sea pequeña y se minimicen los efectos de las variables incontrolables.

Adicionalmente, un diseño experimental tiene como finalidad desarrollar un proceso consistente o robusto, es decir un proceso afectado mínimamente por fuentes externas de variabilidad $\vec{z}$.

Para alcanzar los objetivos anteriormente descritos, realizamos evaluaciones de un conjunto de problemas de optimización con diversas características; comparamos configuraciones de diseño experimental y seleccionamos parámetros de diseño que ajustamos a un determinado intervalo.

## 7.1. Funciones de prueba

Una gran parte de los problemas de optimización presentan restricciones p.ej. físicas, geométricas, de tiempo entre otras, las cuales modifican la forma del espacio de búsqueda. Además, estos problemas pueden tener una alta dimensionalidad o estar sujetos a un gran número de restricciones.

Por lo anterior, en [55] se recopila un conjunto de funciones de prueba estándar para optimización con restricciones. Dicho conjunto se compone de 22 funciones con diversas características (descritas en las tablas 7.1 y 7.2). Algunas consideraciones de estas funciones se describen a continuación.

Todas las funciones utilizadas son transformadas al formato de las ecuaciones (7.1) y (7.2):

$$\text{Minimizar } f(\vec{x}), \vec{x} = [x_1, x_2, ..., x_n] \tag{7.1}$$

sujeto a:

$$\left\{ \begin{array}{c} g_i(\vec{x}) \leqslant 0, i = 1, ..., q \\ h_j(\vec{x}) = 0, i = q+1, ..., m \end{array} \right\} \tag{7.2}$$

Frecuentemente las restricciones de igualdad son transformadas a restricciones de desigualdad de la forma:

$$|h_j(\vec{x})| - \epsilon \leqslant 0, \text{ para } j = q+1, ..., m \tag{7.3}$$

Una solución $\vec{x}$ es considerada como factible sólo si $g_i(\vec{x}) \leqslant 0$ para $i = 1, ..., m$ y $|h_j(\vec{x})| - \epsilon \leqslant 0$, para $j = q + 1, ..., m$. El valor de $\epsilon$ se fija a 0.0001.

La tabla 7.1 muestra un resumen con las características de cada uno de los problemas. El valor del factor $\rho$ es la tasa de factibilidad que se obtuvo generando un millón de soluciones aleatorias para cada problema, y dividiendo la cantidad de soluciones factibles obtenidas entre el total. Por otro lado, la tabla 7.2 muestra el valor de la función objetivo en la solución óptima, así como los límites de las variables de decisión.

| Prob. | n | Tipo de función | $\rho$ | LI | NI | LE | NE | a |
|---|---|---|---|---|---|---|---|---|
| g01 | 13 | cuadrática | 0.0001 % | 9 | 0 | 0 | 0 | 6 |
| g02 | 20 | no lineal | 99.9965 % | 0 | 2 | 0 | 0 | 1 |
| g03 | 10 | polinomial | 0.0001 % | 0 | 0 | 0 | 1 | 1 |
| g04 | 5 | cuadrática | 27.0014 % | 0 | 6 | 0 | 0 | 2 |
| g05 | 4 | cúbica | 0.0000 % | 2 | 0 | 0 | 3 | 3 |
| g06 | 2 | cúbica | 0.0072 % | 0 | 2 | 0 | 0 | 2 |
| g07 | 10 | cuadrática | 0.0003 % | 3 | 5 | 0 | 0 | 6 |
| g08 | 2 | no lineal | 0.8555 % | 0 | 2 | 0 | 0 | 0 |
| g09 | 7 | polinomial | 0.5131 % | 0 | 4 | 0 | 0 | 2 |
| g10 | 8 | lineal | 0.0002 % | 3 | 3 | 0 | 0 | 6 |
| g11 | 2 | cuadrática | 0.0110 % | 0 | 0 | 0 | 1 | 1 |
| g12 | 3 | cuadrática | 4.7626 % | 0 | 1 | 0 | 0 | 0 |
| g13 | 5 | no lineal | 0.0000 % | 0 | 0 | 0 | 3 | 3 |
| g14 | 10 | no lineal | 0.0000 % | 0 | 0 | 3 | 0 | 3 |
| g15 | 3 | cuadrática | 0.0000 % | 0 | 0 | 1 | 1 | 2 |
| g16 | 5 | no lineal | 0.0218 % | 4 | 34 | 0 | 0 | 4 |
| g17 | 6 | no lineal | 0.0000 % | 0 | 0 | 0 | 4 | 4 |
| g18 | 9 | cuadrática | 0.0000 % | 0 | 13 | 0 | 0 | 6 |
| g19 | 15 | no lineal | 33.4464 % | 0 | 5 | 0 | 0 | 0 |
| g21 | 7 | lineal | 0.0000 % | 0 | 1 | 0 | 5 | 6 |
| g23 | 9 | lineal | 0.0000 % | 0 | 2 | 3 | 1 | 6 |
| g24 | 2 | lineal | 44.273 % | 0 | 2 | 0 | 0 | 2 |

Tabla 7.1: Resumen de las características de las funciones de prueba adoptadas. $n$ es el número de variables de decisión, $\rho = |F|/|S|$ es la tasa estimada (en porcentaje) entre la región factible y el espacio de búsqueda, $LI$ es el número de restricciones de desigualdad lineales, $NI$ el número de ecuaciones de desigualdad no lineales, $LE$ el número de funciones de restricción de igualdad lineales y $NE$ es el número de restricciones de igualdad no lineales. $a$ es el número de restricciones activas en $\vec{x}^*$.

| Prob. | n | $f_{(\vec{x}^*)}$ | límites de variables |
|---|---|---|---|
| g01 | 13 | -15.0000000000 | $0 \leqslant x_i \leqslant 1$ $(i = 1, ..., 9)$, $0 \leqslant x_i \leqslant 100$ $(i = 10, 11, 12)$ y $0 \leqslant x_1 3 \leqslant 1$ |
| g02 | 20 | -0.8036191042 | $0 < x_i \leqslant 10$ $(i = 1, ..., n)$ |
| g03 | 10 | -1.0005001000 | $0 \leqslant x_i \leqslant 1$ $(i = 1, ..., n)$ |
| g04 | 5 | -30665.5386717834 | $78 \leqslant x_1 \leqslant 102, 33 \leqslant x_2 \leqslant 45$ y $27 \leqslant x_i \leqslant 45$ $(i = 3, 4, 5)$ |
| g05 | 4 | 5126.4967140071 | $0 \leqslant x_1 \leqslant 1200, 0 \leqslant x_2 \leqslant 1200, -0.55 \leqslant x_3 \leqslant 0.55$ y $-0.55 \leqslant x_4 \leqslant 0.55$ |
| g06 | 2 | -6961.8138755802 | $13 \leqslant x_1 \leqslant 100$ y $0 \leqslant x_2 \leqslant 100$ |
| g07 | 10 | 24.3062090681 | $-10 \leqslant x_i \leqslant 10$ $(i = 1, ..., 10)$ |
| g08 | 2 | -0.0958250415 | $0 \leqslant x_1 \leqslant 10$ y $0 \leqslant x_2 \leqslant 10$ |
| g09 | 7 | 680.6300573745 | $-10 \leqslant x_i \leqslant 10$ para $(i = 1, ..., 7)$ |
| g10 | 8 | 7049.2480205286 | $100 \leqslant x_1 \leqslant 10000, 1000 \leqslant x_i \leqslant 10000$ $(i = 2, 3)$ y $10 \leqslant x_i \leqslant 1000$ $(i = 4, ..., 8)$ |
| g11 | 2 | 0.7499000000 | $-1 \leqslant x_1 \leqslant 1$ y $-1 \leqslant x_2 \leqslant 1$ |
| g12 | 3 | -1.0000000000 | $0 \leqslant x_i \leqslant 10$ $(i = 1, 2, 3)$ y $p, q, r = 1, 2, ..., 9$ |
| g13 | 5 | 0.0539415140 | $-2.3 \leqslant x_i \leqslant 2.3 (i = 1, 2)$ y $-3.2 \leqslant x_i \leqslant 3.2$ $(i = 3, 4, 5)$ |
| g14 | 10 | -47.7648884595 | $0 < x_i \leqslant 10$ $(i = 1, ..., 10)$ |
| g15 | 3 | 961.7150222899 | $0 \leqslant x_i \leqslant 10$ $(i = 1, 2, 3)$ |
| g16 | 5 | -1.9051552586 | $704.4148 \leqslant x_1 \leqslant 906.3855, 68.6 \leqslant x_2 \leqslant 288.88, 0 \leqslant x_3 \leqslant 134.75,$ $193 \leqslant x_4 \leqslant 287.0966, 25 \leqslant x_5 \leqslant 84.1988$ |
| g17 | 6 | 8853.5396748064 | $0 \leqslant x_1 \leqslant 400, 0 \leqslant x_2 \leqslant 1000, 340 \leqslant x_3 \leqslant 420, 340 \leqslant x_4 \leqslant 420,$ $-1000 \leqslant x_5 \leqslant 1000$ y $0 \leqslant x_6 \leqslant 0.5236$ |
| g18 | 9 | -0.8660254038 | $-10 \leqslant x_i \leqslant 10$ $(i = 1, ..., 8)$ y $0 \leqslant x_9 \leqslant 20$ |
| g19 | 15 | 32.6555929502 | $0 \leqslant x_i \leqslant 10$ $(i = 1, ..., 15)$ |
| g21 | 7 | 193.7245100700 | $0 \leqslant x_1 \leqslant 1000, 0 \leqslant x_2, x_3 \leqslant 40, 100 \leqslant x_4 \leqslant 300, 6.3 \leqslant x_5 \leqslant 6.7,$ $5.9 \leqslant x_6 \leqslant 6.4$ y $4.5 \leqslant x_7 \leqslant 6.25$ |
| g23 | 9 | -400.0551000000 | $0 \leqslant x_1, x_2, x_6 \leqslant 300, 0 \leqslant x_3, x_5, x_7 \leqslant 100, 0 \leqslant x_4, x_8 \leqslant 200$ |
| g24 | 2 | -5.5080132716 | $0 \leqslant x_1 \leqslant 3$ y $0 \leqslant x_2 \leqslant 4$ |

Tabla 7.2: Soluciones óptimas (o mejores valores conocidos) al conjunto de problemas. $n$ es el número de variables de decisión y $f_{(\vec{x}^*)}$ el valor de la función objetivo.

## 7.2. Diseño experimental

En el diseño de experimentos realizamos un proceso de planeación de pruebas para obtener datos apropiados, los cuales fueron analizados mediante métodos estadísticos, con el propósito de producir conclusiones válidas y objetivas.

La metodología que empleamos usa 3 principios básicos: obtención de réplicas, aleatorización y elección de factores.

- La **obtención de réplicas** se refiere a la repetición de ejecuciones del algoritmo. Este proceso permitió obtener una estimación para determinar si las diferencias observadas en los resultados son estadísticamente significativas. Además, permitió calcular una estimación precisa de las variaciones en los resultados de las soluciones óptimas de acuerdo a la ecuación (7.5).
- La **aleatorización**, por su parte, se refiere al uso de variables aleatorias independientes. Esto se implementó mediante el uso de diferentes semillas en cada una de las ejecuciones del algoritmo.
- La **elección de factores y sus niveles**. Establecimos parámetros con un valor fijo y otros que variaran sus valores en ciertos intervalos, como se observa más adelante y en el capítulo 7 con el análisis de varianza.

En cada algoritmo generamos una posible solución óptima al conjunto de problemas. Por lo tanto, el experimento consistió en efectuar 100 ejecuciones independientes, con diferente semilla inicial para el generador de números aleatorios de cada algoritmo.

Una vez realizadas las ejecuciones para un determinado problema obtuvimos 100 soluciones, las cuales conformaron nuestro conjunto de datos experimentales. Posteriormente, se aplicó un análisis estadístico al conjunto y se presentaron los siguientes datos, para cada función y cada algoritmo.

- La mejor y la peor solución del conjunto de soluciones.
- La **mediana**, que es el valor que divide la distribución de datos ordenados en dos partes. Por ejemplo, la mediana de los valores (1 1 1 2 3 4 5) es de 2. Si el número de datos es par, la muestra tiene dos valores centrales, y la mediana se encuentra por interpolación entre ellas o mediante la selección de uno de ellos arbitrariamente.
- La **media aritmética** o valor medio que se denota por la ecuación (7.4). Es la suma de todos los valores de la muestra dividida por el número de valores $n$.

$$\mu = \frac{1}{n}\sum_{i=1}^{n} f_i(\vec{x}) \tag{7.4}$$

- La **desviación estándar** es la medida más común de dispersión en torno a la media de una distribución. Se define como la raíz cuadrada de la varianza. La varianza representa la media aritmética de las desviaciones, con respecto a la media, elevadas

al cuadrado. Si se considera la colección completa de datos (población en su totalidad) obtenemos la varianza poblacional. Por el contrario, si sólo consideramos una muestra de la población, obtenemos una varianza muestral. En las ecuaciones (7.5) y (7.6) se muestra la definición matemática para la varianza poblacional y la desviación estándar, respectivamente.

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(f_i(\vec{x}) - \mu)^2 \tag{7.5}$$

$$\sigma = \sqrt{(\sigma)} \tag{7.6}$$

- La **media del número de soluciones factibles** (en la población), de la última iteración en cada una de las 100 ejecuciones.
- La **media del número mínimo de iteraciones** para alcanzar una solución óptima. Esto es, si $f(\hat{x}) - f(\hat{x}^*) \leqslant \epsilon$, donde $\epsilon = 0.00001$ es un factor que indica el error relativo a cada solución del problema.

## 7.3. Parámetros de control

Consideramos un parámetro de control como aquel factor que varía de acuerdo a ciertos rangos establecidos. Nuestra propuesta utiliza los siguientes parámetros:

- Para el mecanismo de búsqueda global (**evolución diferencial**):
  - $F$: probabilidad de mutación.
  - $C$: probabilidad de cruza.

  De acuerdo al algoritmo original (sin restricciones) en el que se basó esta propuesta, se establece que ambos parámetros toman un valor de 0.9.
- Para el mecanismo de búsqueda local (**cruza SPX**):
  - $\epsilon$: tasa de expansión o contracción en el simplex.
  - $n_p$: número de soluciones para generar el vecindario.
- Para el mecanismo de manejo de restricciones (**jerarquías estocásticas**):
  - $Pf$: tasa de balance para establecer prioridad de ordenamiento respecto a la función objetivo y la función de penalización.
- Otros parámetros como:
  - *pob*: número de soluciones (población) que evolucionan en cada iteración del algoritmo.
  - *gen*: número de iteraciones (generaciones) que realiza el algoritmo.
  - F. penalización. Es la función de penalización que involucra a todas las funciones de restricción del problema.

- $R$: de la propuesta de la literatura denominada LEDE [1] se refiere a un factor de penalización para el proceso de manejo de restricciones.

Los valores establecidos para cada parámetro de control en los algoritmos de la literatura especializada se definieron de acuerdo a la fuente original de consulta. Los valores en los parámetros de nuestra propuesta (AMER) se establecieron con base en pruebas exhaustivas. La tabla 7.3 muestra el valor para cada parámetro. El valor para el número total de evaluaciones de la función objetivo EFO, se define de acuerdo a las referencias de cada propuesta.

| Parámetros. | SR | MCODE | LEDE | AMER |
|---|---|---|---|---|
| F. penalización | $\sum_{j=1}^{m} max\{0, g_j(x)\}^2;$ | $\sum_{j=1}^{m} \lvert g_j(x)\rvert;$ | $\sum_{j=1}^{m} max\{0, g_j(x)\};$ | $\sum_{j=1}^{m} max\{0, g_j(x)\}^2;$ |
| población (pob) | 200 | 40 | 101 | 70 |
| generaciones (gen) | 1,750 | 2,000 | 2,500 | 2,500 |
| tasa mutación (F) | ——- | 0.8 | autoadaptable | 0.9 |
| tasa recombinación (C) | ——- | 0.8 | 0.9 | 0.9 |
| tasa balance (Pf) | 0.45 | ——- | ——- | 0.45 |
| penalización (R) | ——- | ——- | 10,000 | —- |
| tasa de expansión ($\epsilon$) | ——- | ——- | ——- | 1.3 |
| eval. fun. obj. (EFO) | 350,000 | 242,000 | 255,000 | 180,000 |

Tabla 7.3: Resumen de los valores utilizados en cada parámetro de los 4 algoritmos: jerarquías estocásticas (SR) [76], evolución diferencial con cooperación co-evolutiva (MCODE) [29], exploración local basada en evolución diferencial (LEDE) [1] y nuestra propuesta, denominada algoritmo memético para espacios restringidos (AMER). EFO se refiere al número total de evaluaciones de la función objetivo.

## 7.4. Resultados experimentales

Las estadísticas mencionadas en la sección de diseño experimental y los diagramas que a continuación describimos forman parte de los resultados experimentales de este trabajo de tesis. Hacemos referencia en cada uno de los algoritmos por sus siglas como sigue: jerarquías estocásticas (SR), evolución diferencial con cooperación co-evolutiva (MCODE), exploración local basada en evolución diferencial (LEDE1 y LEDE2) y nuestra propuesta, denominada algoritmo memético para espacios restringidos(AMER). Los diagramas que se presentan son:

- Diagrama de caja.
- Gráficas de convergencia.
- Histogramas.

**Diagramas de caja**.

El diagrama de caja es un medio muy útil para representar datos. En dicho diagrama, los valores mínimo y máximo, los cuartiles inferior y superior (percentiles 25 y 75, respectivamente) y la mediana (percentil 50) se representa en una caja rectangular alineada ya sea horizontal o verticalmente. La caja se extiende del cuartil inferior al superior, y es atravesada de un lado al otro por la mediana. A partir de los extremos de la caja se extienden las líneas ("bigotes") hasta los valores mínimo y máximo. Además representa posibles valores atípicos en el conjunto de datos (ver figura 7.2).

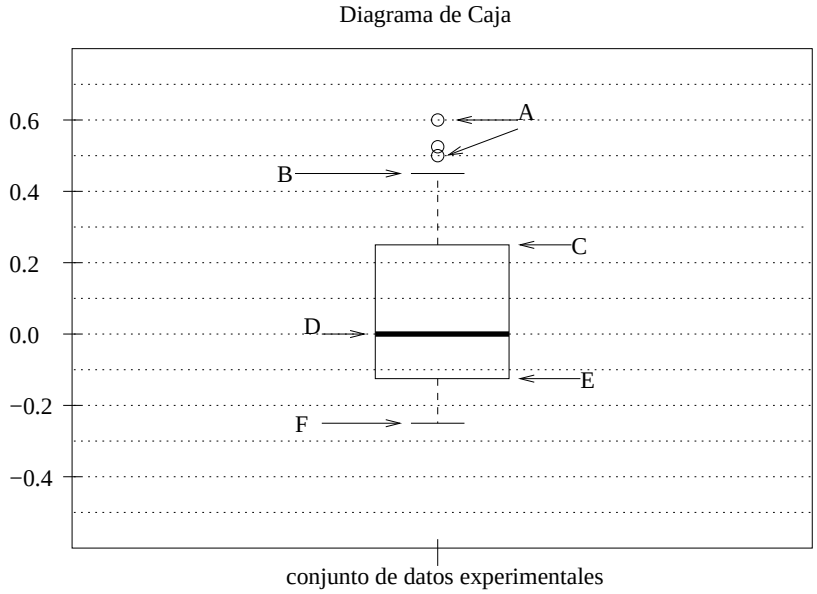



Figura 7.2: Esquema de un diagrama de caja. *A* valores atípicos, *B* valor máximo, *C* cuartil superior, *D* mediana, *E* cuartil inferior, *F* valor mínimo.

En el apéndice B.1 mostramos los diagramas de caja del conjunto de datos experimentales (100 ejecuciones independientes) de cada algoritmo, el eje $y$ representa el valor de la función objetivo, mientras que el $x$ hace mención a los algoritmos que comparamos.

En la figura 7.3 se presentan los resultados para el problema g17. Puede observarse que *SR*, a pesar de no presentar varianza en las soluciones, cae en un mínimo local. Por otro lado, *LEDE* en su versión 1 presenta varias soluciones como valores atípicos y *MCODE* al igual que *AMER* se aproximan al valor del óptimo global en su mediana, pero el valor mínimo de AMER es el valor óptimo del problema.

**Gráficas de convergencia**.

Generamos gráficas de convergencia a partir de 30 ejecuciones independientes, mediante el cálculo del valor promedio de las soluciones en un determinado número de generaciones (épocas). Graficamos en el eje de las abcisas ($x$) el número de evaluaciones de la función objetivo de acuerdo a las generaciones y en el eje de las ordenadas ($y$) el valor de la función objetivo.

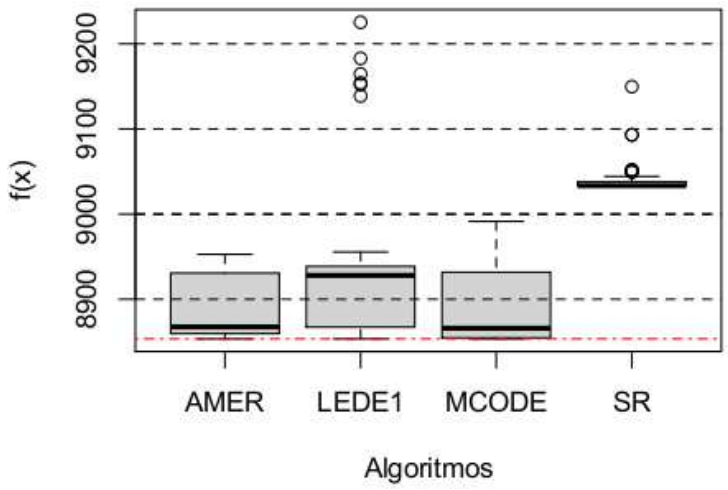



Figura 7.3: Una comparación de los diagramas de caja de cada algoritmo para el problema 17. La línea roja muestra el valor de la solución óptima.

En el apéndice B.2 se presenta una comparación de las gráficas de convergencia de los 4 algoritmos meméticos para espacios restringidos en cada uno de los 22 problemas.

En la figura 7.5 se aprecia la evolución de los 4 algoritmos de acuerdo al número de evaluaciones de la función objetivo. Nuestra propuesta presenta el mejor comportamiento en este problema, aunque no converge tan rápido como LEDE1, LEDE2 y MCODE, logra llegar a la solución óptima antes de caer en un óptimo local. En esta gráfica la convergencia para los algoritmos LEDE1 y LEDE 2 es la misma.

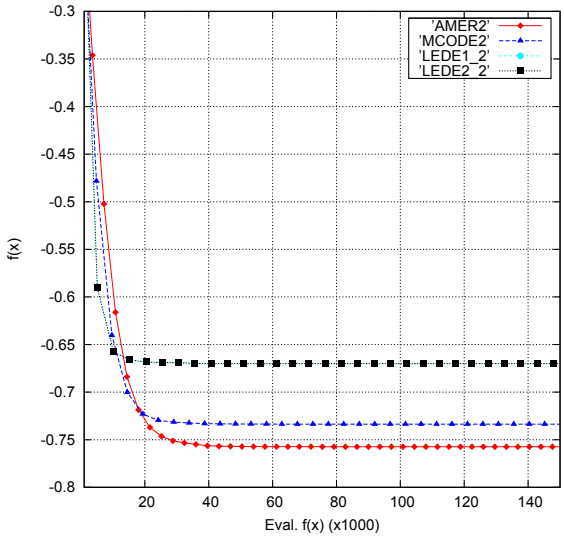



Figura 7.4: Gráfica de convergencia en los 4 algoritmos meméticos para espacios restringidos (MCODE, LEDE1, LEDE2 y AMER) para el problema g02.

**Histogramas**.

En un histograma se representa la tendencia central de la dispersión y la forma general de la distribución de los datos. Un histograma se construye dividiendo el eje horizontal en intervalos (por lo regular de la misma longitud) y trazando sobre el $j-$ésimo intervalo un rectángulo con área proporcional a $n_j$ (el número de observaciones que caen en ese intervalo).

En la figura 7.5 se presenta el histograma de frecuencias de nuestra propuesta para el problema g13. Puede observarse que las frecuencias no se ajustan a una distribución normal.

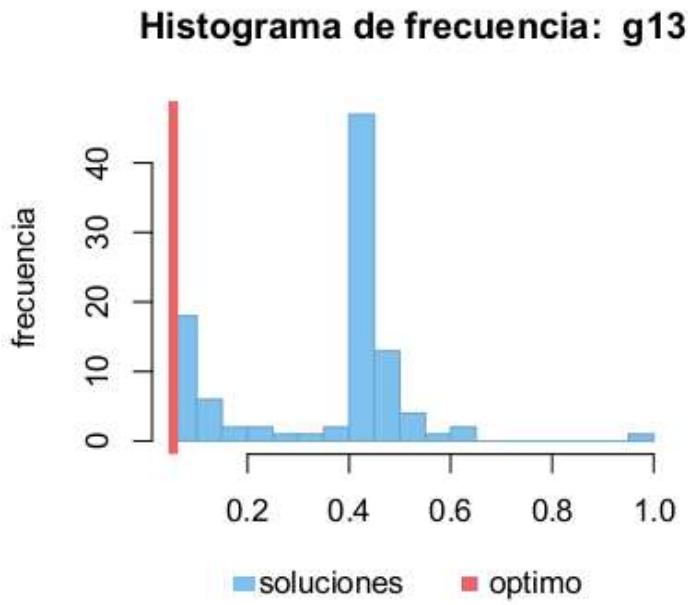



Figura 7.5: histograma de las frecuencias de las soluciones en AMER para el problema g13.

### 7.4.1. Estudio comparativo

Para realizar nuestro estudio comparativo se empleó la versión en línea de jerarquías estocásticas (SR) proporcionada por su autor en el lenguaje de cómputo científico Matlab. Este modelo sólo resuelve 13 de las 22 funciones de prueba; por lo tanto, agregamos las 9 funciones restantes. Los algoritmos meméticos: evolución diferencial con cooperación co-evolutiva (MCODE) y exploración local basada en evolución diferencial (LEDE), en sus dos versiones, fueron implementados en lenguaje C y su población inicial fue generada de la misma forma que en nuestra propuesta (mediante hipercubos latinos).

Los resultados se agrupan en una tabla comparativa (7.4) que muestra los resultados estadísticos de cada uno de los modelos.

De acuerdo a las referencias de cada modelo el número de evaluaciones de la función objetivo está definido de la siguiente manera: SR utiliza 355,00, MCODE 242,000 y LEDE 255,000, mientras que nuestra propuesta sólo requiere de 180,000 evaluaciones.

A partir del conjunto de resultados obtenidos de los experimentos realizados con los 5 algoritmos se realizó una comparativa de acuerdo a la diferencia de cada posible solución óptima $f(\vec{x})$ y el óptimo real conocido $f(\vec{x}^*)$. Graficamos estos valores con diagramas de caja, donde puede apreciarse la robustez de nuestra propuesta al resolver el conjunto de las 22 funciones (ver figura 7.6).

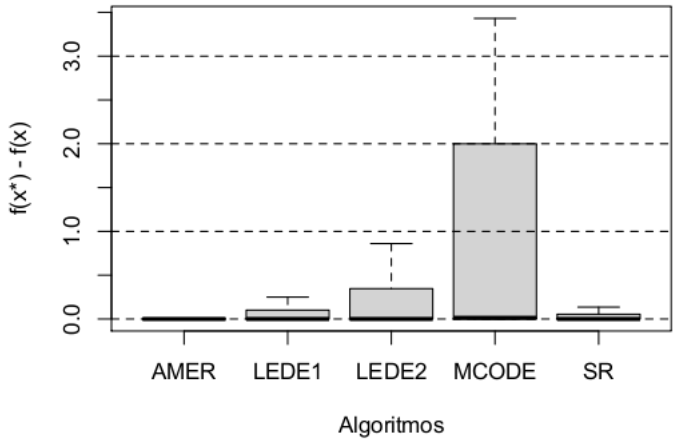



Figura 7.6: Diagramas de caja de las diferencias de la solución óptima conocida $f(\vec{x}^*)$ y las posibles soluciones que proporciona cada algoritmo $f(\vec{x})$.

En la tabla 7.4 presentamos valores resaltados con negritas, los cuales se refieren al mejor valor de los 5 algoritmos. Aquellos resultados que no aparecen en la tabla son porque el algoritmo no convergió a una solución factible y no se aproximan al óptimo global. Es decir, sólo consideramos las ejecuciones exitosas.

De acuerdo a los resultados estadísticos presentados, podemos observar que nuestra propuesta resuelve el conjunto de funciones de forma exitosa, ya que nunca se queda estancada en un óptimo local, como le ocurre en varios problemas a MCODE. A pesar de que SR y LEDE tienen una menor desviación estándar en algunos problemas, estos algoritmos fallan en problemas como g16, g17, g21 y g23. Además, AMER presenta un porcentaje de soluciones factibles de 100 % en todos los casos.

Los problemas en los que AMER presenta mayor valor de la desviación estándar son g05, g17, g21 y g23, los cuales se resuelven en el capítulo 8 al ajustar el valor de los parámetros, mediante un análisis de varianza.

Es importante mencionar que AMER supera en media, mediana, desviación estándar, mejor y peor solución a los demás algoritmos en 13 de las 22 funciones de prueba, y no

presenta diferencias significativas en los demás casos.

### 7.4.2. Efecto del buscador local

Uno de los principales objetivos de esta tesis fue corroborar que el uso de un buscador local mejora el desempeño de un algoritmo. Por lo tanto, modificamos el algoritmo propuesto eliminando la etapa de búsqueda local, y realizamos los mismos experimentos mencionados anteriormente (100 ejecuciones independientes), con las mismas semillas para la generación de números aleatorios que en la propuesta original.

Generamos las gráficas de convergencia (ver tabla 7.4.2) que muestran la forma en que evoluciona una y otra versión del algoritmo, haciéndose evidente la superioridad y el beneficio de un algoritmo que acople de manera adecuada un mecanismo de búsqueda local. En el apéndice A.3 se muestran las gráficas de convergencia de los 22 problemas.

En la tabla 7.6 presentamos los resultados estadísticos de las dos versiones y comprobamos que en 19 de los 22 problemas AMER en su versión original es mejor que en la versión donde no se usa el buscador local. En los 3 problemas restantes la versión sin buscador local únicamente mejora el valor de la desviación estándar mínimamente.

Para realizar un estudio más completo, en la tabla 7.4.2 presentamos otras estadísticas de la versión con buscador local. En nuestro conjunto de datos experimentales calculamos el porcentaje de soluciones factibles de acuerdo al número de soluciones factibles que se tienen en la última generación de cada ejecución. Además, calculamos el número mínimo de evaluaciones de la función objetivo que se requieren para converger al óptimo global y finalmente mostramos el porcentaje de mejoras después de usar el buscador local, es decir del número total de llamadas al mecanismo de búsqueda local, contamos en cuántas ocasiones se mejoró a la solución base.

| **Prob. EFO** | **Estadist** | **SR** 350,000 | **MCODE** 242,000 | **LEDE1** 255,000 | **LEDE2** 255,000 | **AMER** 180,000 |
|---|---|---|---|---|---|---|
| g01 -15.0 | mejor | -15.0 | -15.0 | -15.0 | -15.0 | -15.0 |
| | mediana | -15.0 | -15.0 | -15.0 | -15.0 | -15.0 |
| | media | **-15.0** | -14.941395 | -14.376875 | -14.2025 | **-15.0** |
| | desv.est. | **0.0** | 0.255401 | 0.981845 | 1.017718 | **0.0** |
| | peor | **-15.0** | -13.828125 | -10.65625 | -11.28125 | **-15** |
| | factibles | 100 % | 100 % | 100 % | 100 % | 100 % |
| | $eval_{min}$ | 149,600 | 89,983 | 146,991 | 158,223 | 85,856 |
| g02 -0.803619 | mejor | -0.803516 | -0.803618 | **-0.803619** | **-0.803619** | -0.803618 |
| | mediana | **-0.782996** | -0.743684 | -0.60664 | -0.60664 | -0.770397 |
| | media | **-0.778025** | -0.733693 | -0.606603 | -0.606603 | -0.761270 |
| | desv.est. | **2.3E-02** | 5.51E-02 | 0.105017 | 0.105017 | 3.3E-02 |
| | peor | **-0.712949** | 0.003045 | 0.011028 | 0.011028 | -0.671374 |
| | factibles | 100 % | 100 % | 100 % | 100 % | 100 % |
| | $eval_{min}$ | 237,600 | ———- | 255,000 | 255,000 | 180,000 |
| g03 -1.0005001 | mejor | -1.000 | -1.000499 | **-1.0005001** | **-1.0005001** | **-1.0005001** |
| | mediana | -1.000 | -0.380903 | **-1.0005001** | **-1.0005001** | -1.000492 |
| | media | -1.000 | -0.375746 | **-1.0005** | -0.998551 | -0.999010 |
| | desv.est. | **1.9E-04** | 0.216069 | 9.949875 | 1.93E-02 | 7.04E-03 |
| | peor | -1.000 | -0.023192 | **-1.0005** | -0.806045 | -0.961216 |
| | factibles | 100 % | 97 % | 100 % | 100 % | 100 % |
| | $eval_{min}$ | 219,800 | ———- | 120,396 | 143,337 | 173,002 |
| g04 -30665.538671 | mejor | -30665.539 | -30665.538671 | -30665.538671 | -30665.538671 | -30665.538671 |
| | mediana | -30665.539 | -30665.538671 | -30665.538671 | -30665.538671 | -30665.538671 |
| | media | -30665.331 | -30665.538671 | -30665.538671 | -30665.538671 | -30665.538671 |
| | desv.est. | 1.3 | **7.28E-12** | **7.28E-12** | **7.28E-12** | **7.28E-12** |
| | peor | -30653.396 | -30665.538671 | -30665.538671 | -30665.538671 | -30665.538671 |
| | factibles | 100 % | 100 % | 100 % | 100 % | 100 % |
| | $eval_{min}$ | 84,600 | 39,244 | 40,994 | 40,408 | 59,286 |
| g05 5126.496714 | mejor | 5126.497 | 5126.496714 | 5126.496714 | ———— | 5126.496714 |
| | mediana | 5127.632 | **5126.496714** | **5126.496714** | 5065.704566 | **5126.496714** |
| | media | 5130.752 | 5126.675293 | **5126.56589** | ———— | 5126.947831 |
| | desv.est. | 7.1 | 1.776839 | **0.254565** | 1048.184 | 4.658087 |
| | peor | 5159.492 | 5144.354627 | **5128.068055** | 5450.900752 | 5141.989532 |
| | factibles | 99 % | 100 % | 100 % | 36 % | 100 % |
| | $eval_{min}$ | 51,800 | 27,503 | 158,446 | 237,958 | 157,985 |
| g06 -6961.813875 | mejor | -6961.814 | -6961.813875 | -6961.813875 | -6961.813875 | -6961.813875 |
| | mediana | -6961.814 | -6961.813875 | -6961.813875 | -6961.813875 | -6961.813875 |
| | media | -6882.753 | -6960.596535 | **-6961.813875** | **-6961.813875** | **-6961.813875** |
| | desv.est. | 1.7E+02 | 12.11238 | 5.45E-12 | 5.45E-12 | **2.72E-12** |
| | peor | -6263.125 | -6840.079871 | **-6961.813875** | **-6961.813875** | **-6961.813875** |
| | factibles | 100 % | 100 % | 100 % | 100 % | 100 % |
| | $eval_{min}$ | 121,800 | 19,601 | 19,634 | 21,297 | 22,403 |
| g07 24.306209 | mejor | 24.309 | **24.306209** | **24.306209** | **24.306209** | **24.306209** |
| | mediana | 24.365 | 24.306268 | **24.306209** | **24.306209** | 24.306221 |
| | media | 24.397 | 24.311116 | **24.306209** | **24.306209** | 24.306293 |
| | desv.est. | 1.2E-01 | 4.41E-02 | **7.57-07** | 9.14E-07 | 1.79E-04 |
| | peor | 25.158 | 24.750366 | **24.306215** | 24.306218 | 24.306933 |
| | factibles | 100 % | 100 % | 100 % | 100 % | 100 % |
| | $eval_{min}$ | 142,400 | 240,000 | 255,000 | 250,803 | 178,207 |
| g08 -0.0958250415 | mejor | -0.095825 | -0.0958250414 | -0.0958250414 | -0.0958250414 | -0.0958250414 |
| | mediana | -0.095825 | -0.0958250414 | -0.0958250414 | -0.0958250414 | -0.0958250414 |
| | media | -0.095825 | -0.0958250414 | -0.0958250414 | -0.0958250414 | -0.0958250414 |
| | desv.est. | 3.0E-17 | 4.16E-17 | 4.16E-17 | 4.16E-17 | 4.16E-17 |
| | peor | -0.095825 | -0.0958250414 | -0.0958250414 | -0.0958250414 | -0.0958250414 |
| | factibles | 100 % | 100 % | 100 % | 100 % | 100 % |
| | $eval_{min}$ | 79,600 | 240,000 | 255,000 | 255,000 | 78,700 |

| Prob. EFO | Estadist | SR 350,000 | MCODE 242,000 | LEDE1 255,000 | LEDE2 255,000 | AMER 180,000 |
|---|---|---|---|---|---|---|
| g09 680.630057374 | mejor | 680.630 | 680.630057374 | 680.630057374 | 680.630057374 | 680.630057374 |
| | mediana | 680.643 | 680.630057374 | 680.630057374 | 680.630057374 | 680.630057374 |
| | media | 680.657 | 680.630057650 | 680.630057374 | 680.630057374 | 680.630057374 |
| | desv.est. | 3.7E-02 | 8.11E-07 | **4.54E-13** | **4.54E-13** | 5.68E-13 |
| | peor | 680.798 | 680.630062031 | 680.630057374 | 680.630057374 | 680.630057374 |
| | factibles | 100 % | 100 % | 100 % | 100 % | 100 % |
| | $eval_{min}$ | 114,400 | 221,959 | 44,886 | 45,621 | 53,616 |
| g10 7049.24802052 | mejor | 7049.362 | 7049.2480231 | 7049.2456988 | **7049.2480205** | **7049.2480205** |
| | mediana | 7308.123 | 7049.2506298 | 7049.2480384 | **7049.2480209** | 7049.2480633 |
| | media | 7431.718 | 7050.0327685 | 7049.2484410 | **7049.2480558** | 7049.2482797 |
| | desv.est. | 4.0E+02 | 5.1339 | 1.73E-03 | **1.03E-04** | 6.15E-04 |
| | peor | 9178.518 | 7091.2260265 | 7049.2599387 | **7049.2485324** | 7049.2507597 |
| | factibles | 100 % | 100 % | 100 % | 100 % | 100 % |
| | $eval_{min}$ | 228,400 | ———— | 248,022 | 244,906 | 179,950 |
| g11 0.7499 | mejor | 0.750 | 0.7499 | 0.7499 | 0.7499 | 0.7499 |
| | mediana | 0.750 | 0.7499 | 0.7499 | 0.7499 | 0.7499 |
| | media | 0.750 | 0.753125 | **0.749899** | **0.749899** | **0.749899** |
| | desv.est. | 2.9E-04 | 1.98E-02 | 1.77E-15 | 1.77E-15 | **1.11E-16** |
| | peor | 0.753 | 0.922890 | **0.7499** | **0.7499** | **0.7499** |
| | factibles | 100 % | 100 % | 100 % | 100 % | 100 % |
| | $eval_{min}$ | 16,000 | 19,988 | 25,152 | 28,027 | 30,677 |
| g12 -1.0 | mejor | -1.0 | -1.0 | -1.0 | -1.0 | -1.0 |
| | mediana | -1.0 | -1.0 | -1.0 | -1.0 | -1.0 |
| | media | -1.0 | -1.0 | -1.0 | -1.0 | -1.0 |
| | desv.est. | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| | peor | -1.0 | -1.0 | -1.0 | -1.0 | -1.0 |
| | factibles | 100 % | 100 % | 100 % | 100 % | 100 % |
| | $eval_{min}$ | 14,400 | 5,626 | 10,481 | 10,485 | 10,984 |
| g13 0.053941514 | mejor | 0.053942 | 0.053941514 | 0.053941514 | 0.053941514 | 0.053941514 |
| | mediana | **0.055061** | 0.438802607 | 0.438802607 | 3.62E+22 | 0.438802607 |
| | media | **0.093398** | 0.26628848 | 0.345125387 | 6.09E+38 | 0.331740272 |
| | desv.est. | **1.1E-01** | 0.196880 | 0.202528 | 6.03E+39 | 0.162748 |
| | peor | **0.441809** | 0.666596478 | 0.999999998 | 6.06E+40 | 0.586825517 |
| | factibles | 100 % | 98 % | 100 % | 30 % | 100 % |
| | $eval_{min}$ | 66,400 | 203,441 | 247,473 | 252,844 | 178,786 |
| g14 -47.76488845 | mejor | -47.756 | -47.764880 | **-47.764888** | **-47.764888** | **-47.764888** |
| | mediana | -47.489 | -47.725303 | -47.757635 | -47.744158 | **-47.764845** |
| | media | -47.432 | -47.563182 | -47.698232 | -47.432159 | **-47.736939** |
| | desv.est. | 2.5E-01 | 0.350616 | 0.131304 | 0.809345 | **0.128721** |
| | peor | -46.709 | -46.058498 | **-46.970433** | -43.175205 | -46.719956 |
| | factibles | 100 % | 100 % | 100 % | 84 % | 100 % |
| | $eval_{min}$ | 248,000 | ————— | 255,000 | 255,000 | 178,476 |
| g15 961.7150222899 | mejor | 961.715 | 961.715022 | 961.715022 | 961.715022 | 961.715022 |
| | mediana | 961.716 | **961.715022** | **961.715022** | 961.715512 | **961.715022** |
| | media | 961.727 | 961.726270 | 962.525543 | 962.515159 | **961.715760** |
| | desv.est. | 3.6E-02 | 7.19E-02 | 1.274311 | 1.266436 | **3.56E-03** |
| | peor | **962.016** | 962.266722 | 966.712703 | 967.981741 | 961.734847 |
| | factibles | 100 % | 100 % | 100 % | 100 % | 100 % |
| | $eval_{min}$ | 37,000 | 71,843 | 168688.43 | 186,633 | 76,691 |
| g16 -1.9051552586 | mejor | —— | -1.9051552 | -1.9051552 | -1.9051552 | -1.9051552 |
| | mediana | —— | -1.9051552 | -1.9051552 | -1.9051552 | -1.9051552 |
| | media | —— | -1.9051552 | -1.9051552 | -1.9051552 | -1.9051552 |
| | desv.est. | —— | 1.77E-15 | 1.77E-15 | 1.77E-15 | **1.33E-15** |
| | peor | —— | -1.9051552 | -1.9051552 | -1.9051552 | -1.9051552 |
| | factibles | 0 % | 100 % | 100 % | 100 % | 100 % |
| | $eval_{min}$ | —— | 240,000 | 255,000 | 255,000 | 148,307 |

| **Prob. EFO** | **Estadist** | **SR** 350,000 | **MCODE** 242,000 | **LEDE1** 255,000 | **LEDE2** 255,000 | **AMER** 180,000 |
|---|---|---|---|---|---|---|
| g17 8853.5396748 | mejor | 9037.512 | **8853.533874** | **8853.533874** | ———- | **8853.533874** |
| | mediana | 9061.612 | **8867.712477** | 8928.037827 | ———- | 8913.737242 |
| | media | 9071.548 | **8895.791052** | 8926.138622 | ———- | 8899.960982 |
| | desv.est. | **3.2E+01** | 40.756952 | 70.880665 | ———- | 36.999861 |
| | peor | 9206.596 | 8991.541114 | 9225.310583 | ———- | **8951.245730** |
| | factibles | 100 % | 100 % | 100 % | 0 % | 100 % |
| | $eval_{min}$ | 75,800 | 224,578 | 245,710 | ——— | 180,000 |
| g18 -0.8660254038 | mejor | -0.866 | -0.86602540 | -0.86602540 | -0.86602540 | -0.86602540 |
| | mediana | -0.866 | -0.86602540 | -0.86602540 | -0.86602540 | -0.86602540 |
| | media | -0.775 | -0.80869711 | -0.80887251 | -0.80871072 | **-0.8596571** |
| | desv.est. | 1.3E-01 | 8.75E-02 | 9.10E-02 | 8.75E-02 | **3.42E-02** |
| | peor | -0.369 | -0.67480185 | -0.50000000 | -0.67486887 | **-0.67498108** |
| | factibles | 99 % | 100 % | 100 % | 100 % | 100 % |
| | $eval_{min}$ | 135,800 | 240000 | 255,000 | 255,000 | 115,592 |
| g19 32.6555929502 | mejor | 33.026 | **32.65559916** | 32.65604796 | 32.65568535 | 32.65570696 |
| | mediana | 34.298 | **32.66002390** | 32.77152798 | 32.78120648 | 32.74363011 |
| | media | 34.524 | **32.70564325** | 32.88966567 | 32.86238124 | 32.80925845 |
| | desv.est. | 1.1 | **0.115734** | 0.314603 | 0.228500 | 0.155246 |
| | peor | 38.499 | 33.40959739 | 34.44596299 | 33.74305549 | **33.20651337** |
| | factibles | 100 % | 100 % | 100 % | 100 % | 100 % |
| | $eval_{min}$ | 240,200 | 242,000 | 255,000 | 255,000 | 180,000 |
| g21 193.7245100700 | mejor | —— | 193.72451 | 193.72451 | 193.72451 | 193.72451 |
| | mediana | —— | **193.72451** | **193.72451** | 265.9657262 | 193.72451 |
| | media | —— | 231.2670730 | 229.7260339 | 253.9543500 | **204.80409** |
| | desv.est. | —— | 53.717300 | 54.985011 | 56.459864 | **30.100436** |
| | peor | —— | 324.703787 | **324.702841** | 330.440790 | **324.702841** |
| | factibles | 0 % | 100 % | 96 % | 100 % | 100 % |
| | $eval_{min}$ | —— | 181,060 | 188,252 | 217,581 | 169,271 |
| g23 -400.0551000000 | mejor | —— | -400.0550858 | **-400.0551000** | -683.2805323 | -400.0543932 |
| | mediana | —— | -345.7214344 | -327.4867220 | -283.7895982 | **-371.8092445** |
| | media | —— | -307.3926895 | -239.6527288 | -209.9972935 | **-364.1425693** |
| | desv.est. | —— | 108.031126 | 144.590138 | 181.104037 | **31.567149** |
| | peor | —— | 179.650350 | -0.071326 | 267.894016 | **-288.0700031** |
| | factibles | 0 % | 100 % | 100 % | 79 % | 100 % |
| | $eval_{min}$ | —— | 240,000 | 255,000 | 255,000 | 180,000 |
| g24 -5.5080132716 | mejor | -5.508 | -5.50801327 | -5.50801327 | -5.50801327 | -5.50801327 |
| | mediana | -5.508 | -5.50801327 | -5.50801327 | -5.50801327 | -5.50801327 |
| | media | -5.508 | -5.50801327 | -5.50801327 | -5.50801327 | -5.50801327 |
| | desv.est. | 7.1E-15 | 8.88E-15 | 8.88E-15 | 8.88E-15 | **8.88E-16** |
| | peor | -5.508 | -5.50801327 | -5.50801327 | -5.50801327 | -5.50801327 |
| | factibles | 100 % | 100 % | 100 % | 100 % | 100 % |
| | $eval_{min}$ | 23,200 | 237,602 | 255,000 | 255,000 | 27,597 |

Tabla 7.4: Comparación de los resultados estadísticos de los 5 algoritmos para las 22 funciones de prueba. Consideramos el número de evaluaciones de la función objetivo $f$, requeridas para la solución factible $(\vec{x^*}, f(\vec{x^*}))$. *Factibles* se refiere al porcentaje del valor promedio de soluciones factibles en el conjunto de datos experimentales y $eval_{min}$ es el valor promedio de evaluaciones mínimas que se requieren para que el algoritmo alcance una solución $f(\vec{x}) < \epsilon$, donde $\epsilon$ es el error porcentual de estimación al óptimo global. Se aprecia que los resultados resaltados en negritas son los mejores de los 5 algoritmos.

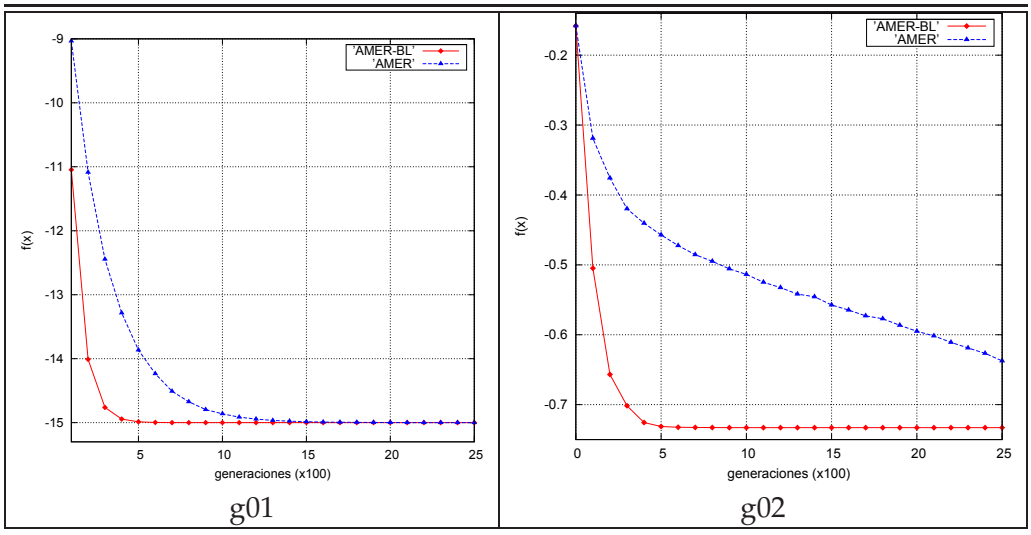



Tabla 7.5: Gráficas de convergencia que ilustran la evolución de las dos versiones del algoritmo con y sin buscador local, para los problemas g01 y g02, siendo evidente la superioridad de la propuesta original.

| Prob | Versión | mejor | mediana | media | desv. est. | peor |
|---|---|---|---|---|---|---|
| g01 | SBL | -14.999946 | -14.999805 | -14.999823 | 9.91E-05 | -14.999427 |
| | CBL | **-15.0** | **-15.0** | **-15.0** | **0.0** | **-15.0** |
| g02 | SBL | -0.716864 | -0.630202 | -0.623580 | 3.37E-02 | -0.545416 |
| | CBL | **-0.803618** | **-0.770397** | **-0.761270** | **3.30E-02** | **-0.671374** |
| g03 | SBL | -0.835070 | -0.273257 | -0.256595 | 0.122793 | -0.083632 |
| | CBL | **-1.0005001** | **-1.000492** | **-0.999010** | **7.04E-03** | **-0.961216** |
| g04 | SBL | -30665.538671 | -30665.538671 | -30665.538671 | 7.27E-12 | -30665.538671 |
| | CBL | -30665.538671 | -30665.538671 | -30665.538671 | 7.28E-12 | -30665.538671 |
| g05 | SBL | 5126.496714 | 5153.272313 | 5131.352670 | 47.776368 | 5312.815637 |
| | CBL | 5126.496714 | **5126.496714** | **5126.947831** | **4.658087** | **5141.989532** |
| g06 | SBL | -6961.813875 | -6961.813875 | -6961.813875 | 5.45E-12 | -6961.8138 |
| | CBL | -6961.813875 | -6961.813875 | -6961.813875 | **2.72E-12** | -6961.813875 |
| g07 | SBL | 24.336617 | 24.364586 | 24.362005 | 1.44E-02 | 24.413607 |
| | CBL | **24.306209** | **24.306221** | **24.306293** | **1.79E-04** | **24.306922** |
| g08 | SBL | -0.095825 | -0.095825 | -0.095825 | 4.16E-17 | -0.095825 |
| | CBL | -0.095825 | -0.095825 | -0.095825 | 4.16E-17 | -0.095825 |
| g09 | SBL | 680.630057 | 680.630057 | 680.630057 | 2.21E-10 | 680.630057 |
| | CBL | 680.630057 | 680.630057 | 680.630057 | **5.68E-13** | 680.630057 |
| g10 | SBL | 7050.313660 | 7052.712113 | 7052.632264 | 1.187864 | 7056.516780 |
| | CBL | **7049.248021** | **7049.248063** | **7049.248279** | **6.15E-04** | **7049.250759** |
| g11 | SBL | 0.74990 | **0.749899** | 0.74990 | 1.17E-15 | 0.74990 |
| | CBL | 0.7499 | 0.7499 | **0.749899** | **1.11E-16** | 0.7499 |
| g12 | SBL | -1.0 | -1.0 | -1.0 | 0.0 | -1.0 |
| | CBL | -1.0 | -1.0 | -1.0 | 0.0 | -1.0 |
| g13 | SBL | 0.346487 | 0.851239 | 0.912186 | **0.153820** | 0.999510 |
| | CBL | **0.053941** | **0.438802** | **0.331740** | 0.162748 | **0.5868255** |
| g14 | SBL | -47.757573 | -47.632699 | -47.666634 | **0.112765** | **-47.074625** |
| | CBL | **-47.764888** | **-47.764845** | **-47.736939** | 0.128721 | -46.719956 |
| g15 | SBL | 961.715022 | 961.767409 | **961.715022** | 0.170656 | 962.834229 |
| | CBL | 961.715022 | **961.715022** | 961.722444 | **3.56E-03** | **961.734847** |
| g16 | SBL | -1.905155 | -1.905155 | -1.905155 | 1.77e-15 | -1.905155 |
| | CBL | -1.9051552 | -1.905155 | -1.905155 | **1.33E-15** | -1.905155 |
| g17 | SBL | 8854.615355 | 8949.925175 | 8950.020171 | 58.898262 | 9176.291456 |
| | CBL | **8853.533874** | **8913.737242** | **8899.960982** | **36.999861** | **8951.24573** |
| g18 | SBL | -0.864304 | -0.861477 | **-0.861748** | **1.83E-03** | **-0.856906** |
| | CBL | **-0.866025** | **-0.866025** | -0.8596571 | 3.42E-02 | -0.674981 |
| g19 | SBL | 35.864368 | 38.325524 | 38.312506 | 1.045887 | 41.130285 |
| | CBL | **32.655706** | **32.743630** | **32.809258** | **0.155246** | **33.206513** |
| g21 | SBL | 193.724531 | 193.724775 | 207.171716 | 35.419985 | 324.718339 |
| | CBL | **193.724510** | **193.7245210** | **204.80409** | **30.100436** | **324.702841** |
| g23 | SBL | -256.175340 | 41.072467 | 31.582880 | 152.277016 | 483.158431 |
| | CBL | **-400.0543932** | **-371.8092445** | **-364.1425693** | **31.567149** | **-288.070003** |
| g24 | SBL | -5.508013 | -5.508013 | -5.508013 | 8.88e-15 | -5.508013 |
| | CBL | -5.508013 | -5.508013 | -5.508013 | **8.88E-16** | -5.508013 |

Tabla 7.6: Resultados estadísticos que muestran la influencia que tiene un buscador local adaptado a un mecanismo de búsqueda global. *SBL* significa el algoritmo *AMER* sin el uso del buscador local y *CBL* se refiere a la propuesta original. Se utilizó la misma semilla de aleatorios para generar las soluciones.

| Prob. | Estadist | mejor | mediana | media | desv. est. | peor |
|---|---|---|---|---|---|---|
| g01 | factibles | 100 % | 100 % | 100 % | 0 | 100 % |
| | $eval_{min}$ | 998 | 1151 | 1155.09 | 53.56 | 1322 |
| | mejoras | 95.0 % | 94.0 % | 93.61 % | 0.7986 | 91.0 % |
| g02 | factibles | 100 % | 100 % | 100 % | 0 | 100 % |
| | $eval_{min}$ | 2500 | 2500 | 2500 | 0 | 2500 |
| | mejoras | 98.0 % | 97.0 % | 97.42 % | 0.5134 | 96.0 % |
| g03 | factibles | 100 % | 100 % | 99.2 % | 7.81 | 21.4 % |
| | $eval_{min}$ | 1494 | 2500 | 2400.03 | 209.33 | 2500 |
| | mejoras | 98.0 % | 82.0 % | 75.38 % | 18.21 | 0.0 % |
| g04 | factibles | 100 % | 100 % | 100 % | 0 | 100 % |
| | $eval_{min}$ | 700 | 777 | 775.51 | 27.40 | 843 |
| | mejoras | 90.0 % | 88.0 % | 87.6 % | 1.01 | 85.0 % |
| g05 | factibles | 100 % | 100 % | 100 % | 0 | 100 % |
| | $eval_{min}$ | 523 | 2500 | 2185.5 | 643.30 | 2500 |
| | mejoras | 85.0 % | 47.0 % | 53.24 % | 15.80 | 31.0 % |
| g06 | factibles | 100 % | 100 % | 100 % | 0 | 100 % |
| | $eval_{min}$ | 216 | 251 | 248.61 | 11.85 | 272 |
| | mejoras | 91.0 % | 89.0 % | 88.89 % | 0.85 | 87.0 % |
| g07 | factibles | 100 % | 100 % | 100 % | 0 | 100 % |
| | $eval_{min}$ | 1918 | 2500 | 2474.39 | 94.57 | 2500 |
| | mejoras | 97.0 % | 96.0 % | 95.57 % | 0.93 | 89.0 % |
| g08 | factibles | 100 % | 100 % | 100 % | 0 | 100 % |
| | $eval_{min}$ | 24 | 42 | 41.29 | 5.81 | 52 |
| | mejoras | 100.0 % | 99.0 % | 99.09 % | 0.2861 | 99.0 % |
| g09 | factibles | 100 % | 100 % | 100 % | 0 | 100 % |
| | $eval_{min}$ | 860 | 979 | 980.23 | 53.72 | 1096 |
| | mejoras | 98.0 % | 93.0 % | 91.75 % | 4.48 | 77.0 % |
| g10 | factibles | 100 % | 100 % | 100 % | 0 | 100 % |
| | $eval_{min}$ | 2429 | 2500 | 2499.29 | 7.06 | 2500 |
| | mejoras | 96.0 % | 94.0 % | 94.04 % | 0.7472 | 92.0 % |
| g11 | factibles | 100 % | 100 % | 100 % | 0 | 100 % |
| | $eval_{min}$ | 116 | 391 | 381.1 | 115.13 | 619 |
| | mejoras | 75.0 % | 70.0 % | 69.31 % | 3.20 | 59.0 % |

| Prob. | Estadist | mejor | mediana | media | desv. est. | peor |
|---|---|---|---|---|---|---|
| g12 | factibles | 100 % | 100 % | 100 % | 0 | 100 % |
| | $eval_{min}$ | 64 | 86 | 85.49 | 8.89 | 103 |
| | mejoras | 99.0 % | 99.0 % | 99 % | 0 | 99.0 % |
| g13 | factibles | 100 % | 100 % | 100 % | 0 | 100 % |
| | $eval_{min}$ | 1766 | 2500 | 2482.66 | 103.14 | 2500 |
| | mejoras | 87.0 % | 82.0 % | 81.53 % | 2.44 | 74.0 % |
| g14 | factibles | 100 % | 100 % | 100 % | 0 | 100 % |
| | $eval_{min}$ | 1591 | 2500 | 2478.23 | 112.08 | 2500 |
| | mejoras | 92.0 % | 89.0 % | 88.66 % | 1.11 | 86.0 % |
| g15 | factibles | 100 % | 100 % | 100 % | 0 | 100 % |
| | $eval_{min}$ | 279 | 727 | 1024.15 | 697.21 | 2500 |
| | mejoras | 92.0 % | 83.0 % | 81.75 % | 7.02 | 51.0 % |
| g16 | factibles | 100 % | 100 % | 100 % | 0 | 100 % |
| | $eval_{min}$ | 551 | 620 | 618.68 | 27.05 | 674 |
| | mejoras | 96.0 % | 94.0 % | 94.07 % | 0.73 | 92.0 % |
| g17 | factibles | 100 % | 100 % | 100 % | 0 | 100 % |
| | $eval_{min}$ | 2500 | 2500 | 2500 | 0 | 2500 |
| | mejoras | 87.0 % | 81.0 % | 81.09 % | 1.95 | 77.0 % |
| g18 | factibles | 100 % | 100 % | 100 % | 0 | 100 % |
| | $eval_{min}$ | 1174 | 1456 | 1508.45 | 277.57 | 2500 |
| | mejoras | 94.0 % | 92.0 % | 92.1 % | 0.7937 | 90.0 % |
| g19 | factibles | 100 % | 100 % | 100 % | 0 | 100 % |
| | $eval_{min}$ | 2500 | 2500 | 2500 | 0 | 2500 |
| | mejoras | 97.0 % | 96.0 % | 95.49 % | 0.6556 | 94.0 % |
| g21 | factibles | 100 % | 100 % | 100 % | 0 | 100 % |
| | $eval_{min}$ | 1807 | 2500 | 2346.73 | 200.98 | 2500 |
| | mejoras | 88.0 % | 84.0 % | 83.53 % | 1.81 | 78.0 % |
| g23 | factibles | 100 % | 100 % | 100 % | 0 | 100 % |
| | $eval_{min}$ | 2500 | 2500 | 2500 | 0 | 2500 |
| | mejoras | 87.0 % | 84.0 % | 83.63 % | 1.59 | 79.0 % |
| g24 | factibles | 100 % | 100 % | 100 % | 0 | 100 % |
| | $eval_{min}$ | 92 | 125 | 122.82 | 10.43 | 142 |
| | mejoras | 97.0 % | 96.0 % | 95.95 % | 0.7123 | 94.0 % |

Tabla 7.7: Resultados estadísticos de nuestra propuesta, se muestran el porcentaje de soluciones factibles, las evaluaciones mínimas que se requieren para converger al óptimo global de acuerdo a un valor de error porcentual y el porcentaje en mejoras que se obtienen después de usar el buscador local.

Capítulo 8

# Análisis estadístico

En una serie de experimentos podemos analizar e interpretar los datos estadísticos con la finalidad de tomar decisiones a partir de la información muestral de las poblaciones.

Para tomar decisiones es conveniente hacer determinados supuestos o conjeturas acerca de las poblaciones que se estudian. Tales supuestos que pueden ser o no ciertos se llaman hipótesis estadísticas y, en general, son sobre las distribuciones de probabilidad de las poblaciones.

Por ejemplo, si queremos decidir si un procedimiento es mejor que otro, se formula la hipótesis de que no hay diferencia entre los procedimientos. Dichas hipótesis son llamadas hipótesis nulas. Cualquier hipótesis que difiere de una hipótesis dada se llama hipótesis alternativa [11].

En otras palabras, se requiere de un estudio estadístico cuando en los experimentos evaluados se presentan frecuencias o repeticiones con ciertas características que requieren de un diagnóstico para corroborar hipótesis dadas.

En este capítulo presentamos dos métodos para el análisis estadístico, uno relacionado con el análisis de varianza (**ANOVA**) y otro que se enfoca al remuestreo de los datos estadísticos iniciales para establecer intervalos de confianza y para estimar la distribución de dichos datos (**BOOTSTRAP**).

## 8.1. Ajuste de parámetros de control

Todo análisis sigue una serie de pasos, como son: establecer perfiles, separar en grupos, determinar influencias entre varias variables, entre otros.

Cuando en un experimento se involucran diversas variables que están relacionadas, resulta importante modelar y explorar esta relación. Supongamos que hay una sola variable dependiente o de respuesta y que depende de $k$ variables independientes, por ejemplo $x_1, x_2, ..., x_k$. Es común que la relación entre estas variables se desconozca. Una alternativa que nos permite conocerla es el análisis de varianza, el cual es un método con una serie de experimentos que ayuda a determinar qué factores son importantes con dicha respuesta.

Los parámetros de control para nuestra propuesta son los que mencionamos en la sección 7.3 de esta tesis. Mediante la variación de cada uno de ellos, veremos un análisis de varianza en la respuesta (variable dependiente) como se describe a continuación.

### 8.1.1. Análisis de varianza (ANOVA)

En muchas situaciones existe la necesidad de conocer el significado de las diferencias entre tres o más medias muestrales, o lo que es equivalente a ensayar la hipótesis nula donde las medias muestrales son iguales.

Para llevar a cabo el análisis de varianza (ANOVA) desarrollamos un experimento de un factor donde se obtuvieron mediciones para $a$ grupos independientes de muestras, donde el número de medias en cada grupo es de $b$. Es decir, $a$ ejecuciones independientes con diferentes valores de parámetros, cada uno de las cuales tiene $b$ repeticiones o réplicas con diferente semilla para la generación de aleatorios.

Los resultados de este experimento pueden apreciarse en una tabla de la figura 8.1. En el esquema se presentan $a$ filas y $b$ columnas. Aquí $x_{jk}$ denota la media en la fila $j$ y la columna $k$, donde $j = 1, 2, ..., a; k = 1, 2, ..., b$

| ejecuciones | repeticiones | |
|---|---|---|
| ejecución 1 | $x_{11}\ x_{12}\ ...\ x_{1b}$ | $\bar{x}_1$ |
| ejecución 2 | $x_{21}\ x_{22}\ ...\ x_{2b}$ | $\bar{x}_2$ |
| . . . | . . . | |
| ejecución $a$ | $x_{a1}\ x_{a2}\ ...\ x_{ab}$ | $\bar{x}_a$ |

Figura 8.1: Tabla de experimentos para el análisis de varianza

Denotamos por $\bar{x}_j$ la media de las medidas de las fila $j$. Y tenemos que:

$$\bar{x}_j = \frac{1}{b} \sum_{k=1}^{b} x_{jk} \quad j = 1, 2, ..., a \tag{8.1}$$

Los valores $\bar{x}_j$ se denominan medias de grupo.

El método de ANOVA que implementamos fue para determinar la sensibilidad de los parámetros de control de nuestra propuesta sobre un subconjunto de las funciones de prueba.

De acuerdo a la experimentación y los intervalos establecidos para cada parámetro de nuestro algoritmo, observamos que ciertas funciones no presentaban variaciones significativas en la variable de respuesta (el valor de la función objetivo) mientras que otras sí presentaban una mayor sensibilidad o en algunos casos no llegaban al óptimo global.

Los problemas a estudiar mediante el método de ANOVA fueron: $g02$, $g03$, $g05$, $g07$, $g13$, $g19$, $g21$ y $g23$.

Los parámetros (factores) que analizamos fueron: $F$ (la probabilidad de mutación), $C$ (la probabilidad de recombinación), $\epsilon$ (la tasa de expansión o contracción para el simplex) $n_p$, (el número de soluciones para generar el vecindario), $Pf$ (la tasa para el manejo de restricciones), $pob$ (el número de individuos en la población) y $G_{max}$ (número máximo de generaciones).

Los niveles para cada parámetro se definieron como sigue:

- $F$: 0.8, 0.9, 0.99
- $C$: 0.6,0.8, 0.9
- $\epsilon$: 0.9, 1.0, 1.3
- $Pf$: 0.4, 0.45, 0.5
- $pob$: 60, 70, 80
- $n_p$: 3, 4
- $G_{max}$: 2500, 3000

Se realizaron 30 ejecuciones independientes para cada una de las 972 combinaciones de valores de parámetros y para cada uno de los 8 problemas, obteniendo un total de 233,280 ejecuciones de nuestro algoritmo.

El método de ANOVA que implementamos llevó a cabo los siguientes pasos:

- Por cada ejecución se obtuvo la diferencia entre el valor del óptimo global y el mejor valor encontrado de acuerdo a $|f(\vec{x}^*) - f(\vec{x})|$.
- Se obtuvo la media de cada diferencia obtenida por cada experimento (972 valores).
- De acuerdo a cada uno de los 8 problemas se identificaron los mejores valores de parámetros para alcanzar el óptimo global
- Se identificaron aquellos valores de los parámetros que permitieran obtener los mejores resultados en el conjunto completo de funciones de prueba.

Los resultados obtenidos prueban que no se cumple con la hipótesis nula, ya que las variaciones para cada experimento no presentan el mismo valor de media. Se concluye que del conjunto completo de funciones de prueba, aquellas que presentan mayor sensibilidad a la variación de parámetros son $g02$, $g03$, $g13$, $g21$ y $g23$. Además, calculamos el porcentaje de ejecuciones que logró converger al óptimo global, estableciéndose un 1 % de error porcentual e identificamos los mejores valores de los parámetros. En la tabla 8.1 presentamos dichos resultados.

Por último, identificamos los mejores valores de los parámetros que resolvían con una mayor precisión todo el conjunto de prueba. Estos valores fueron: $F = 0.8$, $C = 0.9$, $\epsilon = 0.9$, $n_p = 4$, $Pf = 0.45$, $pob = 80$, $G_{max} = 2500$.

| Función | F | C | $\epsilon$ | $n_p$ | Pf | pob | $G_{max}$ | $E_p$ |
|---|---|---|---|---|---|---|---|---|
| g02 | 0.4 | 0.6 | 0.99 | 80 | 4 | 1.0 | 3000 | 3.08 % |
| g03 | 0.5 | 0.9 | 0.99 | 80 | 4 | 1.0 | 3000 | 2.26 % |
| g05 | 0.4 | 0.9 | 0.8 | 80 | 4 | 0.9 | 2500 | 97.11 % |
| g07 | 0.5 | 0.9 | 0.8 | 80 | 4 | 0.9 | 3000 | 100.0 % |
| g13 | 0.5 | 0.8 | 0.9 | 80 | 4 | 1.3 | 3000 | 0.0 % |
| g19 | 0.4 | 0.9 | 0.8 | 80 | 4 | 1.0 | 3000 | 65.43 % |
| g21 | 0.45 | 0.9 | 0.8 | 70 | 3 | 1.0 | 3000 | 5.24 % |
| g23 | 0.45 | 0.9 | 0.8 | 80 | 4 | 0.9 | 3000 | 0.92 % |

Tabla 8.1: Resultados obtenidos a partir del método de ANOVA. $F$, $C$, $\epsilon$, $n_p$, $Pf$, $pob$, $G_{max}$ son los mejores valores para los parámetros de cada problema y $E_p$ es el porcentaje del promedio de ejecuciones exitosas estableciendo un 1 % de error porcentual.

## 8.2. Resultados obtenidos con ajuste de parámetros

A partir del ajuste de parámetros mediante el método de ANOVA, presentamos nuevamente las mismas estadísticas planteadas en la sección en 7.4.2. En la tabla 8.2 puede observarse una mejora en los problemas $g_{17}, g_{19}, g_{21}, g_{23}$. Sin embargo, empeoran los valores de la media y desviación estándar de los problemas $g_{02}, g_{03}, g_{05}$ aunque se mantienen constantes los resultados de los demás problemas. Lo anterior prueba que es difícil establecer un solo valor de los parámetros para resolver el conjunto completo de funciones, ya que mientras unos problemas se solucionan con un cierto valor en otros casos estos valores obtienen un desempeño pobre.

| Prob | mejor | mediana | media | desv. est. | peor |
|---|---|---|---|---|---|
| g01 | -15.0 | -15.0 | -15.0 | 0 | -15.0 |
| g02 | -0.8036188 | -0.7428514 | -0.731844215 | 5.85e-02 | -0.4875701 |
| g03 | -1.0005000 | -0.3163272 | -0.388349397 | 0.2275 | -0.1050999 |
| g04 | -30665.5386717 | -30665.5386717 | -30665.5386717 | 7.27e-12 | -30665.5386717 |
| g05 | 5126.496714 | 5126.496714 | 5127.2443680 | 5.8858 | 5183.251575 |
| g06 | -6961.8138755 | -6961.8138755 | -6961.81387558 | 5.45e-12 | -6961.8138755 |
| g07 | 24.3062090 | 24.3062090 | 24.306209218 | 7.22e-07 | 24.3062153 |
| g08 | -0.0958250 | -0.0958250 | -0.0958250 | 4.16e-17 | -0.0958250 |
| g09 | 680.6300573 | 680.6300573 | 680.630057378 | 6.17e-10 | 680.6300573 |
| g10 | 7049.2480205875 | 7049.2480237968 | 7049.24802589383 | 7.51e-06 | 7049.2480798153 |
| g11 | 0.7499 | 0.7499 | 0.74989999 | 1.77e-15 | 0.7499 |
| g12 | -1 | -1 | -1 | 0 | -1 |
| g13 | 0.05394151 | 0.43880260 | 0.388846572 | 0.1648 | 0.85612552 |
| g14 | -47.7648884 | -47.7648884 | -47.76488841 | 2.52e-07 | -47.7648863 |
| g15 | 961.71502229 | 961.71502229 | 961.7156441234 | 0.004e-03 | 961.75475784 |
| g16 | -1.90515525 | -1.90515525 | -1.90515525 | 1.77e-15 | -1.90515525 |
| g17 | 8853.53847439 | 8861.63800765 | 8894.122487980 | 39.6332 | 8957.63249965 |
| g18 | -0.86602540 | -0.86602540 | -0.8660254036 | 2.81e-10 | -0.86602540 |
| g19 | 32.65559426 | 32.65639423 | 32.6637106789 | 2.81e-02 | 32.87554421 |
| g21 | 193.7245100 | 193.7245101 | 201.968307836 | 24.2364 | 290.1950741 |
| g23 | -400.0550365 | -398.0397053 | -393.253349847 | 8.7068 | -363.9486188 |
| g24 | -5.5080132 | -5.5080132 | -5.5080132716 | 8.88e-15 | -5.5080132 |

Tabla 8.2: Resultados estadísticos del conjunto de funciones después del ajuste de parámetros.

## 8.3. Método de remuestreo (BOOTSTRAP)

El método de Bootstrap surge por primera vez en una publicación de Bradley Efron en 1979 [19], donde introduce una metodología para estimar las distribuciones de algunas estadísticas cuando el tamaño muestral es pequeño o las expresiones de dichas distribuciones son analíticamente intratables. Desde entonces numerosos autores han desarrollado métodos de *bootstrap* para diversos procedimientos inferenciales, tales como modelos de regresión, datos censurados, construcción de intervalos de confianza y estimación de parámetros, entre otros.

Los métodos de *bootstrap* se basan en la reproducción de los datos originales mediante un remuestreo. Si nuestras observaciones tienen una estructura de dependencia, está debe estar reflejada en los nuevos datos. Este método de remuestreo se usa para estimar o aproximar la distribución muestral de una estadística, basado en los principios de sustitución y aproximación numérica.

Supongamos que los valores de un muestreo son el resultado de variables aleatorias distribuidas idénticamente e independientemente $Y_1, \ldots, Y_n$ las cuales son denotadas por $f$ y $F$, que corresponde a la función de densidad de probabilidad y a la función de densidad acumulada, respectivamente.

El muestreo es utilizado para hacer inferencia acerca de las características de la población, generalmente denotada por $\theta$ usando una estadística $T$ la cual tiene el valor $t$ en el muestreo. Asumimos que en el momento de escoger $T$ ésta debe ser una estimación para $\theta$ [14].
Las observaciones se centran en la probabilidad de la distribución de $T$. Por ejemplo, cuál es la tendencia (sesgo), cuál el error estándar, cuáles son las distancias intercuartiles y cuáles son los valores probables bajo una cierta hipótesis nula de interés.

Cuando no hay un modelo matemático disponible, el análisis estadístico es no paramétrico y usa sólo el hecho de que las variables aleatorias $Y_j$ son independientes e igualmente distribuidas. De hecho, incluso si hay un modelo paramétrico plausible, un análisis no paramétrico puede ser útil para evaluar la solidez de las conclusiones extraídas de un análisis paramétrico.

El algoritmo 23 muestra el procedimiento básico de *bootstrap* para $K$ remuestreos, empleando una muestra $S$ de tamaño $n$ con la cual se busca una distribución muestral para un estimador $\theta^*$ del parámetro de la población $\theta$. Este un procedimiento que nos permite determinar distribuciones para varias estadísticas, por ejemplo: la media, la varianza, la distancia inter-cuartil, etc.

**Algoritmo 23**: Algoritmo básico de bootstrap

```
1 begin
2   for i ← 1, ..., K do
3     Generar una remuestra S_i^* de la muestra original de S con reemplazo, igual
      probabilidad y mismo tamaño. Calcular el valor θ_i^* de la estadística de S_i^*
4   Los valores θ_1^*, ..., θ_K^* componen la distribución muestral de bootstrap
5 end
```

Nuestro estudio se centró en un procedimiento de *bootstrap* básico con 1000 remuestreos, a partir de un conjunto de 100 muestras independientes.

El procedimiento de *bootstrap* se llevó a cabo usando el lenguaje de programación R[1] en su versión 2.12.0 (para Mac OS X 10.5, Leopard).

### 8.3.1. Intervalos de confianza

Una vez que se prueba si la hipótesis nula $\mu_1 = \mu_2$ se cumple, suele ser más interesante conocer la diferencia entre esas medias, lo que se refiere a proporcionar un intervalo que se espera contenga el valor del parámetro o de los parámetros bajo estudio. Estos intervalos se conocen como **intervalos de confianza**.

[1]Desarrollado por Robert Gentleman y Ross Ihaka del Departamento de Estadística de la Universidad de Auckland en 1993

Con el objeto de definir un intervalo de confianza supongamos que $\theta$ es un parámetro desconocido. Para obtener una estimación del intervalo de $\theta$ se necesita la determinación de dos estadísticas $L$ y $U$ de manera que se cumpla la probabilidad de la ecuación (8.2)

$$P(L \leqslant \theta \leqslant U) = 1 - \alpha \tag{8.2}$$

$$L \leqslant \theta \leqslant U \tag{8.3}$$

El intervalo de la ecuación (8.3) se denomina **intervalo de confianza** de $100(1-\alpha)$ por ciento para el parámetro $\theta$. Para interpretar este intervalo debe tomarse en cuenta que si en muestras aleatorias repetidas, se considera un gran número de estos intervalos, el $100(1-\alpha)$ por ciento de ellos contendrá el valor verdadero de $\theta$. Las estadísticas $L$ y $U$ se conocen como los límites superior e inferior de confianza, respectivamente, y $1-\alpha$ se denomina coeficiente de confianza. La ecuación (8.2) se llama intervalo de confianza del 95 % para $\theta$ si $\alpha = 0.05$. Hay que notar que los intervalos de confianza poseen una interpretación de frecuencia.

En la figura 8.2 se ilustra el cálculo de los intervalos de confianza.

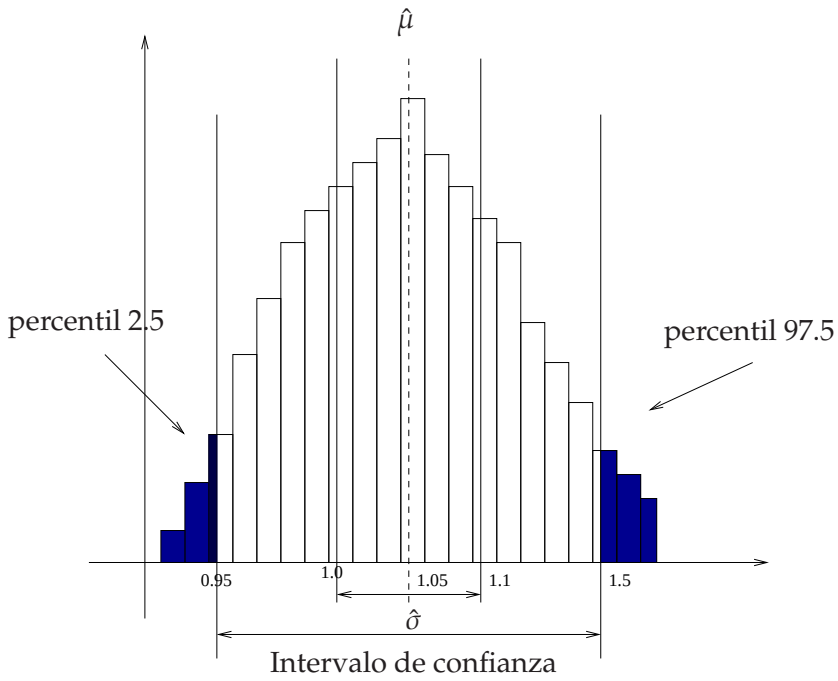



Figura 8.2: Distribución de *bootstrap*

Los pasos que sigue el método de *bootstrap* para el cálculo de intervalos de confianza son:

1. Construir una distribución a partir de $K$ muestras de *bootstrap* para una estadística $u$.
2. Ordenar los valores en la distribución.

3. El límite inferior de el $100(1-2\alpha)$ porciento de intervalos de confianza es el $K\alpha-$ésimo valor, y el límite superior es el $K(1-\alpha)-$ésimo valor en la distribución ordenada.

Las tablas 8.3 y 8.4 presentan los intervalos de confianza de las 22 funciones de prueba para cada uno de los algoritmos analizados en esta tesis. Se resaltan en negritas aquellos intervalos que contienen el valor óptimo de cada problema.

| Función | Óptimo | SR | MCODE | LEDE1 |
|---|---|---|---|---|
| g01 | -15 | **(-15, -15 )** | (-15.00, -14.89 ) | (-14.57, -14.20 ) |
| g02 | -0.8036191 | (-0.7210, -0.7094) | (-0.7448, -0.7236 ) | (-0.6274, -0.5864 ) |
| g03 | -1.00050 | (-0.999479,-0.999479) | (-0.4160, -0.3311 ) | **(-1.0005, -1.0005 )** |
| g04 | -30665.538 | **(-30665.54, -30665.54)** | **(-30665.54, -30665.54 )** | **(-30665.54 , -30665.54 )** |
| g05 | 5126.4967 | **(5126.497,5126.497)** | **(5126, 5127 )** | **(5126.56, 5127 )** |
| g06 | -6961.8138 | (-6961.814,-6961.814) | (-6963, -6959 ) | (-6962, -6961.81 ) |
| g07 | 24.306209 | (24.31, 24.32 ) | **(24.30, 24.32 )** | **(24.30, 24.31 )** |
| g08 | -0.0958250 | **(-0.095825, -0.095825)** | **(-0.095825, -0.095825 )** | **(-0.095825, -0.095825 )** |
| g09 | 680.63005 | **(680.6311,680.6311)** | **(680.63, 680.63 )** | **(680.63,680.63 )** |
| g10 | 7049.248 | (7055, 7079 ) | **(7049, 7051 )** | **(7049.248, 7049.248 )** |
| g11 | 0.7499 | **( 0.7499, 0.7499 )** | ( 0.7487, 0.7561 ) | **( 0.7499, 0.7499 )** |
| g12 | -1 | ( -0.9998, -0.9997 ) | **(-1, -1 )** | **(-1, -1 )** |
| g13 | 0.0539415 | **( 0.0539, 0.0543 )** | ( 0.2275, 0.3052 ) | ( 0.3048, 0.3803 ) |
| g14 | -47.764888 | (-47.73, -47.70 ) | (-47.63, -47.50 ) | (-47.73, -47.68 ) |
| g15 | 961.71502 | (961.752, 961.752 ) | (961.7263, 961.7263) | (962.3, 962.8 ) |
| g16 | -1.9051552 | (-1.2833, -0.3152 ) | **(-1.905155, -1.905155 )** | **(-1.905155 , -1.905155 )** |
| g17 | 8853.5396 | (9036, 9041 ) | (8887, 8904 ) | (8911, 8939 ) |
| g18 | -0.8660254 | (-0.866021, -0.866021 ) | (-0.8259, -0.7915 ) | (-0.8277, -0.7918 ) |
| g19 | 32.655592 | (33.07, 33.17 ) | (32.68, 32.73 ) | (32.82, 32.95 ) |
| g21 | 193.72451 | ———— | (220.8, 241.9 ) | (218.8, 240.5 ) |
| g23 | -400.0551 | ———— | (-328.8, -287.9 ) | (-267.0, -210.9 ) |
| g24 | -5.508013 | **(-5.508013, -5.508013 )** | **(-5.508013 , -5.508013 )** | **(-5.508013 , -5.508013 )** |

Tabla 8.3: Intervalos de confianza de las 22 funciones de prueba para los algoritmos SR (jerarquización estocástica), MCODE (evolución diferencial con cooperación co-evolutiva) y LEDE1 (exploración local basada en evolución diferencial en su versión 1).

En la tabla 8.5 se muestra para cada una de las funciones, la media y desviación estándar de nuestra propuesta, después de aplicar el procedimiento de *bootstrap* para 1000 remuestreos y un intervalo de confianza de 95 %.

La distribución muestral de *bootstrap* puede visualizarse en histogramas de frecuencia y gráficas de cuantil que se ajustan a una distribución normal y que establecen el intervalo de confianza resultado de este método (ver apéndice B.5). La tabla 8.6 muestra estos gráficos para las funciones $g10$ y $g12$, donde puede apreciarse un ajuste a una distribución normal a diferencia de los histogramas presentados en el apéndice B.4, esto para el primer caso. Se hace notar que en $g12$, nuestros datos estadísticos no presentan una variación significativa mostrando un valor constante en las gráficas.

| Función | Óptimo | LEDE2 | AMER |
|---|---|---|---|
| g01 | -15 | (-14.41, -14.00 ) | **(-15, -15 )** |
| g02 | -0.8036191 | (-0.6270, -0.5855 ) | (-0.7432, -0.7210 ) |
| g03 | -1.00050 | (-1.0024, -0.9966 ) | (-0.4318, -0.3409 ) |
| g04 | -30665.538 | **(-30665.54 , -30665.54 )** | **(-30665.54 , -30665.54 )** |
| g05 | 5126.4967 | (4202, 4600 ) | **(5126, 5128 )** |
| g06 | -6961.8138 | **(-6961.814, -6961.814 )** | **(-6961.814, -6961.814 )** |
| g07 | 24.306209 | **(24.306, 24.31 )** | **(24.30622, 24.30622 )** |
| g08 | **-0.0958250** | **(-0.09582504, -0.09582504 )** | **(-0.09582504, -0.09582504 )** |
| g09 | **680.63005** | **(680.63,680.63 )** | **(680.63,680.63 )** |
| g10 | 7049.248 | **(7049.248, 7049.248 )** | **(7049.248, 7049.248 )** |
| g11 | 0.7499 | **( 0.7499, 0.7499 )** | **( 0.7499, 0.7499 )** |
| g12 | -1 | **(-1, -1 )** | **(-1, -1 )** |
| g13 | 0.0539415 | ———— | ( 0.3559, 0.4201 ) |
| g14 | -47.764888 | (-47.60, -47.29 ) | **(-47.76486, -47.76 )** |
| g15 | 961.71502 | (962.3, 962.8 ) | **(961.715, 961.715 )** |
| g16 | -1.9051552 | **(-1.905155 , -1.905155 )** | **(-1.905155, -1.905155 )** |
| g17 | 8853.5396 | (5601, 6395 ) | (8887, 8901 ) |
| g18 | -0.8660254 | (-0.8259, -0.7915 ) | **(-0.8660254, -0.866 )** |
| g19 | 32.655592 | (32.82, 32.91 ) | **(32.66, 32.67 )** |
| g21 | 193.72451 | (243.1, 265.1 ) | (196-7, 206.4 ) |
| g23 | -400.0551 | (-248.1, -174.6 ) | (-395.1, -391.7 ) |
| g24 | -5.508013 | **(-5.508013 , -5.508013 )** | **(-5.508013 , -5.508013 )** |

Tabla 8.4: Intervalos de confianza de las 22 funciones de prueba para los algoritmos LEDE2 (exploración local basada en evolución diferencial en su versión 2) y AMER (algoritmo memético para espacios restringidos).

| Prob. | Estadist | SR | MCODE | LEDE1 | LEDE2 | AMER |
|---|---|---|---|---|---|---|
| g01 | media | **-15** | -14.9414 | -14.3768 | -14.2025 | **-15** |
| -15.0 | desv.est. | **0** | 0.0256 | 0.0949 | 0.1021 | **0** |
| g02 | media | -0.715428 | **-0.733692** | -0.606603 | -0.606603 | -0.7318442 |
| -0.803619 | desv.est. | **0.0029** | 0.0054 | 0.0104 | 0.0105 | 0.0056 |
| g03 | media | -0.999479 | -0.375746 | **-1.0005** | -0.998551 | -0.388394 |
| -1.0005001 | desv.est. | **0** | 0.0214 | 9.98E-13 | 0.0019 | 0.0228 |
| g04 | media | **-30665.54** | **-30665.54** | **-30665.54** | **-30665.54** | **-30665.54** |
| -30665.538671 | desv.est. | **0** | **0** | **0** | **0** | **0** |
| g05 | media | **5126.497** | 5126.675 | 5126.566 | 4385.473 | 5127.244 |
| 5126.496714 | desv.est. | **5.23E-06** | 0.1742 | 0.0242 | 105.2458 | 0.0283 |
| g06 | media | **-6961.814** | -6960.597 | **-6961.814** | **-6961.814** | **-6961.814** |
| -6961.813875 | desv.est. | **0** | 1.2287 | **0** | **0** | **0** |
| g07 | media | 24.31265 | 24.31112 | **24.30621** | **24.30621** | **24.30621** |
| 24.306209 | desv.est. | 0.0021 | 0.0043 | 7.27E-08 | 9.00E-08 | **7.23E-08** |
| g08 | media | **-0.095825** | **-0.095825** | **-0.095825** | **-0.095825** | **-0.095825** |
| -0.0958250415 | desv.est. | **0** | **0** | **0** | **0** | **0** |
| g09 | media | 680.6311 | 680.63 | **680.63** | **680.63** | **680.63** |
| 680.630057374 | desv.est. | 0.0005 | 8.18E-08 | **0** | **0** | **0** |
| g10 | media | 7069.193 | 7050.033 | 7049.248 | 7049.248 | **7049.248** |
| 7049.24802052 | desv.est. | 6.5266 | 0.4971 | 0.0001 | 1.06E-05 | **7.83E-07** |
| g11 | media | 0.7499 | 0.7531254 | **0.7499** | 0.**7499** | **0.7499** |
| 0.7499 | desv.est. | 2.75E-09 | 0.0020 | **0** | **0** | **0** |
| g12 | media | -0.999762 | **-1** | **-1** | **-1** | **-1** |
| -1.0 | desv.est. | 1.46E-05 | **0** | **0** | **0** | **0** |
| g13 | media | **0.05413312** | 0.2662885 | 0.3451254 | —— | 0.388846 |
| 0.053941514 | desv.est. | **0.0001** | 0.0194 | 0.0194 | —— | 0.01611 |
| g14 | media | -47.71481 | -47.56318 | -47.69823 | -47.43216 | **-47.76489** |
| -47.76488845 | desv.est. | 0.0088 | 0.0348 | 0.0126 | 0.0782 | **2.46E-08** |
| g15 | media | 961.7152 | 961.7263 | 962.5255 | 962.5152 | **961.715** |
| 961.7150222899 | desv.est. | 0.0001 | 0.0072 | 0.1242 | 0.1273 | **4.41E-04** |
| g16 | media | -0.637914 | **-1.905155** | **-1.905155** | **-1.905155** | **-1.905155** |
| -1.9051552586 | desv.est. | 0.3265 | **0** | **0** | **0** | **0** |
| g17 | media | 9039.09 | 8895.791 | 8926.139 | 5995.146 | **8894.122** |
| 8853.5396748 | desv.est. | **1.4717** | 4.1429 | 7.3410 | 201.6275 | 3.7993 |
| g18 | media | -0.866021 | -0.8086971 | -0.8088725 | -0.8087107 | **-0.8660254** |
| -0.8660254038 | desv.est. | 8.37E-07 | 0.0088 | 0.0090 | 0.0088 | **2.74E-11** |
| g19 | media | 33.1182 | 32.70564 | 32.88967 | 32.86238 | **32.66371** |
| 32.6555929502 | desv.est. | 0.0261 | 0.0112 | 0.0327 | 0.0220 | **0.0028** |
| g21 | media | —— | 231.2671 | 229.726 | 253.9544 | **201.9683** |
| 193.72451007 | desv.est. | —— | 5.3602 | 5.6394 | 5.6439 | **2.4197** |
| g23 | media | —— | -307.3927 | -239.6527 | -209.9973 | **-393.2533** |
| -400.0551000 | desv.est. | —— | 10.57446 | 14.72521 | 18.42314 | **0.8475** |
| g24 | media | **-5.508013** | **-5.508013** | **-5.508013** | **-5.508013** | **-5.508013** |
| -5.5080132716 | desv.est. | **0** | **0** | **0** | **0** | **0** |

Tabla 8.5: Media y desviación estándar de la función objetivo de nuestra propuesta, calculada a través de un procedimiento de *bootstrap* con 1000 remuestreos.

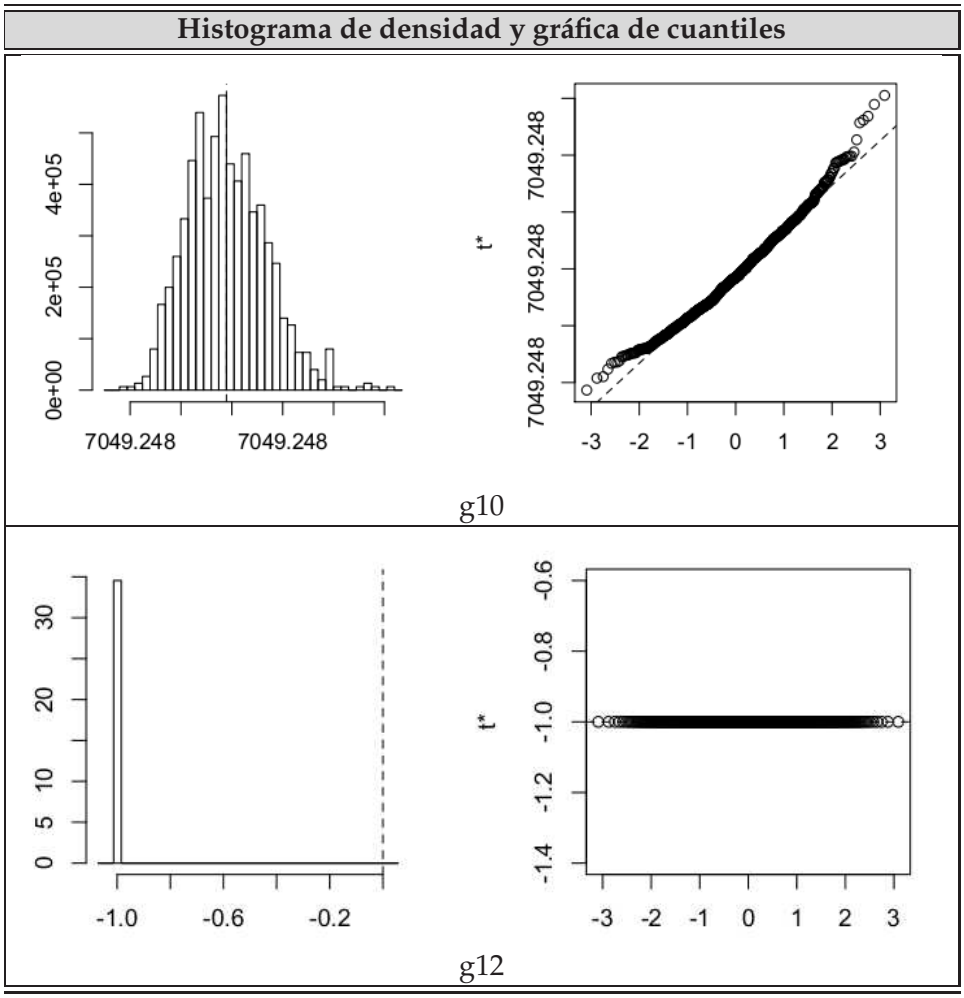



Tabla 8.6: Histograma de densidad y gráficas de cuantiles de las soluciones después de aplicar método de *bootstrap* con 1000 remuestreos para las funciones $g10$ *y* $g12$

Capítulo 9

# Conclusiones y trabajo futuro

## 9.1. Conclusiones

Los algoritmos evolutivos (AEs) son procedimientos iterativos de búsqueda, que surgieron como una alternativa para dar solución a problemas de optimización que comúnmente son intratables mediante métodos matemáticos clásicos. Estos algoritmos exploran de forma estratégica el espacio de búsqueda con el fin de encontrar soluciones óptimas globales. El espacio de búsqueda de las soluciones puede ser discreto, continuo, multidimensional o estar restringido por otras funciones. Buscar buenas soluciones en espacios con estas características implica utilizar técnicas más eficientes de exploración.

La solución de problemas definidos en espacios restringidos requiere incorporar al AE algún mecanismo para el manejo de restricciones. En esta tesis se abordaron diversas técnicas, siendo la más competitiva la denominada jerarquías estocásticas (JE), la cual establece un método de ordenamiento de las soluciones de acuerdo a un balance que da prioridad a minimizar la función objetivo o minimizar el grado de violación de las funciones de restricción. Además, esta técnica no incorpora coeficientes de penalización dependientes del problema lo cual facilita su uso.

El propósito de cualquier AE es incrementar la velocidad de convergencia hacia las soluciones óptimas globales, es decir, disminuir el tiempo de cómputo sea mínimo. Para obtener información del costo computacional, se utiliza como medida estándar el número de evaluaciones de la función objetivo (EFO). Esta medida fue empleada para la comparación de esta propuesta y los algoritmos estudiados de la literatura, siendo el algoritmo presentado en esta tesis el que utiliza el menor número de EFO (180,000 y 205,000 en el caso de ajuste de parámetros).

Algunas propuestas que intentan acelerar la convergencia hacia óptimos globales, son los métodos llamados algoritmos meméticos (AMs), cuya idea se basa en la combinación de mecanismos de búsqueda global y búsqueda local. Mientras el primer mecanismo explora todo el espacio de búsqueda del problema el otro mecanismo explota una determinada región, denominada vecindario, con la finalidad de refinar las soluciones. Sin embargo estas propuestas tienden a converger a óptimos locales, por lo que es indispensable mantener un correcto balance entre ambos mecanismos.

Uno de los procedimientos de búsqueda global estudiado y elegido para esta propuesta es el algoritmo de evolución diferencial (ED), que ha mostrado ser robusto en la solución de una amplia gama de problemas, maneja codificación real y sólo usa dos parámetros de control. Realiza la diferencia entre vectores para perturbar un tercer vector de la población.

Esta diferencia es escalada por un factor, y el vector final permite guiar la búsqueda en una determinada dirección.

Se presentaron buscadores locales basados en métodos de búsqueda directa e indirecta y en heurísticas y operadores genéticos basados en vecindarios, siendo esta última categoría donde recae el operador simplex SPX que usa esta propuesta. Este operador genera un vecindario a partir de la construcción de un $n-$simplex, donde $n$ es un determinado número de soluciones. El algoritmo selecciona estratégicamente estas soluciones de acuerdo al ordenamiento que proporciona JE, escogiendo a los mejores y peores individuos del ordenamiento. En otras palabras, explota los vecindarios de las mejores y peores soluciones, con la finalidad de encontrar una nueva solución que minimice el valor de la función objetivo y el valor de la función de penalización.

Se estudió detalladamente el artículo de Noman e Iba [66]. Ellos propusieron un algoritmo híbrido de ED y SPX para problemas de optimización mono-objetivo sin restricciones. Se corroboró que esta propuesta acelera la convergencia de ED hasta en un 80 %, además de mantener esta eficacia aún escalando las funciones de prueba. Por lo tanto, se planteó una primera hipótesis al diseñar e implementar un algoritmo memético que usara estos mismos mecanismos y extrapolara este comportamiento de convergencia a problemas mono-objetivo con restricciones.

Se empleó un conjunto de 22 problemas de optimización con restricciones, los cuales presentaban diversas características, por ejemplo: tipos de funciones (cuadrática, lineal, no lineal, polinomial, cúbica), número de variables, número de funciones de igualdad y desigualdad, o diferente tasa entre la región factible y el espacio de búsqueda. Esta diversidad de características crea un universo de espacios de búsqueda, lo cual complica que un algoritmo resuelva todas las funciones.

En resumen, la propuesta de AM para espacios restringidos se compone de un mecanismo de búsqueda global basado en ED en su versión *rand/1/bin*; un método para el manejo de restricciones, JE, que ordena las soluciones de acuerdo a la minimización de la función objetivo y la minimización de la función de penalización; un procedimiento de búsqueda local basado en el operador de cruza SPX, el cual explota dos vecindarios definidos por las $n$ mejores y las $n$ peores soluciones de cada generación. En cada iteración estos tres mecanismos son ejecutados.

La estrategia de reemplazo usada para ED y SPX se estableció de acuerdo a tres reglas: si se tienen dos soluciones, una factible y otra infactible, se tiene preferencia por aquella que es factible; si las dos son infactibles, se selecciona aquella que viole en menor grado las funciones de restricción y si ambas son soluciones factibles se prefiere a la de menor valor de la función objetivo.

Para comprobar la eficacia de la propuesta se realizó un estudio comparativo con cuatro algoritmos representativos del estados del arte tomados de la literatura especializada. Estos algoritmos fueron jerarquías estocásticas (SR) y los AMs: evolución diferencial con

cooperación co-evolutiva (MCODE) y exploración local basada en evolución diferencial (LEDE), en sus dos versiones. Para SR se usó la versión en línea del autor, desarrollada en el software Matlab. Éste sólo tenía 13 de las 22 funciones empleadas, por lo que se implementaron las demás. Los algoritmos MCODE y LEDE fueron implementados en lenguaje C.

Se realizaron una serie de experimentos con el fin de validar el rendimiento de la propuesta y los otros algoritmos con respecto a los cuales se comparan resultados. Los experimentos consistieron en realizar 100 ejecuciones independientes (diferente semilla) para cada una de las funciones del conjunto de problemas.

De esta comparación se observa que el algoritmo SR en ciertos problemas queda estancado en óptimos locales o no encuentra soluciones factibles, como fue el caso de los problemas $g16$, $g21$ *y* $g23$. Los algoritmos MCODE y LEDE, por su parte, presentaban una convergencia rápida hacia el óptimo global, lo cual en ciertos casos producía pérdida de diversidad en las soluciones provocando una convergencia prematura a óptimos locales. En el caso de $g17$, LEDE divergía.

Se realizó un análisis estadístico con las medidas de tendencia central que se incluyen en el cálculo de la media, la mediana, la desviación estándar y el valor mínimo y máximo de los conjuntos de datos estadísticos obtenidos. También se estudió el efecto que producía el uso del buscador local en la propuesta.

Se utilizaron gráficos de caja e histogramas de frecuencia en las soluciones, con el objetivo de apreciar la distribución de los datos. Además, se obtuvieron gráficas de convergencia para comparar el comportamiento entre los algoritmos, así como del efecto que produce el uso del buscador local en la propuesta. Puede observarse en las gráficas de convergencia cómo se aceleran cada uno de los métodos a partir del uso de un buscador local. Sin embargo, puede verse que usar este tipo de procedimientos hace caer fácilmente en óptimos locales. Esto se consideró evidentemente como una desventaja para encontrar la solución óptima global.

De acuerdo a los resultados presentados en la sección 7.4.1, en la p.g. 93, se observa que la propuesta es consistente en la solución de los diversos problemas. Sin embargo, presenta ciertos problemas para determinadas funciones. Por lo tanto, se desarrollaron procedimientos de cómputo intensivo. Uno de éstos fue el método de análisis de varianza ANOVA que permitió obtener un ajuste de los parámetros y proporcionó información acerca de la influencia que tiene cada uno de los parámetros de la propuesta en su desempeño al resolver cada problema.

Se aplicó el método de *bootstrap*, el cual permitió obtener aproximaciones a distribuciones muestrales, establecer los intervalos de confianza con un 5 % de error y conocer a partir de 1000 remuestreos la media y desviación estándar de los datos estadísticos.

De los resultados experimentales se concluye que la propuesta es robusta al obtener claramente mejores resultados en todo el conjunto de problemas.

## 9.2. Trabajo a futuro

Como parte del trabajo a futuro de esta tesis se contemplan diversos aspectos detallados a continuación:

- Explorar otros mecanismos de búsqueda global, (p. ej. el algoritmo de cúmulos de partículas) y observar su comportamiento al hibridizarse con el buscador local SPX.
- Realizar un estudio a fondo de la influencia del factor de expansión y contracción el buscador local SPX.
- Examinar la resolución de otros conjuntos de funciones de prueba para optimización con restricciones e identificar en qué tipo de problemas es buena la propuesta.
- Presentar un estudio detallado de las características que hacen que un determinado buscador local sea mejor que otros en algunos problemas.
- Acoplar algún mecanismo autoadaptativo que ejecute el mecanismo de búsqueda local para determinadas épocas, es decir una determinada frecuencia de uso.
- Autoadaptar los parámetros que presentan mayor sensibilidad en la respuesta de la solución de los problemas para que se encuentren las soluciones óptimas globales del conjunto completo de funciones.
- Extender la propuesta del AM que combina ED y SPX a problemas de optimización multiobjetivo.

# Funciones de prueba

**g01**

Minimizar $f(\vec{x}) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=5}^{13} x_i^2 - \sum_{i=15}^{13} x_i$ sujeto a:

$$g_1(\vec{x}) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leqslant 0$$
$$g_2(\vec{x}) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leqslant 0$$
$$g_3(\vec{x}) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leqslant 0$$
$$g_4(\vec{x}) = -8x_1 + x_{10} \leqslant 0$$
$$g_5(\vec{x}) = -8x_2 + x_{11} \leqslant 0$$
$$g_6(\vec{x}) = -8x_3 + x_{12} \leqslant 0$$
$$g_7(\vec{x}) = -2x_4 - x_5 + x_{10} \leqslant 0$$
$$g_8(\vec{x}) = -2x_6 - x_7 + x_{11} \leqslant 0$$
$$g_9(\vec{x}) = -2x_8 - x_9 + x_{12} \leqslant 0$$

donde los límites de las variables son: $0 \leqslant x_i \leqslant 1$ $(i = 1, ..., 9), 0 \leqslant x_i \leqslant 100$ $(i = 10, 11, 12)$ y $0 \leqslant x_{13} \leqslant 1$. El óptimo global es $\vec{x}^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$ donde $f(\vec{x}^*) = -15$. Las restricciones $g_1, g_2, g_3, g_7, g_8$ y $g_9$ están activas, en el óptimo.

**g02**

Minimizar $f(\vec{x}) = -\left| \dfrac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right|$ sujeto a:

$$g_1(\vec{x}) = 0.75 - \prod_{i=1}^{n} x_i \leqslant 0$$
$$g_2(\vec{x}) = \sum_{i=1}^{n} x_i - 7.5n \leqslant 0$$

donde $n = 20$ y $0 \leqslant x_i \leqslant 10$ $(i = 1, ..., n)$. El óptimo global es $\vec{x}^* = (3.16246061572185, 3.12833142812967, 3.09479212988791, 3.06145059523469, 3.02792915885555, 2.99382606701730, 2.95866871765285, 2.92184227312450, 0.49482511456933, 0.48835711005490, 0.48231642711865,$

0.47664475092742, 0.47129550835493, 0.46623099264167, 0.46142004984199, 0.45683664767217, 0.45245876903267, 0.44826762241853, 0.44424700958760, 0.44038285956317) , y el $f(\vec{x}^*) = -0.80361910412559$. La restricción $g_1$ está casi activa ($g_1 = -10^{-8}$).

**g03**

Minimizar $f(\vec{x}) = -(\sqrt{n})^n \prod_{i=1}^{n} x_i$ sujeto a:

$$h_1(\vec{x}) = \sum_{i=1}^{n} x_i^2 - 1 = 0$$

donde $n = 10$ y $0 \leqslant x_i \leqslant 1$ $(i = 1, ...., n)$. El óptimo global es $\vec{x}^* = (0.31624357647283069, 0.31624357741433839, 0.31624357801234592, 0.31624357566401789, 0.31624357820552606, 0.31624357382885506, 0.31624357573748291, 0.31624357615737491)$, y el $f(\vec{x}^*) = -1.00050010001000$.

**g04**

Minimizar $f(\vec{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$ sujeto a:

$$g_1(\vec{x}) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leqslant 0$$

$$g_2(\vec{x}) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leqslant 0$$

$$g_3(\vec{x}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leqslant 0$$

$$g_4(\vec{x}) = -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leqslant 0$$

$$g_5(\vec{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leqslant 0$$

$$g_6(\vec{x}) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leqslant 0$$

donde $78 \leqslant x_1 \leqslant 102, 33 \leqslant x_2 \leqslant 45$ y $27 \leqslant x_i \leqslant 45$ $(i = 3, 4, 5)$. La solución óptima es $\vec{x}^* = (78, 33, 29.9952560256815985, 45, 36.7758129057882073)$ y $f(\vec{x}^*) = -30665.53867178332$. Dos restricciones ($g1$ y $g6$) están activas, en el óptimo.

**g05**

Minimizar $f(\vec{x}) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3$ sujeto a:

$$g_1(\vec{x}) = -x_4 + x_3 - 0.55 \leqslant 0$$

$$g_2(\vec{x}) = -x_3 + x_4 - 0.55 \leqslant 0$$

$$h_3(\vec{x}) = 1000 \operatorname{sen}(-x_3 - 0.25) + 1000 \operatorname{sen}(-x_4 - 0.25) + 894.8 - x_1 = 0$$

$$h_4(\vec{x}) = 1000 \operatorname{sen}(x_3 - 0.25) + 1000 \operatorname{sen}(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$$

$$h_5(\vec{x}) = 1000 \operatorname{sen}(x_4 - 0.25) + 1000 \operatorname{sen}(x_4 - x_3 - 0.25) + 1294.8 = 0$$

donde $0 \leqslant x_1 \leqslant 1200$, $0 \leqslant x_2 \leqslant 1200$, $-0.55 \leqslant x_3 \leqslant 0.55$ y $-0.55 \leqslant x_4 \leqslant 0.55$. La mejor solución conocida es $\vec{x}^* = (679.945148297028709, 1026.06697600004691, 0.118876369094410433, -0.39623348521517826)$ donde $f(\vec{x}^*) = 5126.4967140071$.

**g06**

Minimizar $f(\vec{x}) = (x_1 - 10)^3 - (x_2 - 20)^3$ sujeto a:

$$g_1(\vec{x}) = -(x_1 - 10)^2 - (x_2 - 5)^2 + 100 \leqslant 0$$

$$g_2(\vec{x}) = -(x_1 - 6)^2 - (x_2 - 5)^2 - 82.81 \leqslant 0$$

donde $13 \leqslant x_1 \leqslant 100$ y $0 \leqslant x_2 \leqslant 100$. La solución óptima es $\vec{x} = (14.09500000000000064, 0.8429607892154795668)$ donde $f(\vec{x}) = -6961.81387558015$. Ambas restricciones están activas, en el óptimo.

**g07**

Minimizar $f(\vec{x}) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$ sujeto a:

$$g_1(\vec{x}) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leqslant 0$$

$$g_2(\vec{x}) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leqslant 0$$

$$g_3(\vec{x}) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leqslant 0$$

$$g_4(\vec{x}) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leqslant 0$$

$$g_5(\vec{x}) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leqslant 0$$

$$g_6(\vec{x}) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leqslant 0$$

$$g_7(\vec{x}) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leqslant 0$$

$$g_8(\vec{x}) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leqslant 0$$

donde $-10 \leqslant x_i \leqslant 10$ $(i = 1, ..., 10)$. La solución óptima es $\vec{x} = (2.17199634142692, 2.3636830416034, 8.77392573913157, 5.09598443745173, 0.990654756560493, 1.43057392853463, 1.32164415364306, 9.82872576524495, 8.2800915887356, 8.3759266477347)$ donde $g07(\vec{x}) = 24.30620906818$ Las restricciones $(g_1, g_2, g_3, g_4, g_5$ y $g_6)$ están activas, en el óptimo.

**g08**

Minimizar $f(\vec{x}) = -\frac{\text{sen}^3(2\pi x_1)\ \text{sen}(2\pi x_2)}{x_1^3(x_1+x_2)}$ sujeto a:

$$g_1(\vec{x}) = x_1^2 - x_2 + 1 \leqslant 0$$

$$g_2(\vec{x}) = 1 - x_1 + (x_2 - 4)^2 \leqslant 0$$

donde $0 \leqslant x_1 \leqslant 10$ y $0 \leqslant x_2 \leqslant 10$. El óptimo se localiza en $\vec{x} = (1.22797135260752599, 4.24537336612274885)$ donde $f(\vec{x}) = -0.0958250414180359$.

**g09**

Minimizar $f(\vec{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$ sujeto a:

$$g_1(\vec{x}) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leqslant 0$$

$$g_2(\vec{x}) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leqslant 0$$

$$g_3(\vec{x}) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leqslant 0$$

$$g_4(\vec{x}) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leqslant 0$$

donde $0 \leqslant x_i \leqslant 10$ for $(i = 1, ..., 7)$. La solución óptima es $\vec{x} = (2.33049935147405174, 1.95137236847114592, -0.477541399510615805, 4.36572624923625874, -0.624486959100388983, 1.03813099410962173, 1.5942266780671519)$ donde $f(\vec{x}) = 680.630057374402$. Dos restricciones están activas ($g_1$ y $g_4$), en el óptimo.

**g10**

Minimizar $f(\vec{x}) = x_1 + x_2 + x_3$ sujeto a:

$$g_1(\vec{x}) = -1 + 0.0025(x_4 + x_6) \leqslant 0$$

$$g_2(\vec{x}) = -1 + 0.0025(x_5 + x_7 - x_4) \leqslant 0$$

$$g_3(\vec{x}) = -1 + 0.01(x_8 - x_5) \leqslant 0$$

$$g_4(\vec{x}) = -x_1x_6 + 833.33252x_4 + 100x_1 - 83333.333 \leqslant 0$$

$$g_5(\vec{x}) = -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \leqslant 0$$

$$g_6(\vec{x}) = -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \leqslant 0$$

donde $100 \leqslant x_1 \leqslant 10000$, $1000 \leqslant x_i \leqslant 10000$ $(i = 2, 3)$ y $10 \leqslant x_i \leqslant 1000$ $(i = 4, ...., 8)$. La solución óptima es $\vec{x} = (579.306685017979589, 1359.97067807935605, 5109.97065743133317, 182.01769963061534, 295.601173702746792, 217.982300369384632, 286.41652592786852,$

395.601173702746735), donde $f(\vec{x}) = 7049.24802052867$. Las restricciones ($g_1, g_2$ y $g_3$), están activas, en el óptimo.

**g11**

Minimizar $f(\vec{x}) = x_1^2 + (x_2 - 1)^2$ sujeto a:

$$h_1(\vec{x}) = x_2 - x_1^2 = 0$$

donde $-1 \leqslant x_1 \leqslant 1$ y $-1 \leqslant x_2 \leqslant 1$. La solución óptima es $\vec{x} = (-0.707036070037170616, 0.500000004333606807)$ donde $f(\vec{x}) = 0.7499$.

**g12**

Minimizar $f(\vec{x}) = -(100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2)/100$ sujeto a:

$$g_1(\vec{x}) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leqslant 0$$

donde $0 \leqslant x_i \leqslant 10 (i = 1, 2, 3)$ y $p, q, r = 1, 2, ..., 9$. La región factibles del espacio de búsqueda consta de 93 esferas disjuntas. Un punto $(x_1, x_2, x_3)$ es factible si y solo si existe $p, q, r$ tal que se cumpla la desigualdad antes indicada. El óptimo se localiza en $\vec{x} = (5, 5, 5)$ donde $f(\vec{x}) = -1$.

**g13**

Minimizar $f(x) = e^{x_1 x_2 x_3 x_4 x_5}$ sujeto a:

$$h_1(\vec{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0$$
$$h_2(\vec{x}) = x_2 x_3 - 5 x_4 x_5 = 0$$
$$h_3(\vec{x}) = x_1^3 + x_2^3 + 1 = 0$$

donde $-2.3 \leqslant x_i \leqslant 2.3$ $(i = 1, 2)$ y $-3.2 \leqslant x_i \leqslant 3.2$ $(i = 3, 4, 5)$. La solución óptima es $\vec{x} = (-1.71714224003, 1.59572124049468, 1.8272502406271, -0.763659881912867, -0.76365986736498)$ donde $f(\vec{x}) = 0.053941514041898$.

**g14**

Minimizar $f(\vec{x}) = \sum_{i=1}^{10} x_i (c_i + \ln \frac{x_i}{\sum_{j=1}^{10} x_j})$ sujeto a:

$$h_1(\vec{x}) = x_1 + 2x_2 + 2x_3 + x_6 + x_{10} - 2 = 0$$
$$h_2(\vec{x}) = x_4 + 2x_5 + x_6 + x_7 - 1 = 0$$

$$h_3(\vec{x}) = x_3 + x_7 + x_8 + 2x_9 + x_{10} - 1 = 0$$

donde los límites de las variables están establecidos por $0 \leqslant x_i \leqslant 10$ $(i = 1, ..., 10)$ y $c_1 = -6.089, c_2 = -17.164, c_3 = -34.054, c_4 = -5.914, c_5 = -24.721, c_6 = -14.986, c_7 = -24.1, c_8 = -10.708, c_9 = -26.662, c_{10} = -22.179,$. La mejor solución conocida es $\vec{x} =$ (0.0406684113216282, 0.147721240492452, 0.783205732104114, 0.00141433931889084, 0.485293636780388, 0.000693183051556082, 0.0274052040687766, 0.0179509660214818, 0.0373268186859717, 0.0968844604336845) donde $f(\vec{x}) = -47.7648884594915$.

**g15**

Minimizar $f(\vec{x}) = 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x_3$ sujeto a:

$$h_1(\vec{x}) = x_1^2 + x_2^2 + x_3^2 - 25 = 0$$

$$h_2(\vec{x}) = 8x_1 + 14x_2 + 7x_3 - 56 = 0$$

donde los límites de las variables están establecidos por $0 \leqslant x_i \leqslant 10$ $(i = 1, 2, 3)$. La mejor solución conocida es $\vec{x} =$ (3.51212812611795133, 0.216987510429556135, 3.55217854929179921) donde $f(\vec{x}) = 961.715022289961$.

**g16**

Minimizar $f(\vec{x}) = 0.000117y_{14} + 0.1365 + 0.00002358y_{13} + 0.000001502y_{16} + 0.0321y_{12} + 0.004324y_5 + 0.0001(\frac{c_{15}}{c_{16}}) + 37.48(\frac{y_2}{c_{12}}) - 0.0000005843y_{17}$ sujeto a:

$$g_1(\vec{x}) = \tfrac{0.28}{0.72}y_5 - y_4 \leqslant 0$$

$$g_2(\vec{x}) = x_3 - 1.5x_2 \leqslant 0$$

$$g_3(\vec{x}) = 3496\tfrac{y_2}{c_{12}} - 21 \leqslant 0$$

$$g_4(\vec{x}) = 110.6 + y_1 - \tfrac{62212}{c_{17}} \leqslant 0$$

$$g_5(\vec{x}) = 213.1 - y_1 \leqslant 0$$

$$g_6(\vec{x}) = y_1 - 405.23 \leqslant 0$$

$$g_7(\vec{x}) = 17.505 - y_2 \leqslant 0$$

$$g_8(\vec{x}) = y_2 - 1053.6667 \leqslant 0$$

$$g_9(\vec{x}) = 11.275 - y_3 \leqslant 0$$

$$g_{10}(\vec{x}) = y_3 - 35.03 \leqslant 0$$

$$g_{11}(\vec{x}) = 214.228 - y_4 \leqslant 0$$

$$g_{12}(\vec{x}) = y_4 - 665.585 \leqslant 0$$

$$g_{13}(\vec{x}) = 7.458 - y_5 \leqslant 0$$

$$g_{14}(\vec{x}) = y_5 - 584.463 \leqslant 0$$

$$g_{15}(\vec{x}) = 0.961 - y_6 \leqslant 0$$

$$g_{16}(\vec{x}) = y_6 - 265.916 \leqslant 0$$

$$g_{17}(\vec{x}) = 1.612 - y_7 \leqslant 0$$

$$g_{18}(\vec{x}) = y_7 - 7.046 \leqslant 0$$

$$g_{19}(\vec{x}) = 0.146 - y_8 \leqslant 0$$

$$g_{20}(\vec{x}) = y_8 - 0.222 \leqslant 0$$

$$g_{21}(\vec{x}) = 107.99 - y_9 \leqslant 0$$

$$g_{22}(\vec{x}) = y_9 - 273.366 \leqslant 0$$

$$g_{23}(\vec{x}) = 922.693 - y_{10} \leqslant 0$$

$$g_{24}(\vec{x}) = y_{10} - 1286.105 \leqslant 0$$

$$g_{25}(\vec{x}) = 926.832 - y_{11} \leqslant 0$$

$$g_{26}(\vec{x}) = y_{11} - 1444.046 \leqslant 0$$

$$g_{27}(\vec{x}) = 18.766 - y_{12} \leqslant 0$$

$$g_{28}(\vec{x}) = y_{12} - 537.141 \leqslant 0$$

$$g_{29}(\vec{x}) = 1072.163 - y_{13} \leqslant 0$$

$$g_{30}(\vec{x}) = y_{13} - 3247.039 \leqslant 0$$

$$g_{31}(\vec{x}) = 8961.448 - y_{14} \leqslant 0$$

$$g_{32}(\vec{x}) = y_{14} - 26844.086 \leqslant 0$$

$$g_{33}(\vec{x}) = 0.063 - y_{15} \leqslant 0$$

$$g_{34}(\vec{x}) = y_{15} - 0.386 \leqslant 0$$

$$g_{35}(\vec{x}) = 71084.33 - y_{16} \leqslant 0$$

$$g_{36}(\vec{x}) = y_{16} - 140000 \leqslant 0$$

$$g_{37}(\vec{x}) = 2802713 - y_{17} \leqslant 0$$

$$g_{38}(\vec{x}) = y_{17} - 12146108 \leqslant 0$$

donde:

$$y_1 = x_2 + x_3 + 41.6$$

$$c_1 = 0.024x_4 - 4.62$$

$$y_2 = \frac{12.5}{c_1} + 12.0$$

$$c_2 = 0.0003535x_1^2 + 0.5311x_1 + 0.08705y_2x_1$$

$$c_3 = 0.052x_1 + 78.0 + 0.002377y_2x_1$$

$$y_3 = \frac{c_2}{c_3}$$

$$y_4 = 19.0y_3$$

$$c_4 = 0.04782(x_1 - y_3) + \frac{(0.1956(x_1 - y_3)^2)}{x_2} + 0.6376y_4 + 1.594y_3$$

$$c_5 = 100.0x_2$$

$$c_6 = x_1 - y_3 - y_4$$

$$c_7 = 0.950 - \frac{c_4}{c_5}$$

$$y_5 = c_6c_7$$

$$y_6 = x_1 - y_5 - y_4 - y_3$$

$$c_8 = (y_5 + y_4)0.995$$

$$y_7 = \frac{c_8}{y_1}$$

$$y_8 = \frac{c_8}{3798.0}$$

$$c_9 = y_7 - \frac{(0.0663y_7)}{y_8 - 0.3153}$$

$$y_9 = \frac{96.82}{c_9} + 0.321y_1$$

$$y_{10} = 1.29y_5 + 1.258y_4 + 2.29y_3 + 1.71y_6$$

$$y_{11} = 1.71x_1 - 0.452y_4 + 0.58y_3$$

$$c_{10} = \frac{12.3}{752.3}$$

$$c_{11} = 1.75y_20.995x_1$$

$$c_{12} = 0.995y_{10} + 1998.0$$

$$y_{12} = c_{10}x_1 + \frac{c_{11}}{c_{12}}$$

$$y_{13} = c_{12} - 1.75y_2$$

$$y_{14} = 3623.0 + 64.4x_2 + 58.4x_3 + 146312.0/(y_9 + x_5)$$

$$c_{13} = 0.995y_{10} + 60.8x_2 + 48.0x_4 - 0.1121y_{14} - 5095.0$$

$$y_{15} = \frac{y_{13}}{c_{13}}$$

$$y_{16} = 148000.0 - 331000.0y_{15} + 40.0y_{13} - 61.0y_{15}y_{13}$$

$$c_{14} = 2324.0y_{10} - 28740000.0y_2$$

$$y_{17} = 14130000.0 - 1328.0y_{10} - 531.0y_{11} + \frac{c_{14}}{c_{12}}$$

$$c_{15} = \frac{y_{13}}{y_{15}} - \frac{y_{13}}{0.52}$$

$$c_{16} = 1.104 - 0.72y_{15}$$

$$c_{17} = y_9 + x_5$$

y donde los límites de las variables son $704.4148 \leqslant x_1 \leqslant 906.3855$, $68.6 \leqslant x_2 \leqslant 288.88$, $0 \leqslant x_3 \leqslant 134.75$, $193 \leqslant x_4 \leqslant 287.0966$ y $25 \leqslant x_5 \leqslant 84.1988$. La mejor solución conocida es $x = (705.174537070090537, 68.5999999999999943, 102.899999999999991, 282.324931593660324, 37.5841164258054832)$ donde $f(x) = -1.90515525853479$.

**g17**

Minimizar $f(\vec{x}) = f(x_1) + f(x_2)$

donde:

$$f(x_1) = \begin{cases} 30x_1 & 0 \leqslant x_1 < 300 \\ 31x_1 & 300 \leqslant x_1 < 400 \end{cases}$$

$$f(x_2) = \begin{cases} 28x_2 & 0 \leqslant x_2 < 100 \\ 29x_2 & 100 \leqslant x_2 < 200 \\ 30x_2 & 200 \leqslant x_2 < 1000 \end{cases}$$

sujeto a:

$$h_1(\vec{x}) = -x_1 + 300.0 - \left(\frac{(x_3x_4)}{131.078}\right)\cos(1.48477 - x_6) + \left(\frac{0.90798x_3^2}{131.078}\right)\cos(1.47588) = 0$$

$$h_2(\vec{x}) = -x_2 - \left(\frac{(x_3x_4)}{131.078}\right)\cos(1.48477 + x_6) + \left(\frac{0.90798x_4^2}{131.078}\right)\cos(1.47588) = 0$$

$$h_3(\vec{x}) = -x_5 - \left(\frac{(x_3x_4)}{131.078}\right)\text{sen}(1.48477 + x_6) + \left(\frac{0.90798x_4^2}{131.078}\right)\text{sen}(1.47588) = 0$$

$$h_4(\vec{x}) = 200.0 - \left(\frac{(x_3x_4)}{131.078}\right)\text{sen}(1.48477 - x_6) + \left(\frac{0.90798x_3^2}{131.078}\right)\text{sen}(1.47588) = 0$$

donde los límites de las variables son $0 \leqslant x_1 \leqslant 400$, $0 \leqslant x_2 \leqslant 1000$, $340 \leqslant x_3 \leqslant 420$, $340 \leqslant x4 \leqslant 420$, $-1000 \leqslant x_5 \leqslant 1000$ y $0 \leqslant x_6 \leqslant 0.5236$. La mejor solución conocida es $x = (201.784467214523659, 99.9999999999999005, 383.071034852773266, 420, -10.9076584514292652, 0.0731482312084287128)$ donde $f(x) = 8853.53967480648$.

**g18**

Minimizar $f(\vec{x}) = -0.5(x_1x_4 - x_2x_3 + x_3x_9 - x_5x_9 + x_5x_8 - x_6x_7)$ sujeto a:

$$g_1(\vec{x}) = x_3^2 + x_4^2 - 1.0 \leqslant 0$$
$$g_2(\vec{x}) = x_9^2 - 1.0 \leqslant 0$$
$$g_3(\vec{x}) = x_5^2 + x_6^2 - 1.0 \leqslant 0$$
$$g_4(\vec{x}) = x_1^2 + (x_2 - x_9, 2.0) - 1.0 \leqslant 0$$
$$g_5(\vec{x}) = (x_1 - x_5)^2 + (x_2 - x_6)^2 - 1.0 \leqslant 0$$
$$g_6(\vec{x}) = (x_1 - x_7)^2 + (x_2 - x_8)^2 - 1.0 \leqslant 0$$
$$g_7(\vec{x}) = (x_3 - x_5)^2 + (x_4 - x_6)^2 - 1.0 \leqslant 0$$
$$g_8(\vec{x}) = (x_3 - x_7)^2 + (x_4 - x_8)^2 - 1.0 \leqslant 0$$
$$g_9(\vec{x}) = x_7^2 + (x_8 - x_9)^2 - 1.0 \leqslant 0$$
$$g_{10}(\vec{x}) = x_2x_3 - x_1x_4 \leqslant 0$$
$$g_{11}(\vec{x}) = -x_3x_9 \leqslant 0$$
$$g_{12}(\vec{x}) = x_5x_9 \leqslant 0$$
$$g_{13}(\vec{x}) = x_6x_7 - x_5x_8 \leqslant 0$$

donde los límites son $-10 \leqslant x_i \leqslant 10(i = 1, ..., 8)$ y $0 \leqslant x_9 \leqslant 20$. La mejor solución conocida es $x = (-0.657776192427943163, -0.153418773482438542, 0.323413871675240938, -0.946257611651304398, -0.657776194376798906, -0.753213434632691414, 0.323413874123576972, -0.346462947962331735, 0.59979466285217542)$
donde $f(x) = -0.866025403784439$.

**g19**

Minimizar $f(\vec{x}) = \sum_{j=1}^{5}\sum_{i=1}^{5} c_{ij}x_{(10+i)}x_{(10+j)} + 2\sum_{i=1}^{5} d_j x_{(10+j)}^3 - \sum_{i=1}^{10} b_i x_i$ sujeto a:

$$g_j(\vec{x}) = -2\sum_{i=1}^{5} c_{ij}x_{(10+i)} - 3d_j x_{10+j}^2 - e_j + \sum_{i=1}^{10} a_{ij}x_i \leqslant 0$$

donde $\vec{b} = [-40, -2, -0.25, -4, -4, -1, -40, -60, 5, 1]$ y los datos restantes se muestran en la tabla A. Los límites son $0 \leqslant x_i \leqslant 10(i = 1, ..., 15)$. La mejor solución conocida es $x = (1.66991341326291344e - 17, 3.95378229282456509e - 16, 3.94599045143233784, 1.06036597479721211e-16, 3.2831773458454161, 9.99999999999999822, 1.12829414671605333e-17, 1.2026194599794709e - 17, 2.50706276000769697e - 15, 2.24624122987970677e - 15,$

0.370764847417013987, 0.278456024942955571, 0.523838487672241171, 0.388620152510322781, 0.298156764974678579) donde $f(x) = 32.6555929502463$.

| j | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $e_j$ | -15 | -27 | -36 | -18 | -12 |
| $c_{1j}$ | 30 | -20 | -10 | 32 | -10 |
| $c_{2j}$ | -20 | 39 | -6 | -31 | 32 |
| $c_{3j}$ | -10 | -6 | 10 | -6 | -10 |
| $c_{4j}$ | 32 | -31 | -6 | 39 | -20 |
| $c_{5j}$ | -10 | 32 | -10 | -20 | 30 |
| $d_j$ | 4 | 8 | 10 | 6 | 2 |
| $a_{1j}$ | -16 | 2 | 0 | 1 | 0 |
| $a_{2j}$ | 0 | -2 | 0 | 0.4 | 2 |
| $a_{3j}$ | -3.5 | 0 | 2 | 0 | 0 |
| $a_{4j}$ | 0 | -2 | 0 | -4 | -1 |
| $a_{5j}$ | 0 | -9 | -2 | 1 | -2.8 |
| $a_{6j}$ | 2 | 0 | -4 | 0 | 0 |
| $a_{7j}$ | -1 | -1 | -1 | -1 | -1 |
| $a_{8j}$ | -1 | -2 | -3 | -2 | -1 |
| $a_{9j}$ | 1 | 2 | 3 | 4 | 5 |
| $a_{10j}$ | 1 | 1 | 1 | 1 | 1 |

Tabla A.1: Conjunto de datos para el problema g19

**g21**
Minimizar $f(\vec{x}) = x_1$ sujeto a:

$$g_1(\vec{x}) = -x_1 + 35x_2^{0.6} + 35x_3^{0.6} \leqslant 0$$

$$h_1(\vec{x}) = -300x_3 + 7500x_5 - 7500x_6 - 25x_4x_5 + 25x_4x_6 + x_3x_4 = 0$$

$$h_2(\vec{x}) = 100x_2 + 155.365x_4 + 2500x_7 - x_2x_4 - 25x_4x_7 - 15536.5 = 0$$

$$h_3(\vec{x}) = -x_5 + \ln(-x_4 + 900) = 0$$

$$h_4(\vec{x}) = -x_6 + \ln(x_4 + 300) = 0$$

$$h_5(\vec{x}) = -x_7 + \ln(-2x_4 + 700) = 0 = 0$$

donde los límites son $0 \leqslant x_1 \leqslant 1000$, $0 \leqslant x_2, x_3 \leqslant 40$, $100 \leqslant x_4 \leqslant 300$, $6.3 \leqslant x_5 \leqslant 6.7$, $5.9 \leqslant x_6 \leqslant 6.4$ y $4.5 \leqslant x_7 \leqslant 6.25$. La mejor solución es $x = (193.724510070034967, 5.56944131553368433e{-}27, 17.3191887294084914, 100.047897801386839, 6.68445185362377892, 5.99168428444264833, 6.21451648886070451)$ donde $f(x) = 193.724510070035$.

**g23**
Minimizar $f(\vec{x}) = -9x_5 - 15x_8 + 6x_1 + 16x_2 + 10(x_6 + x_7)$ sujeto a:

$$g_1(\vec{x}) = x_9x_3 + 0.02x_6 - 0.025x_5 \leqslant 0$$
$$g_2(\vec{x}) = x_9x_4 + 0.02x_7 - 0.015x_8 \leqslant 0$$
$$h_1(\vec{x}) = x_1 + x_2 - x_3 - x_4 = 0$$
$$h_2(\vec{x}) = 0.03x_1 + 0.01x_2 - x_9(x_3 + x_4) = 0$$
$$h_3(\vec{x}) = x_3 + x_6 - x_5 = 0$$
$$h_4(\vec{x}) = x_4 + x_7 - x_8 = 0$$

donde los límites son $0 \leqslant x_1, x_2, x_6 \leqslant 300$, $0 \leqslant x_3, x_5, x_7 \leqslant 100$, $0 \leqslant x_4, x_8 \leqslant 200$ y $0.01 \leqslant x_9 \leqslant 0.03$. La mejor solución conocida es $x = (0.0051000000000259465, 99.9947000000000514, 9.01920162996045897e-18, 99.9999000000000535, 0.00010000000027086086, 2.75700683389584542e-14, 99.9999999999999574, 2000.0100000100000100008)$ donde $f(x) = -400.055099999999584$.

**g24**
Minimizar $f(\vec{x}) = -x_1 - x_2$ sujeto a:

$$g_1(\vec{x}) = -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 2 \leqslant 0$$
$$g_2(\vec{x}) = -4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x_2 - 36 \leqslant 0$$

donde los límites son $0 \leqslant x_1 \leqslant 3$ y $0 \leqslant x_2 \leqslant 4$. El mínimo global factible es $x = (2.329520197477623.17849307411774)$ donde $f(x) = -5.50801327159536$. El problema tiene una región factible que consiste en dos subregiones discontinuas.

Apéndice B

# Gráficas de resultados

Este apéndice muestra una serie de diagramas que representan el comportamiento de los algoritmos implementados en esta tesis.

## B.1. Diagramas de caja

En la primera sección presentamos los diagramas de caja como un medio útil para representar datos. En dichos diagramas son representados los valores mínimo y máximo y los cuartiles inferior y superior (percentiles 25 y 75, respectivamente) y la mediana (percentil 50). La caja se extiende del cuartil inferior al superior, y es atravesada de un lado al otro por la mediana. A partir de los extremos de la caja se extienden las líneas (”bigotes”) hasta los valores mínimo y máximo. Además representa, mediante pequeños círculos, posibles valores atípicos del conjunto de datos.

## B.2. Comparación de gráficas de convergencia entre AMs

En esta sección presentamos las gráficas de convergencia de cada uno de los algoritmos. Puede apreciarse la evolución de cada uno de ellos, ya que en el eje de las $x$ se muestra el número de evaluaciones de la función objetivo, mientras que en el eje de las $y$ se muestra el valor a la aproximación del óptimo global. En varios de los casos se presenta un estancamiento de la solución, es decir que el algoritmo converge a una solución óptima local.

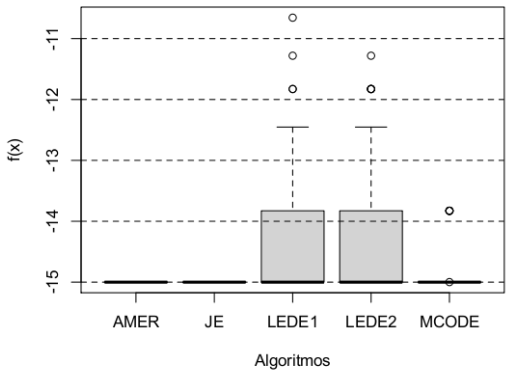
## B.3. Gráficas de convergencia del efecto del buscador local

Nuestro objetivo al presentar estas gráficas fue demostrar el uso benéfico de un buscador local acoplado de manera adecuada a un mecanismo de búsqueda local. Para ello, se realizaron los mismos experimentos (100 ejecuciones independientes con diferente semilla) con y sin el procedimiento de búsqueda local.

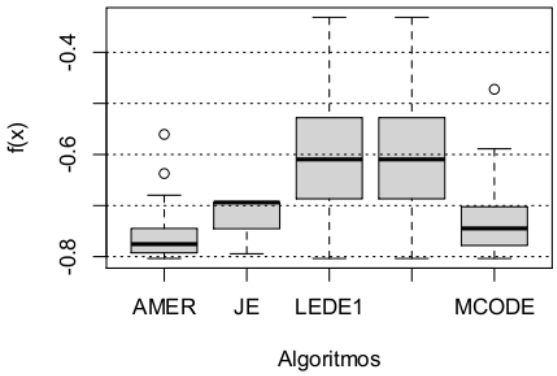
## B.4. Histograma de Frecuencias

Presentamos la tendencia del conjunto de datos experimentales de nuestra propuesta, mediante los histogramas de frecuencia de las 22 funciones de prueba. Un histograma re-

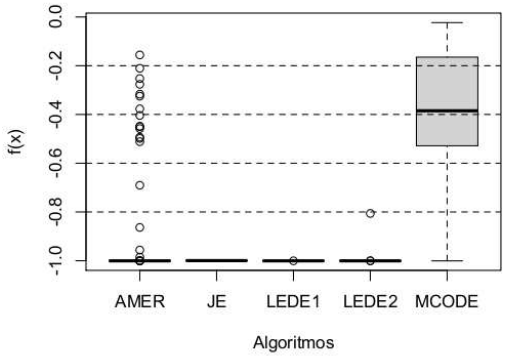
presenta el número de observaciones que caen en un mismo intervalo. Puede verse que las distribuciones no son normales, a diferencia del ajuste calculado por el método de *bootstrap*.



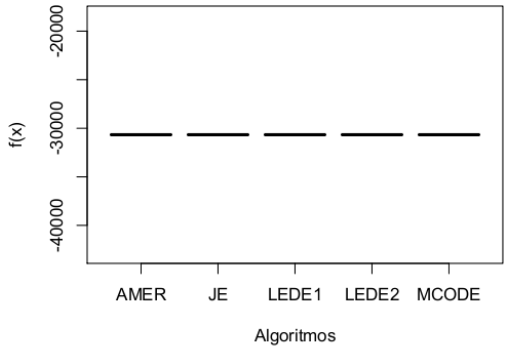

g01



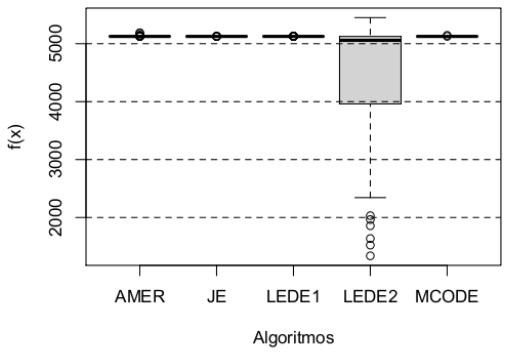

g02



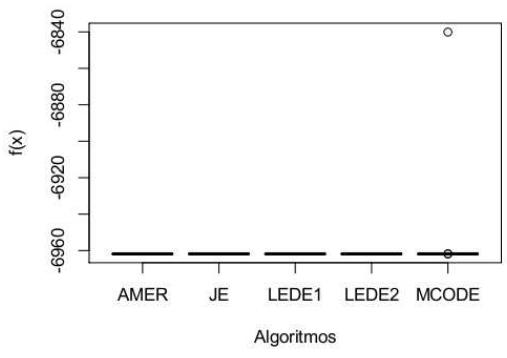

g03



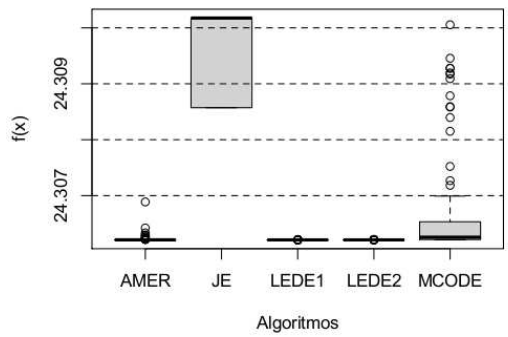

g04



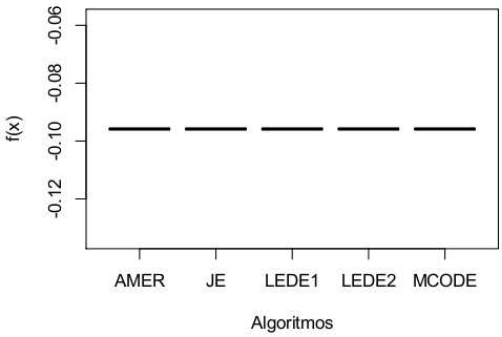

g05



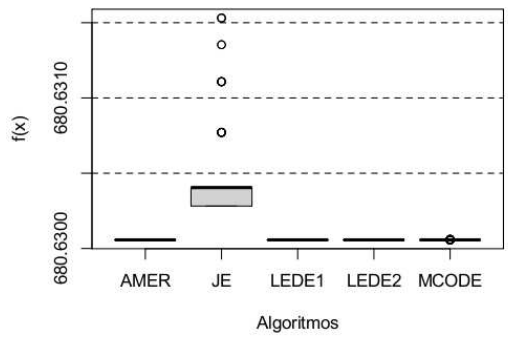

g06



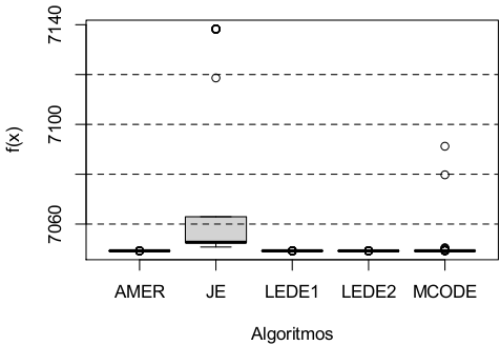

g07



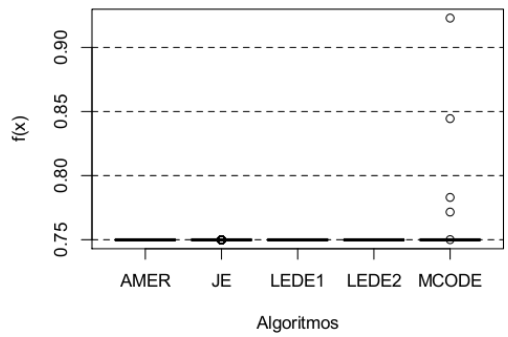

g08



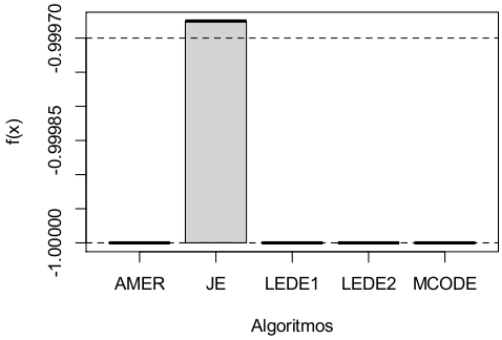

g09



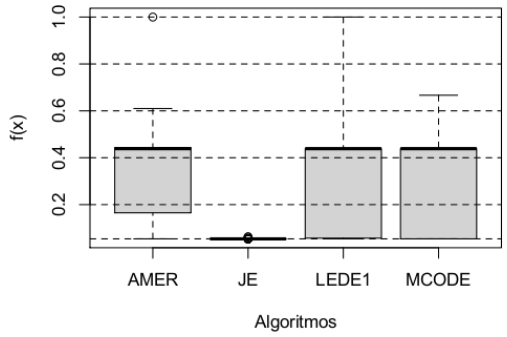

g10



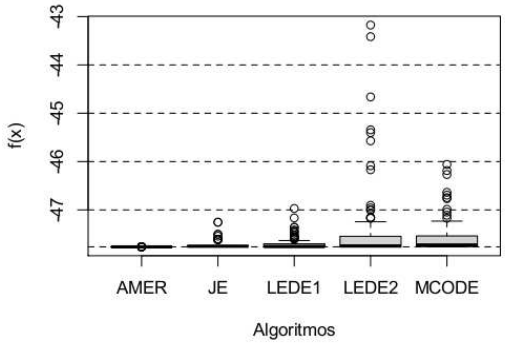

g11



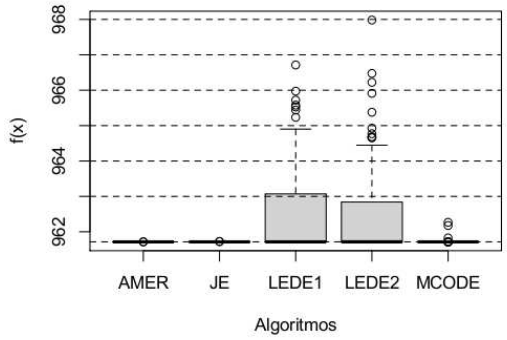

g12



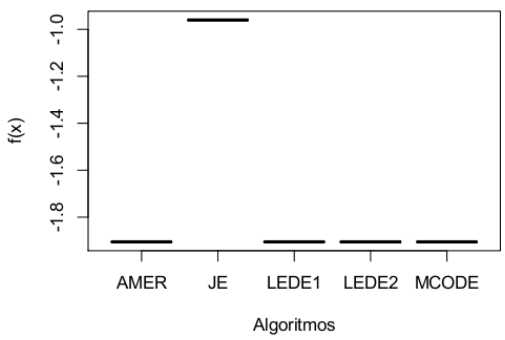

g13



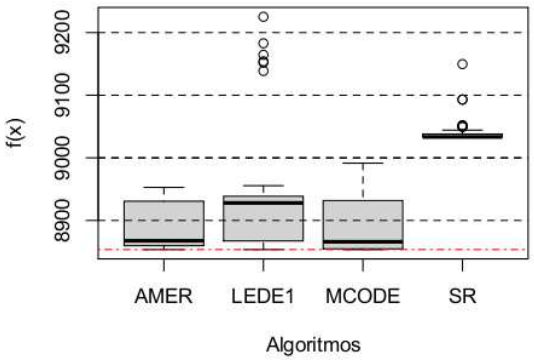

g14



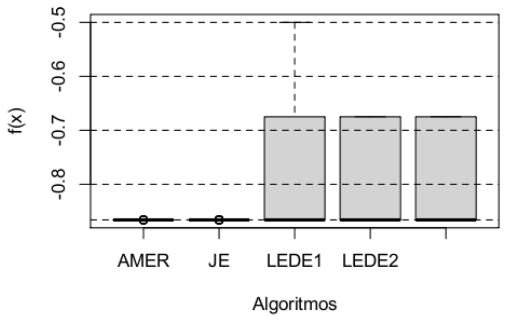

g15



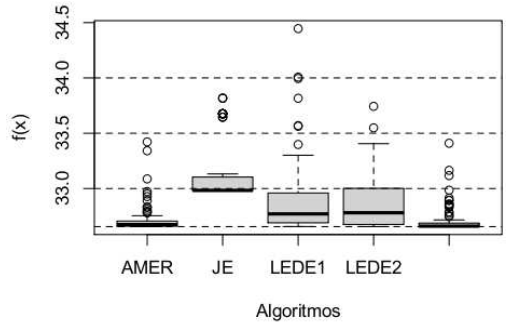

g16



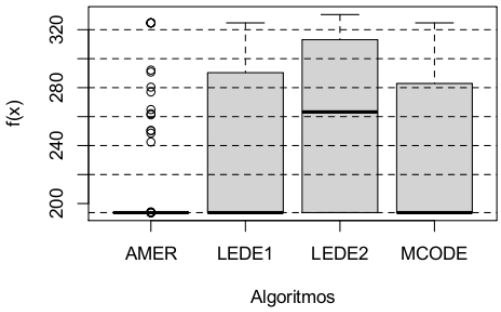

g17



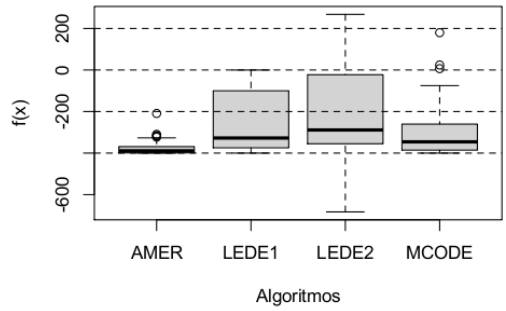

g18



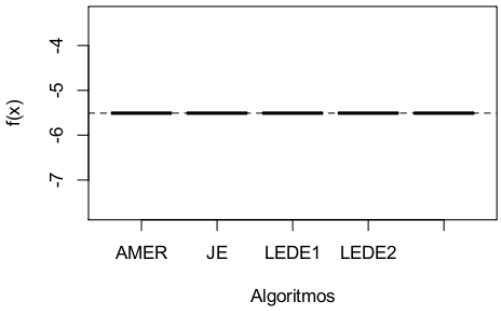

g19



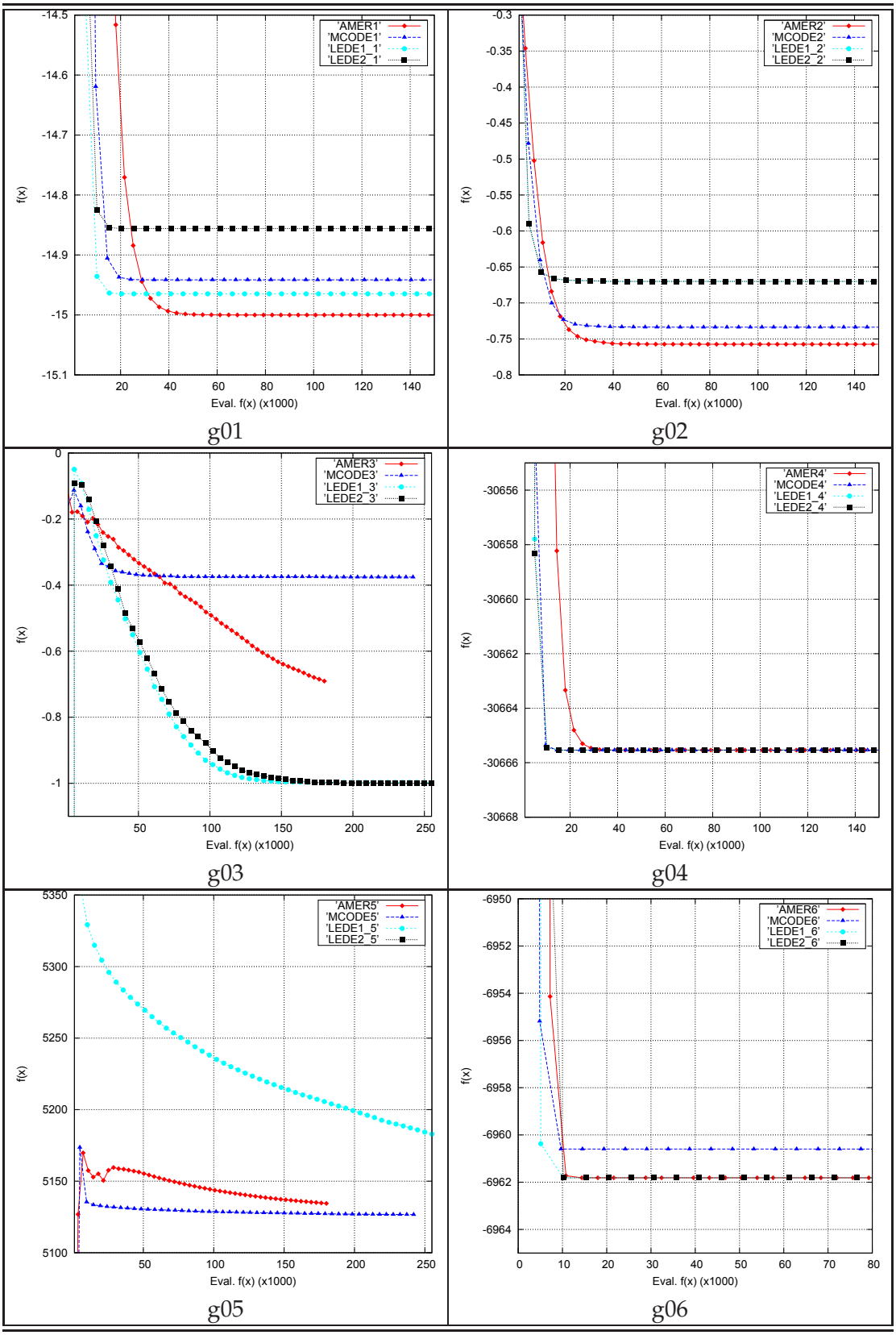

g21

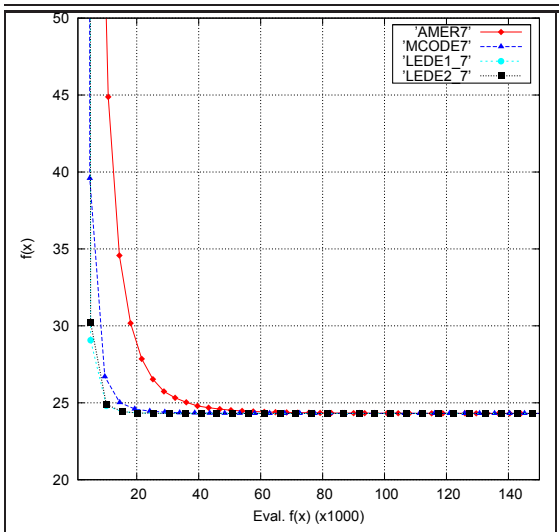



g23

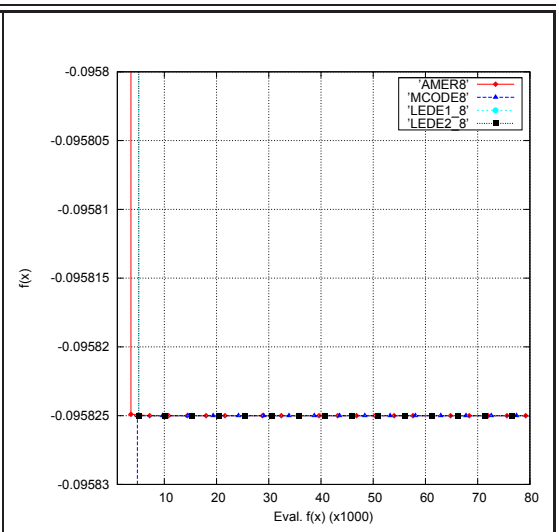



g24

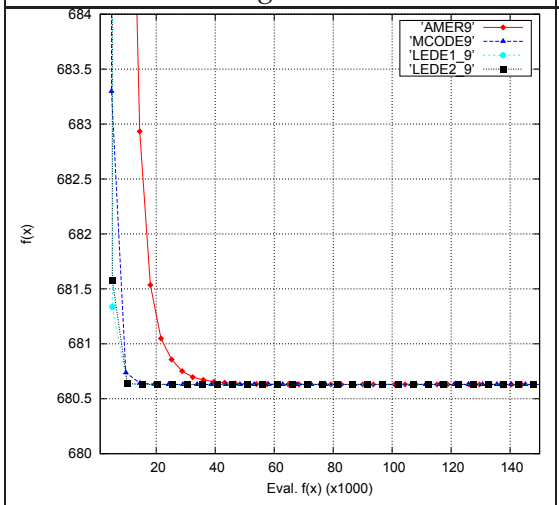

'AMER1'
'MCODE1'
'LEDE1_1'
'LEDE2_1'
f(x)
Eval. f(x) (x1000)
g01
'AMER2'
'MCODE2'
'LEDE1_2'
'LEDE2_2'
f(x)
Eval. f(x) (x1000)
g02
'AMER3'
'MCODE3'
'LEDE1_3'
'LEDE2_3'
f(x)
Eval. f(x) (x1000)
g03
'AMER4'
'MCODE4'
'LEDE1_4'
'LEDE2_4'
f(x)
Eval. f(x) (x1000)
g04
'AMER5'
'MCODE5'
'LEDE1_5'
'LEDE2_5'
f(x)
Eval. f(x) (x1000)
g05
'AMER6'
'MCODE6'
'LEDE1_6'
'LEDE2_6'
f(x)
Eval. f(x) (x1000)
g06



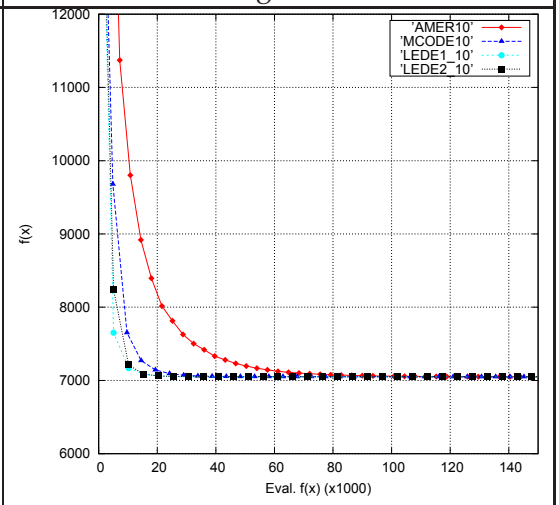

g07



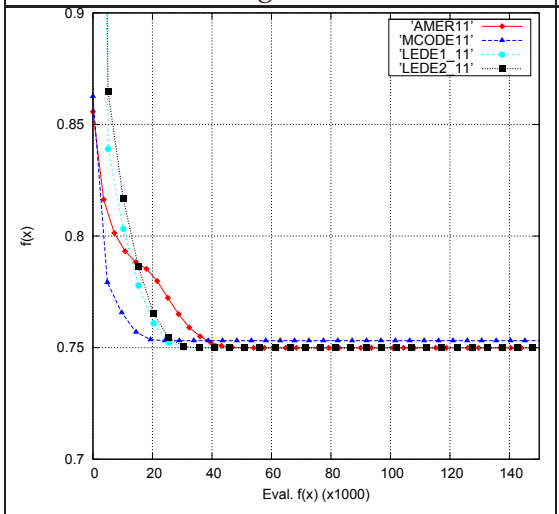

g08



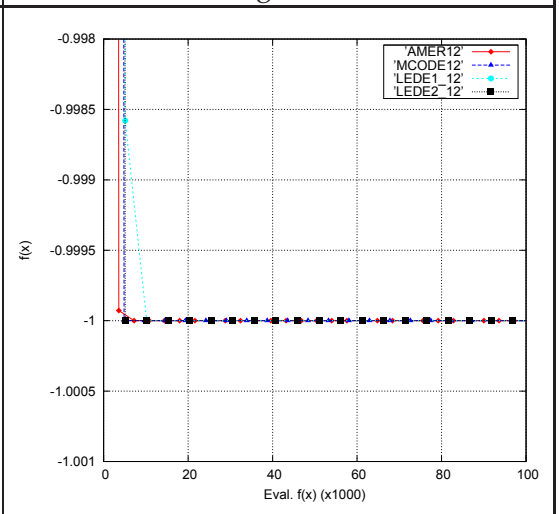

g09



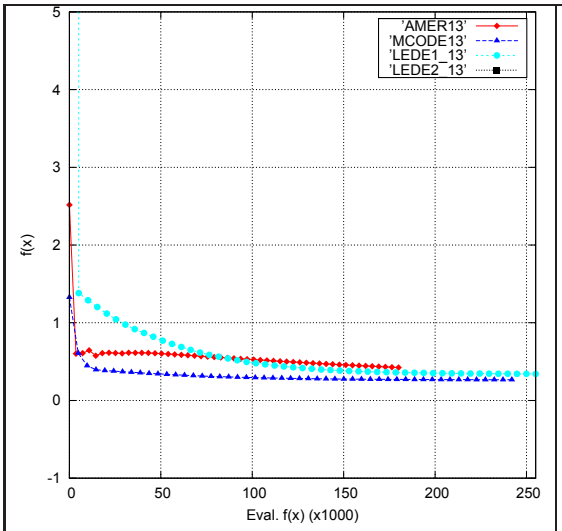

g10



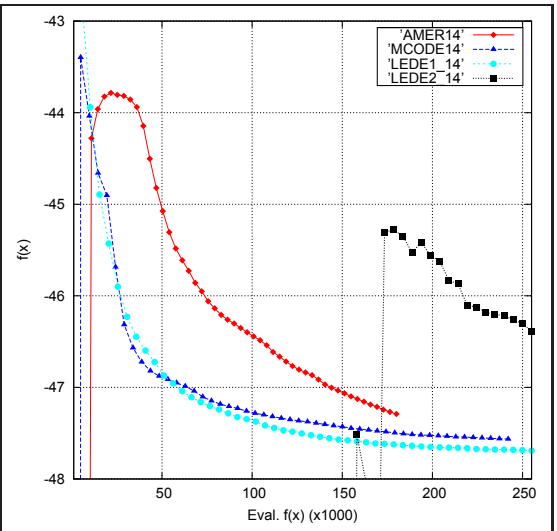

g11



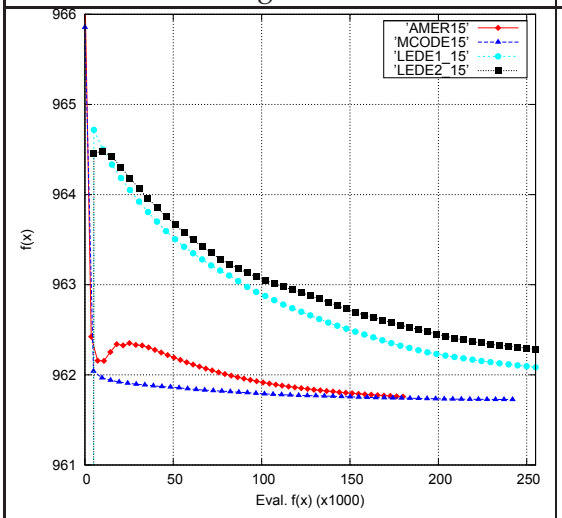

g12



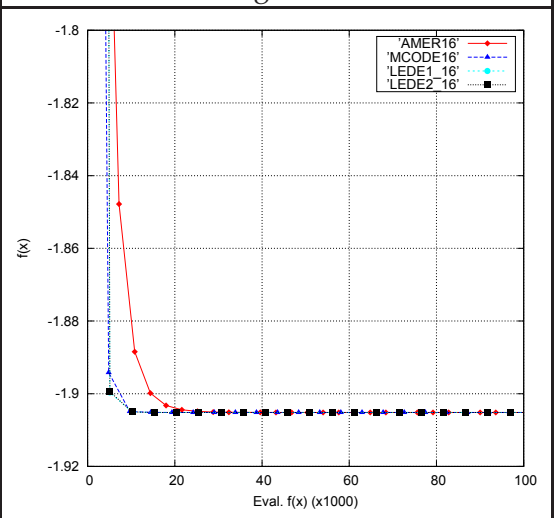

g13



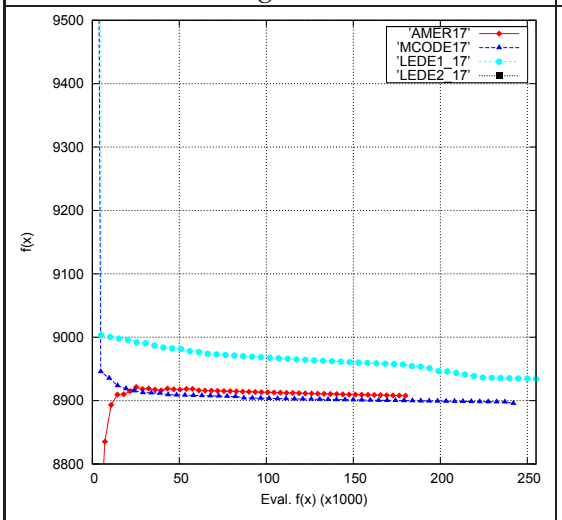

g14



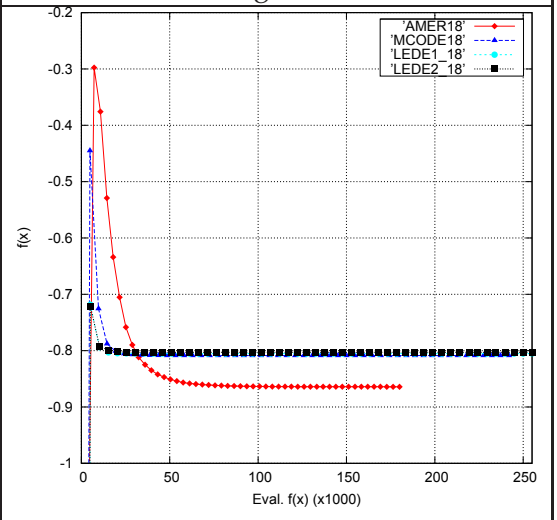

g15



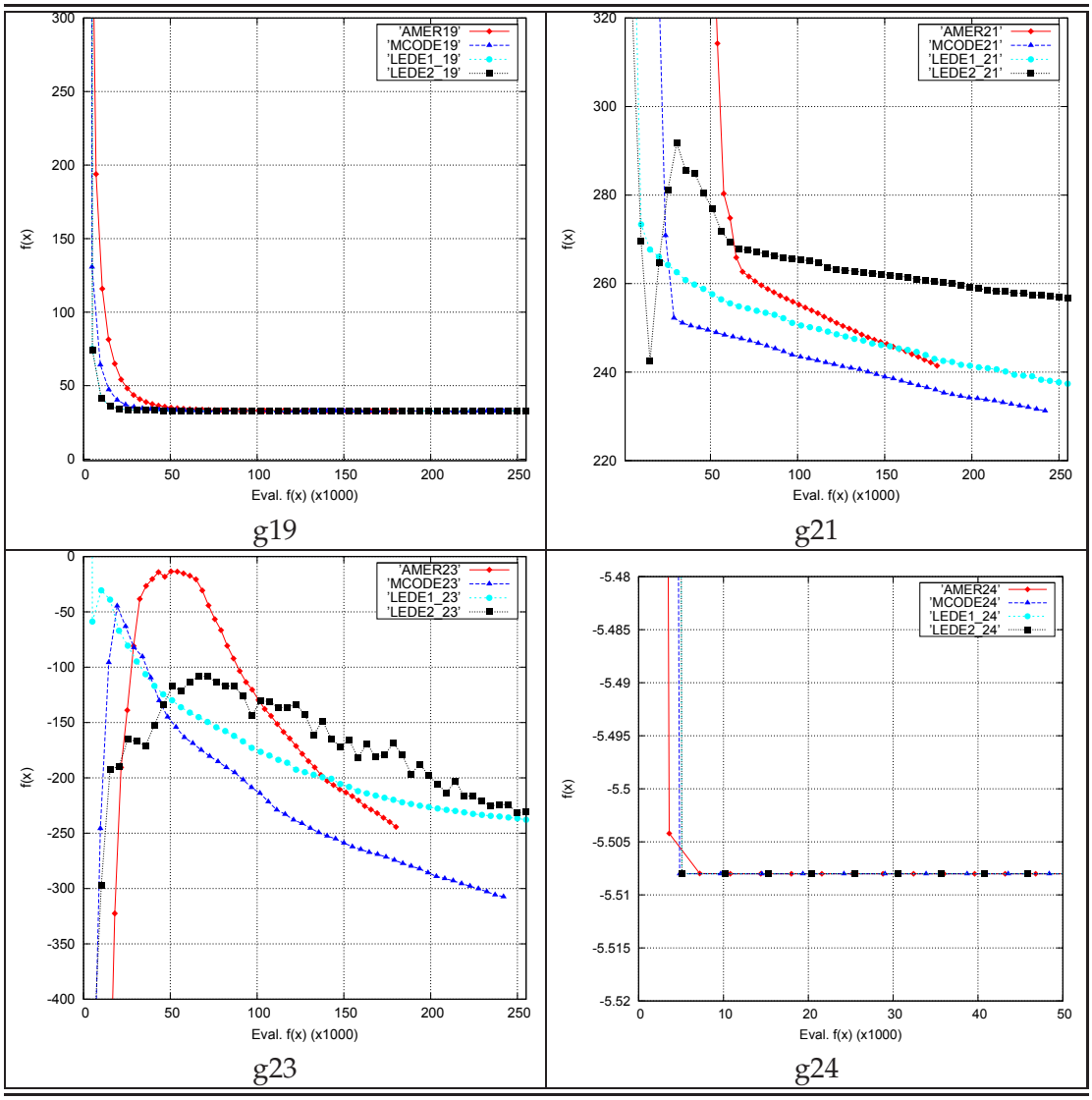

g16




g17

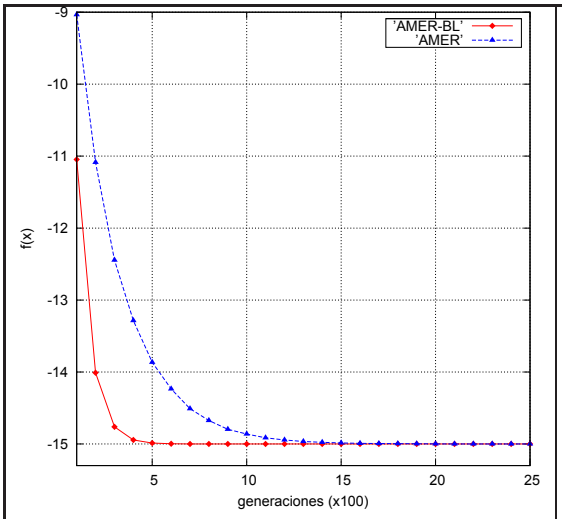



g18

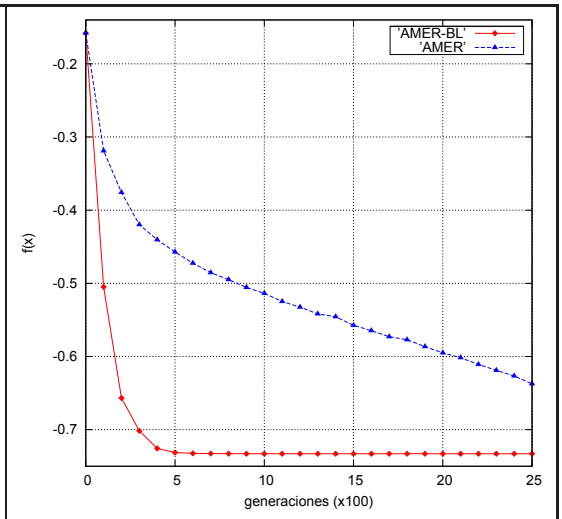



g19 g21

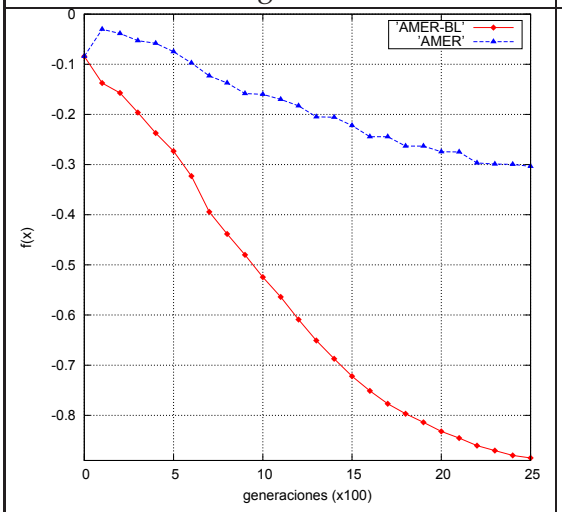
g23 g24



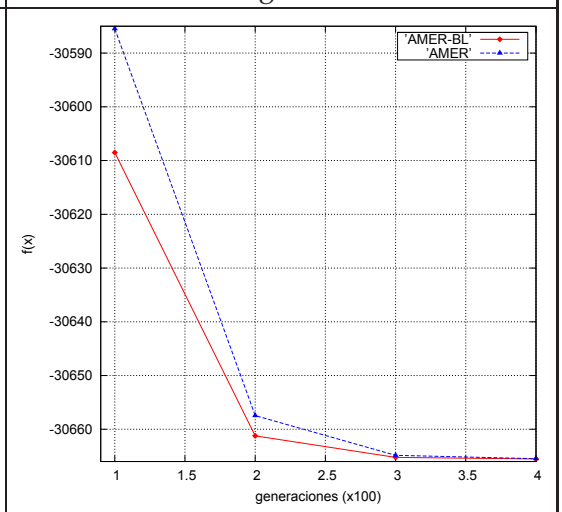

g01



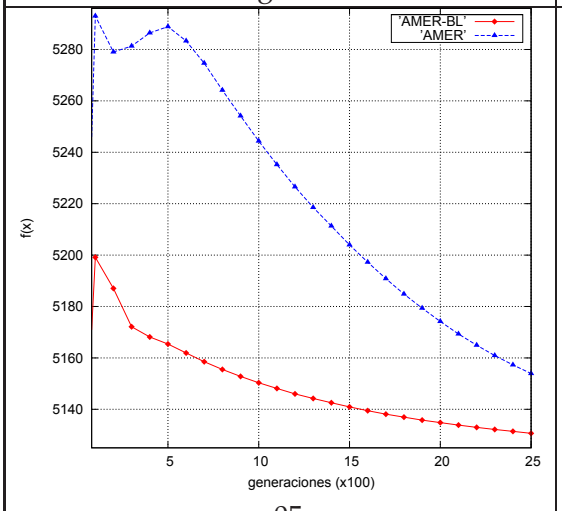

g02



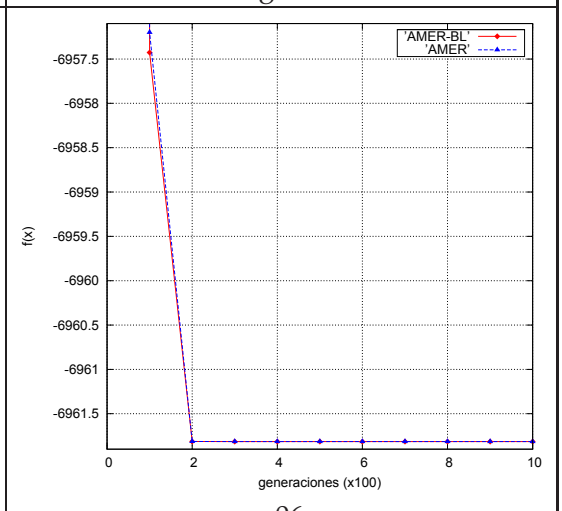

g03



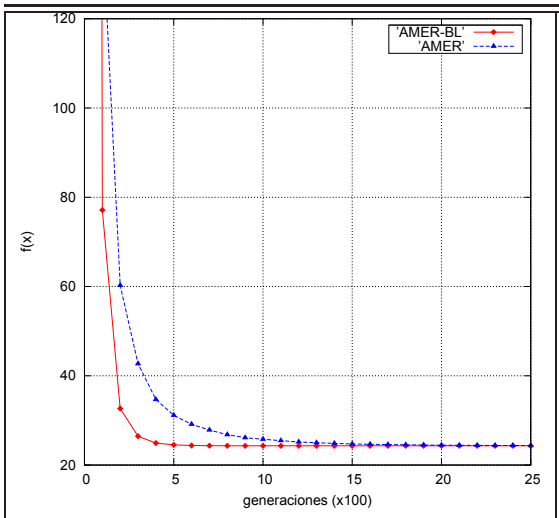

g04



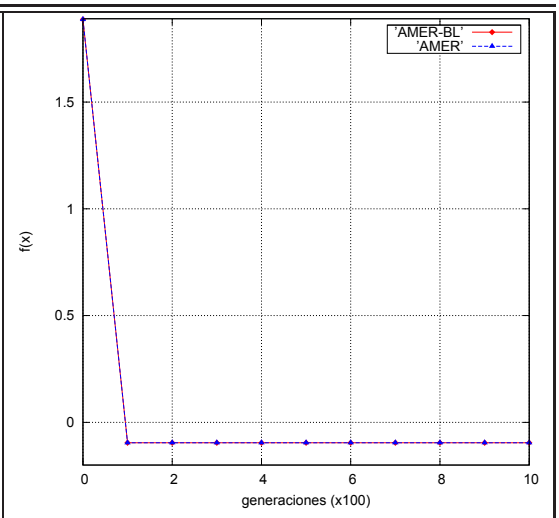

g05



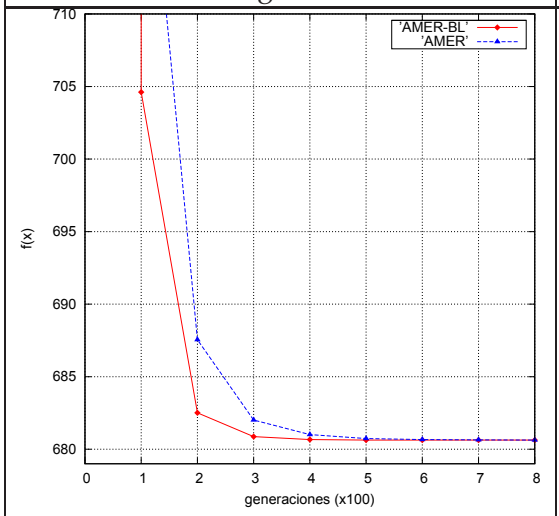

g06



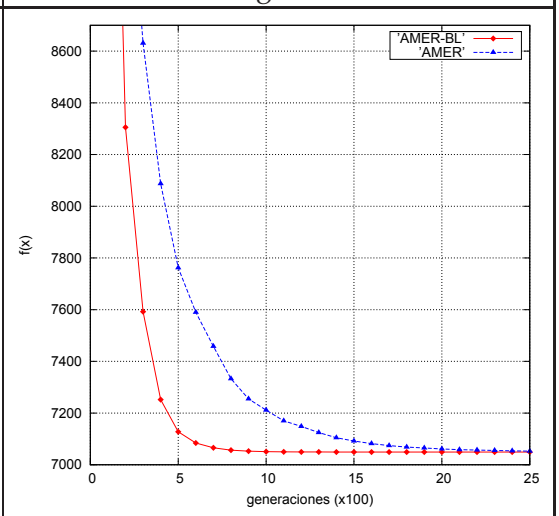

g07



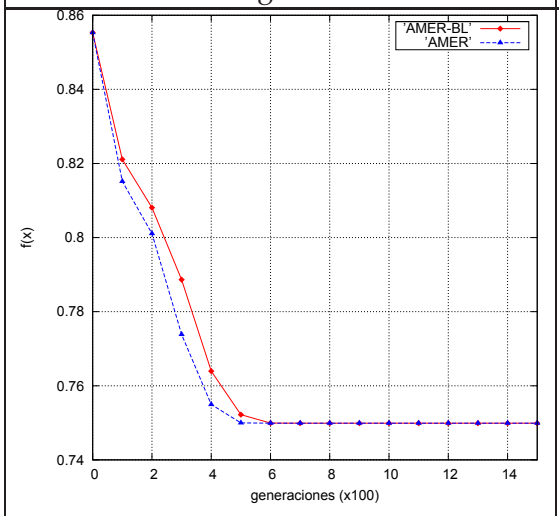

g08



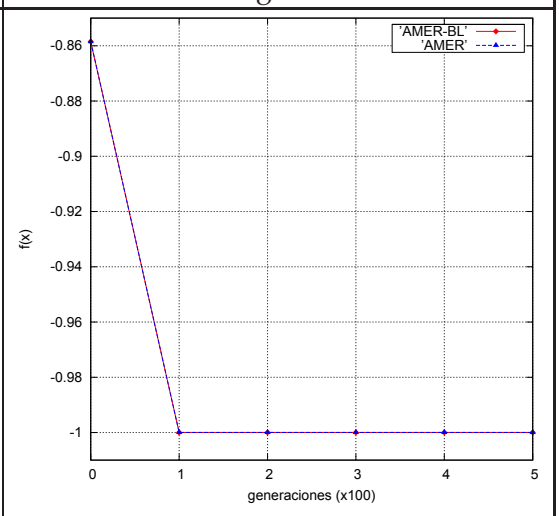

g09



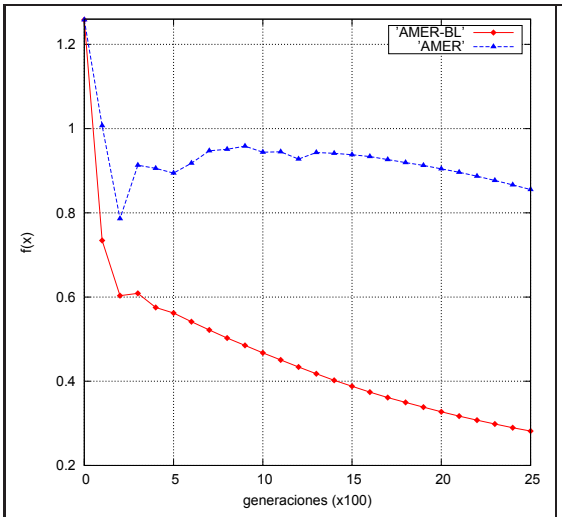

g10



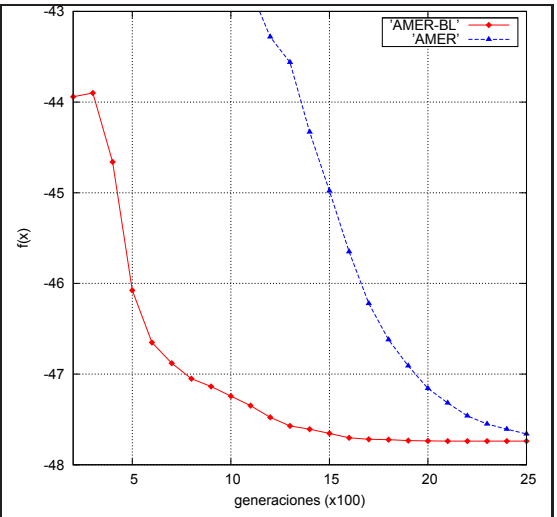

g11



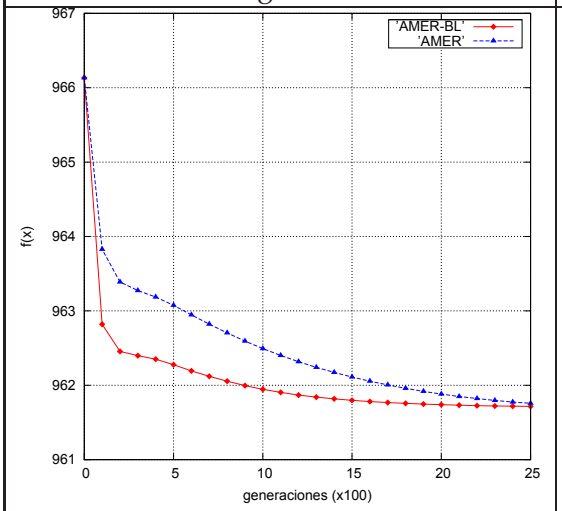

g12



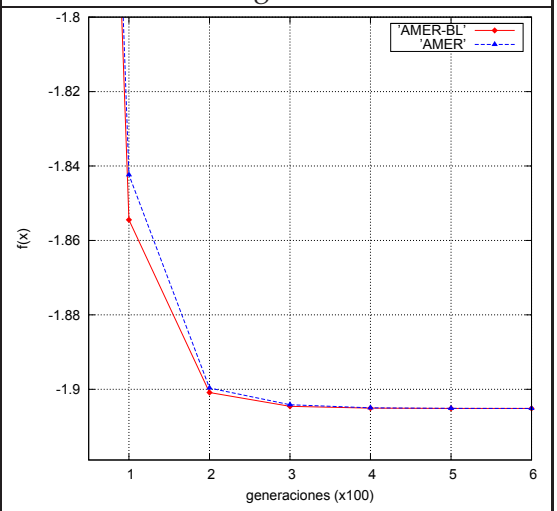

g13



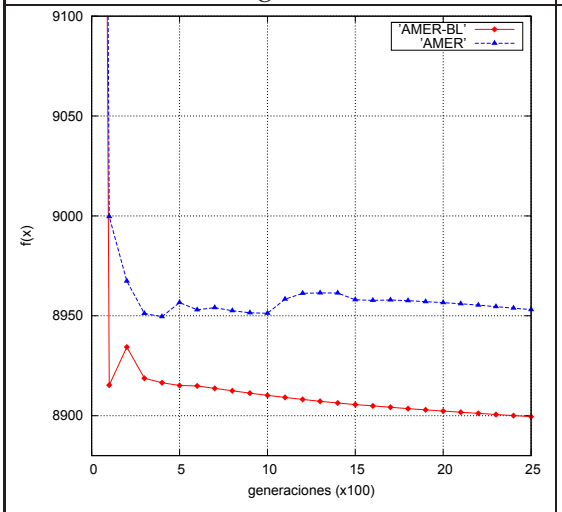

g14



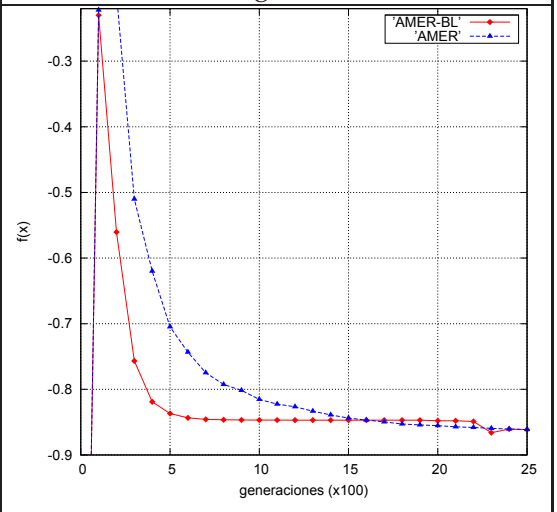

g15



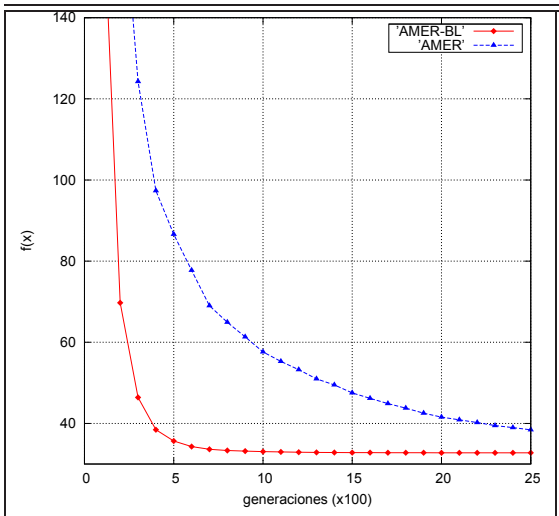

g16



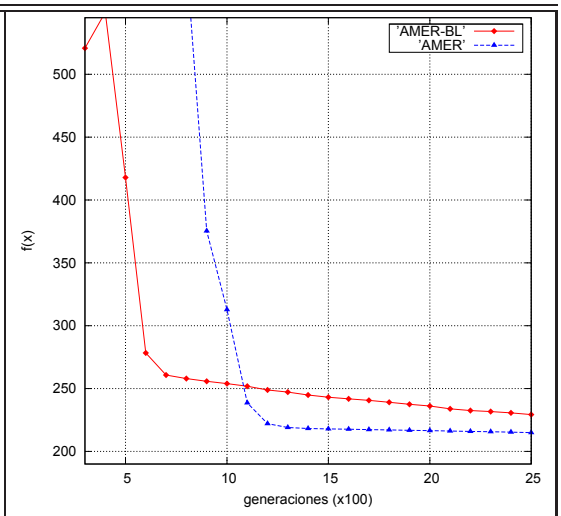

g17



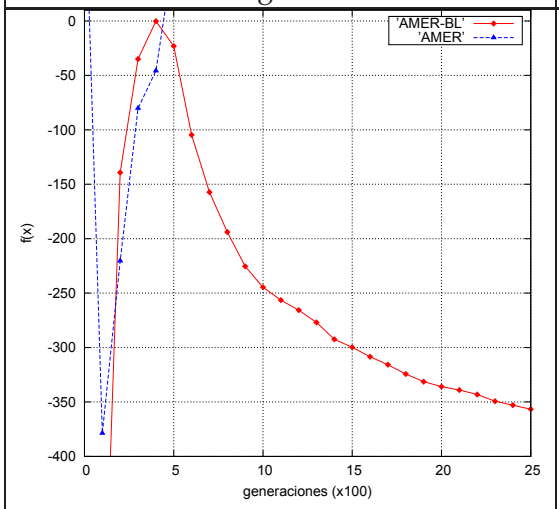

g18



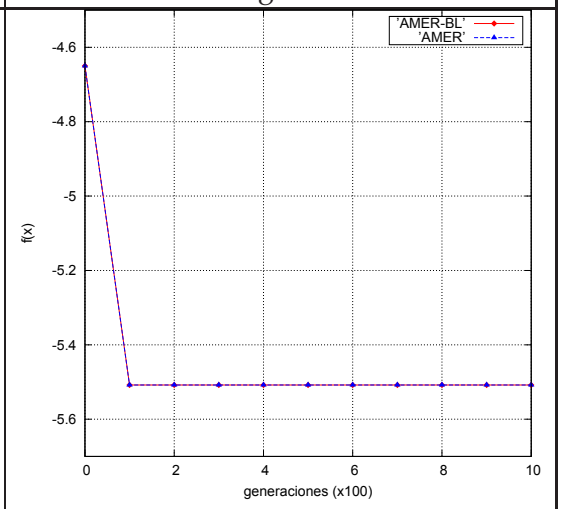

g19



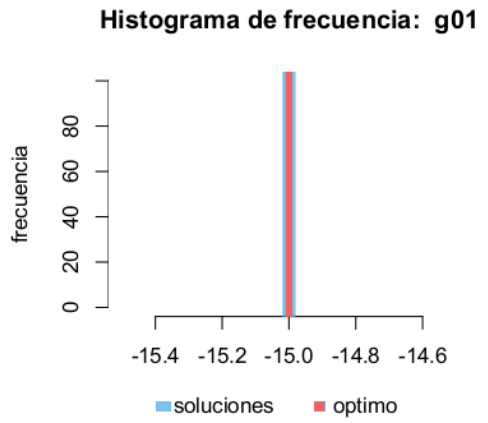

g21



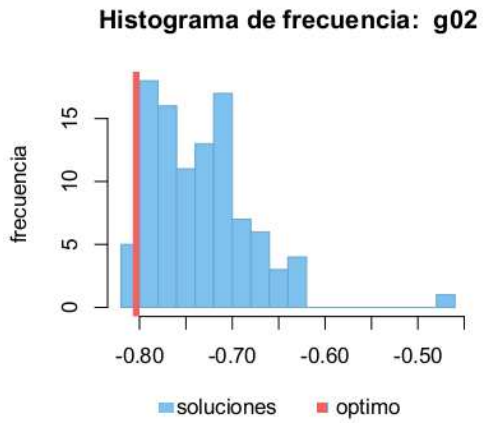

g23



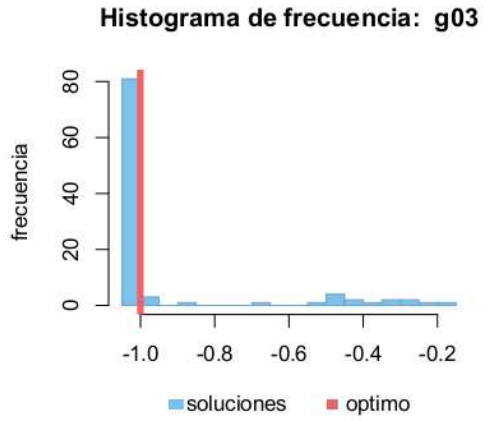

g24



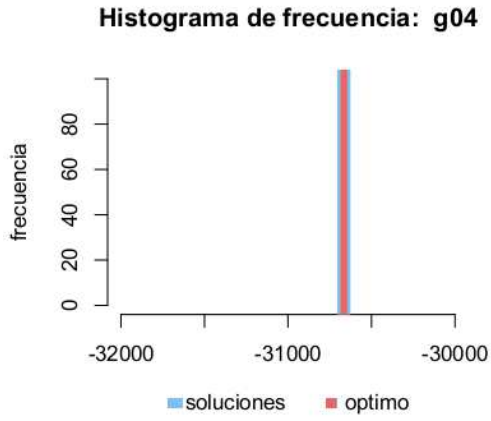

g01



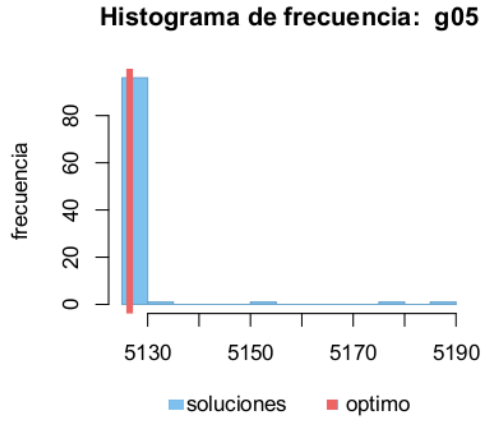

g02



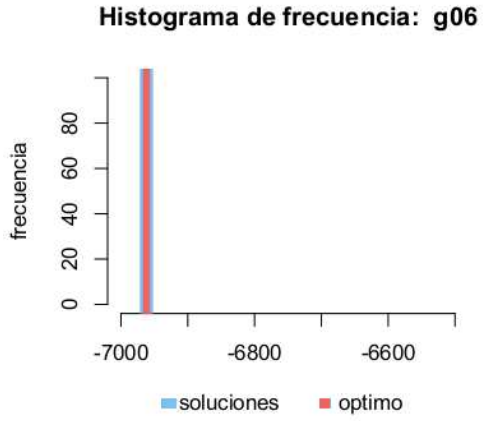

g03



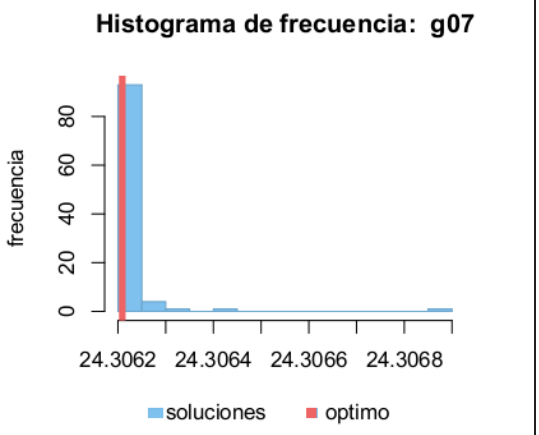

g04



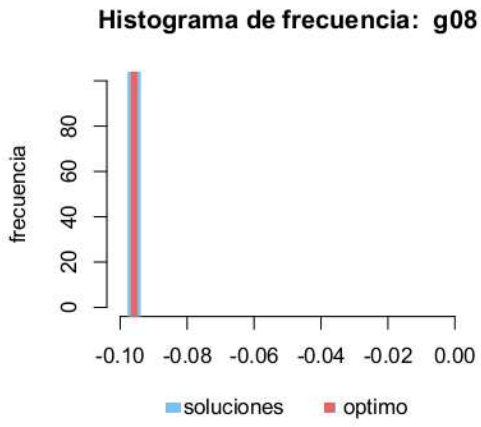

g05



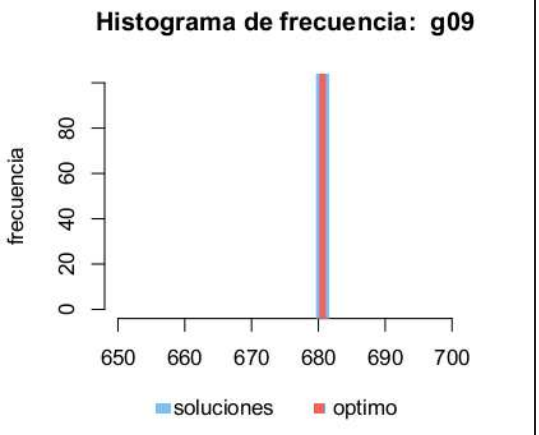

g06



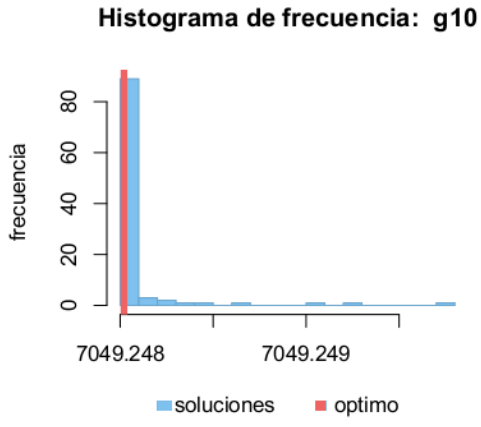

g07



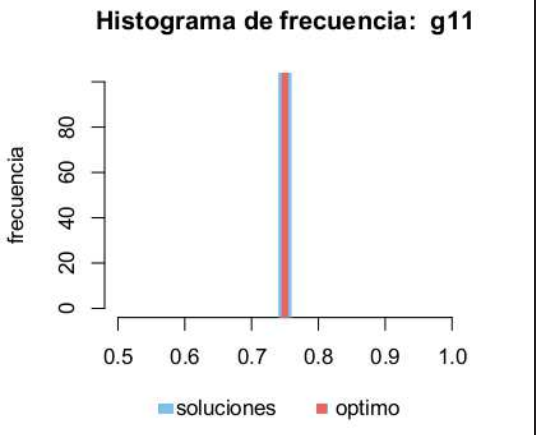

g08



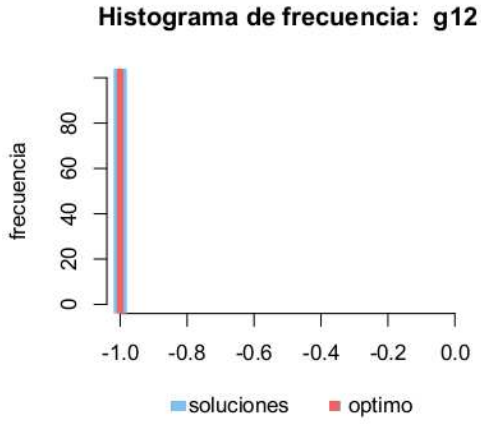

g09



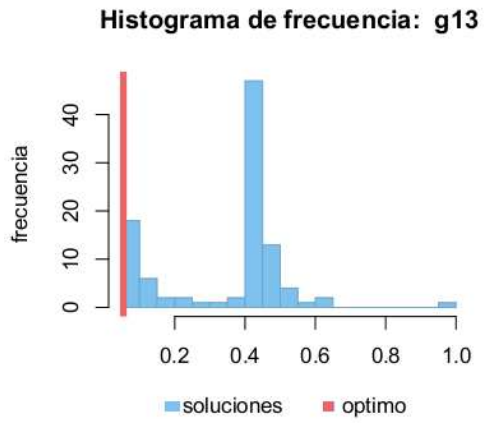

g10



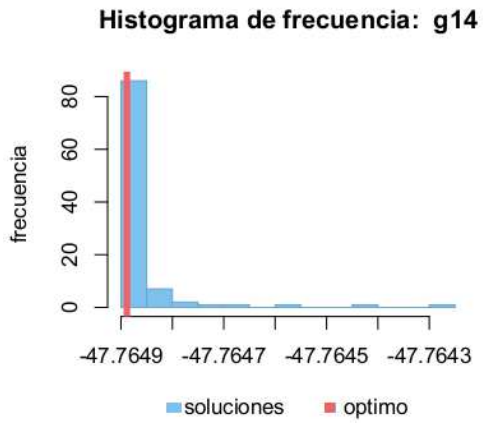

g11



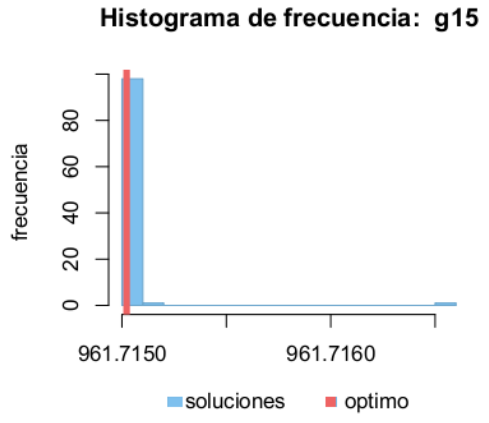

g12



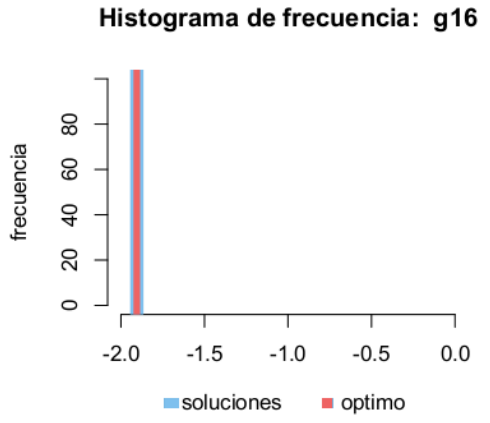

g13



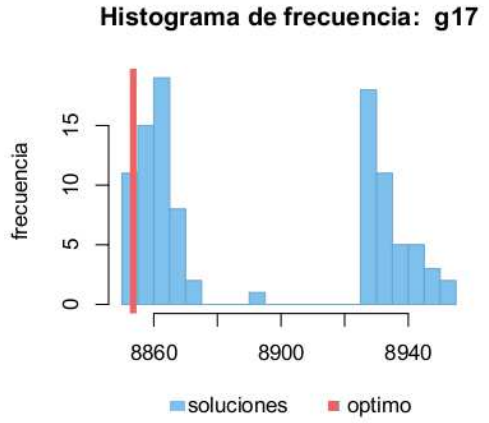

g14



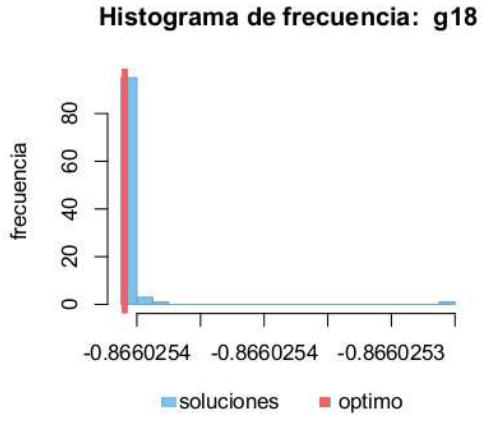

g15



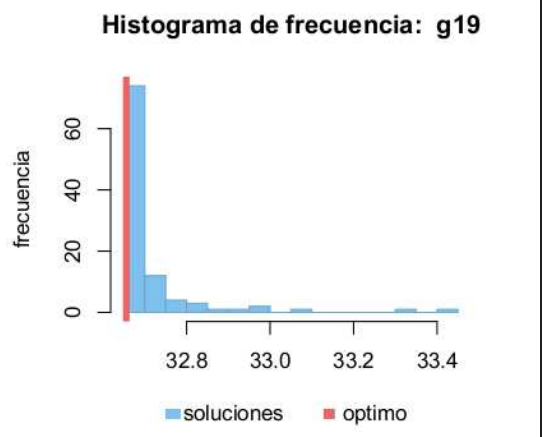

g16



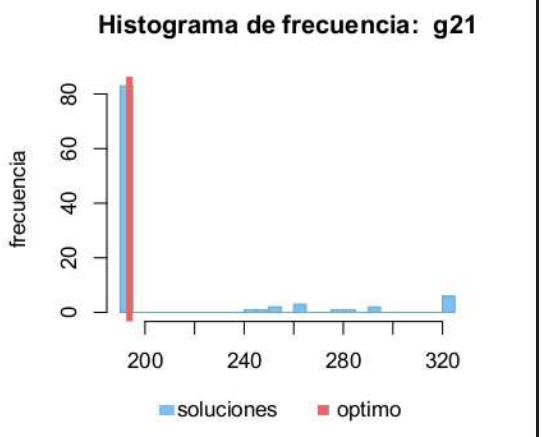

g17



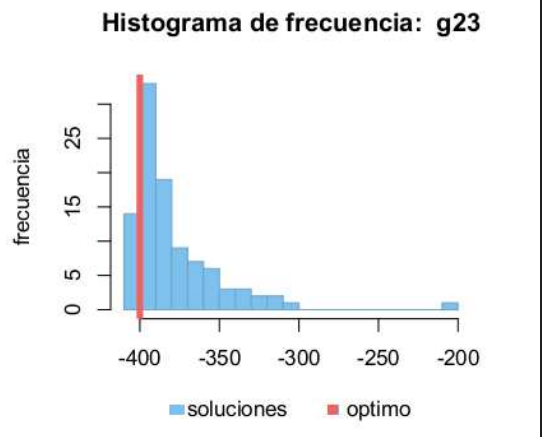

g18



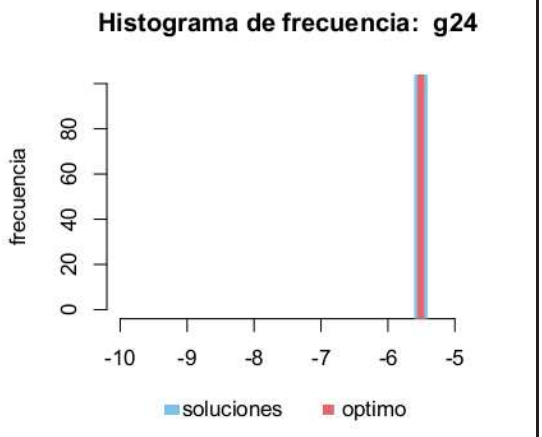

g19




g21




g23

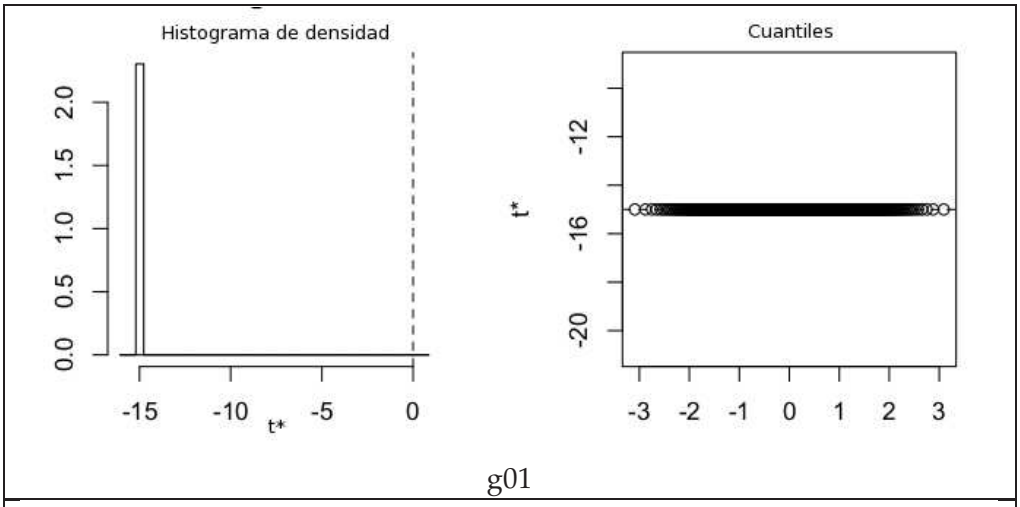



g24

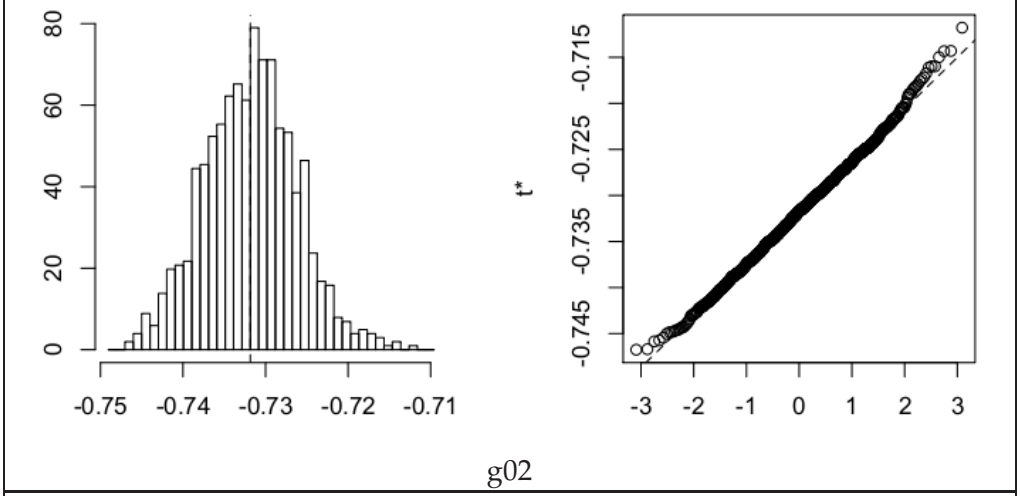
## B.5. Histograma de densidad y gráficas de cuantiles de bootstrap

Presentamos los histogramas de densidad y gráficas de cuantiles de *bootstrap* después de 1000 remuestreos y aplicando un intervalo de confianza del 95 %.

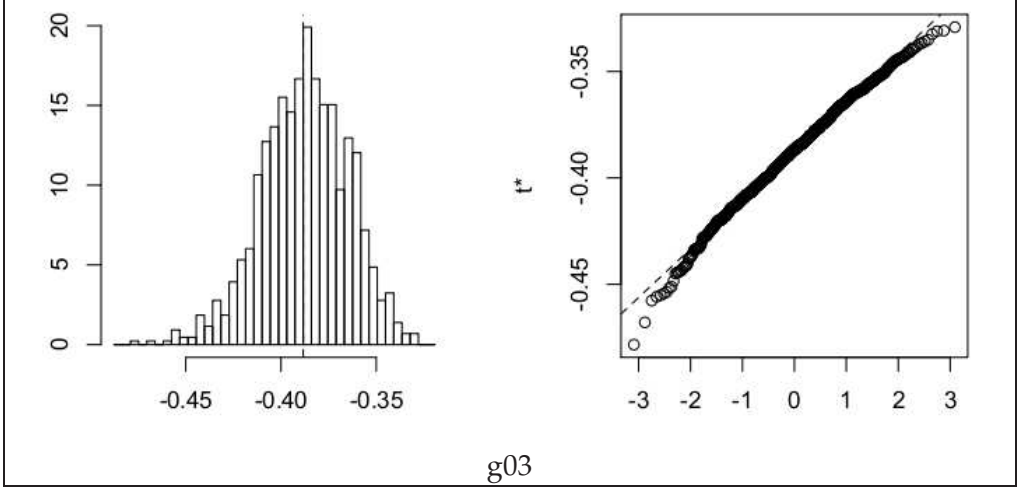
**Histograma de densidad y gráfica de cuantiles**




g01

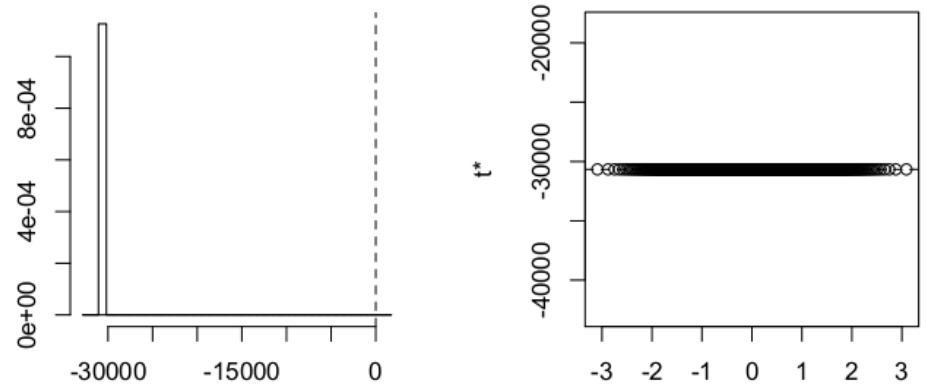



g02

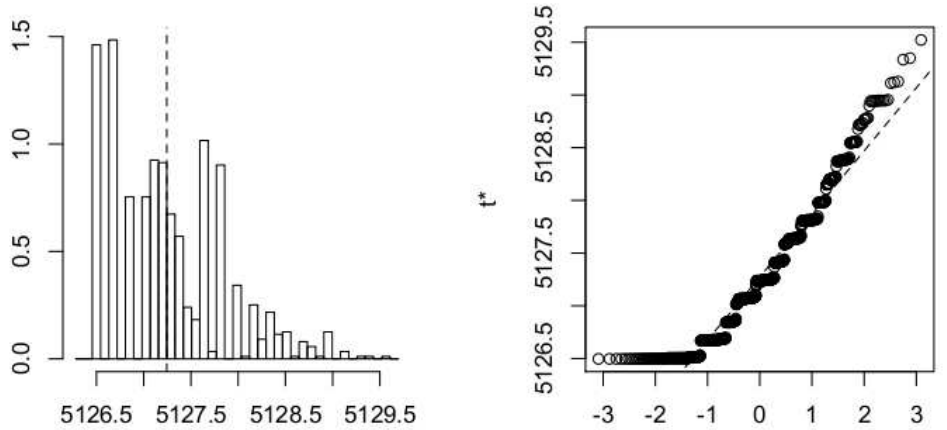



g03

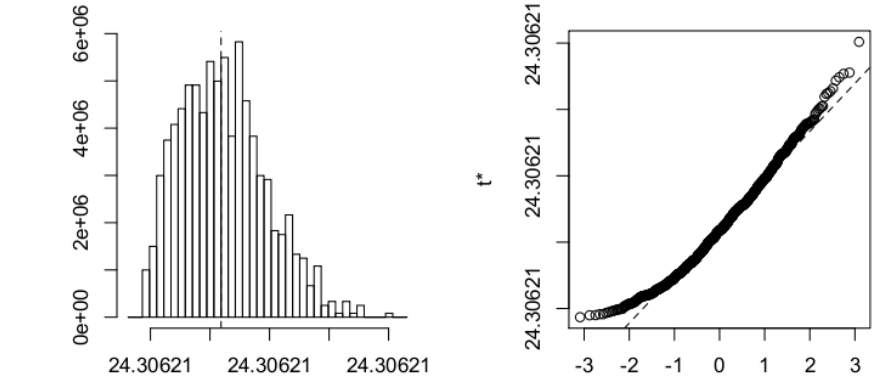
**Histograma de densidad y gráfica de cuantiles**




g04

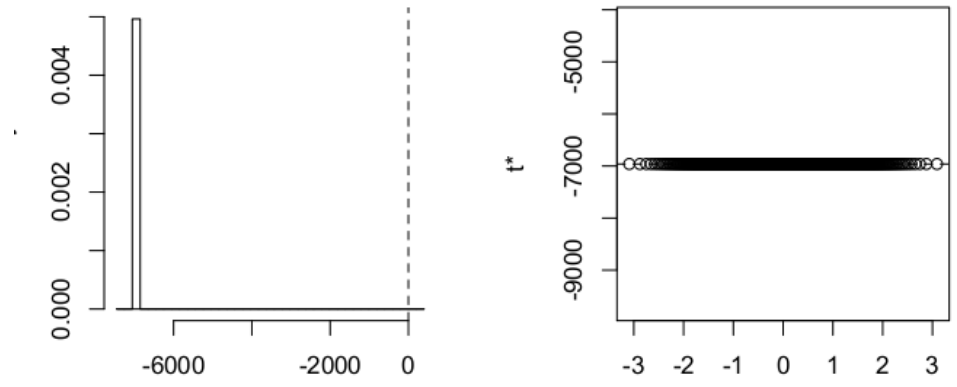



g05

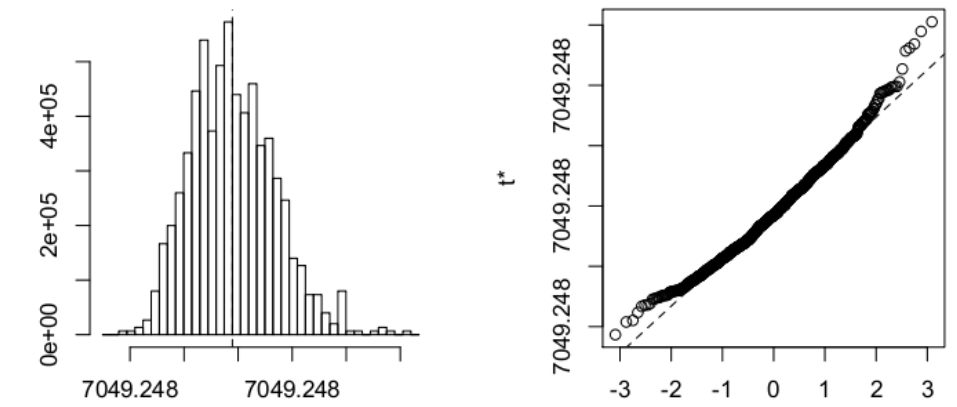



g07

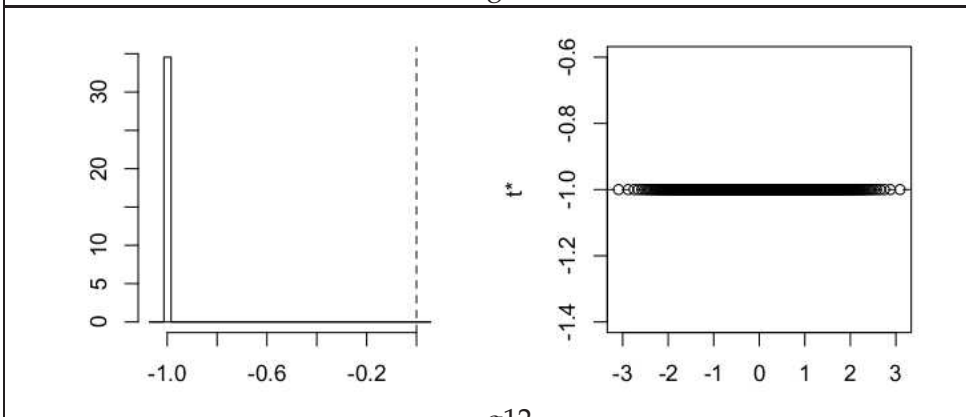
**Histograma de densidad y gráfica de cuantiles**




g09

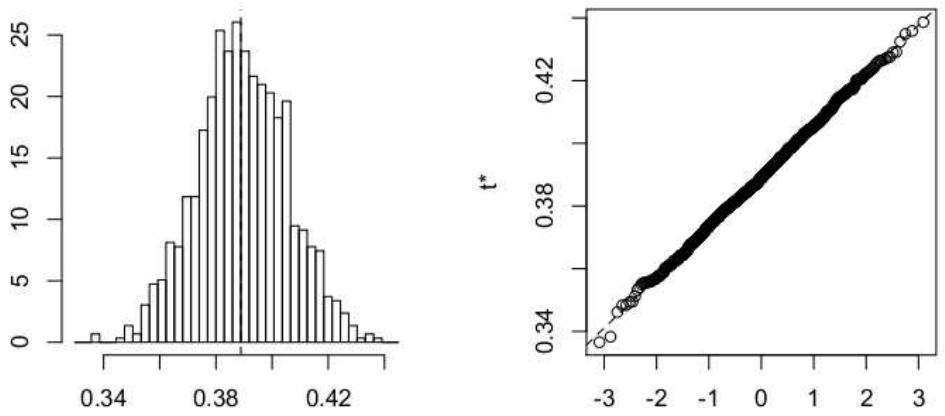



g10

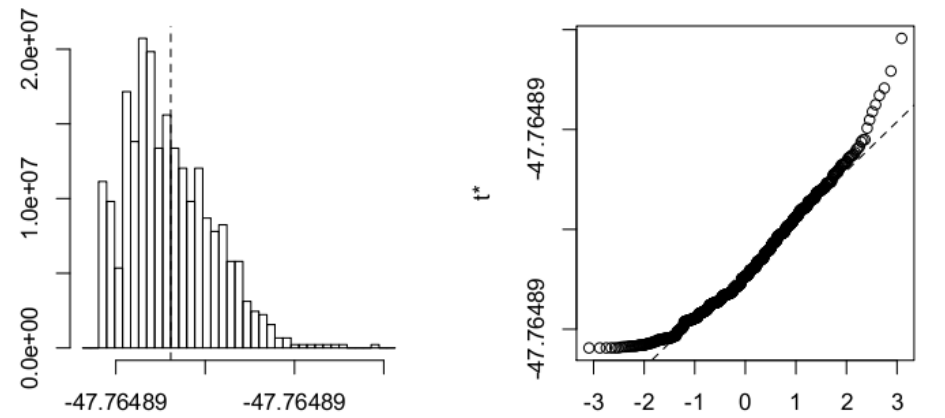



g12

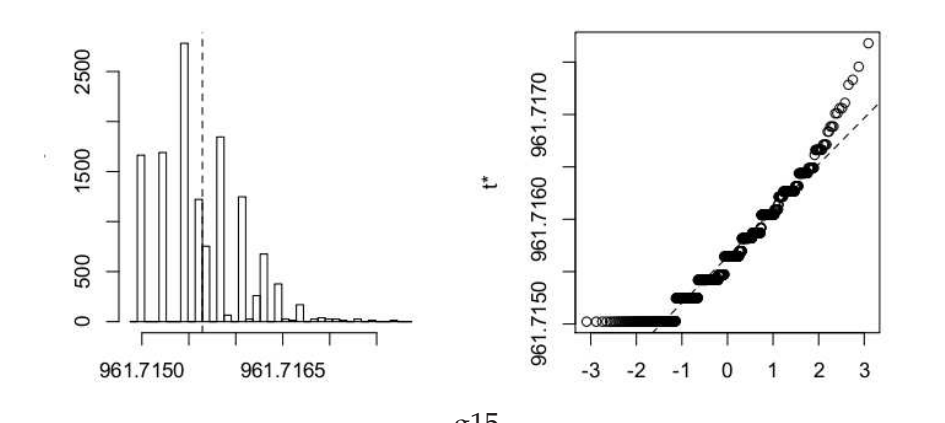
**Histograma de densidad y gráfica de cuantiles**




g13

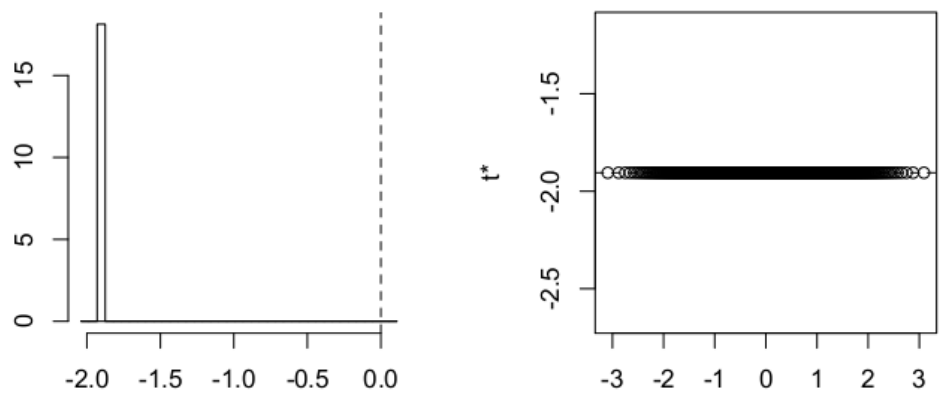



g14

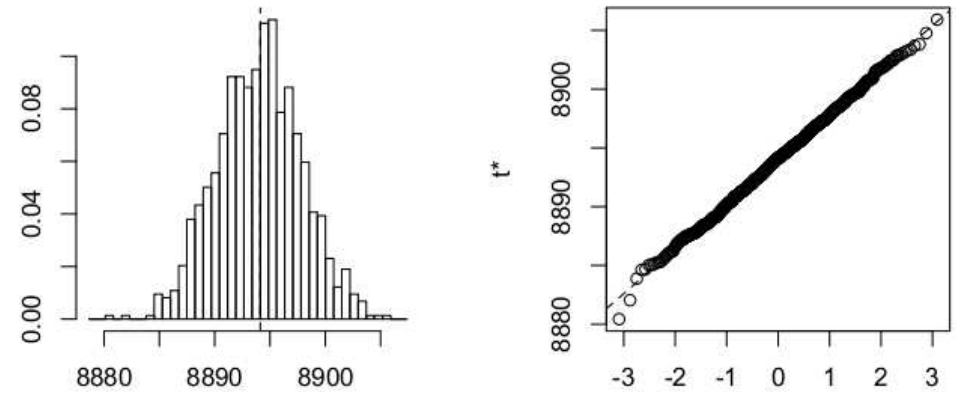



g15

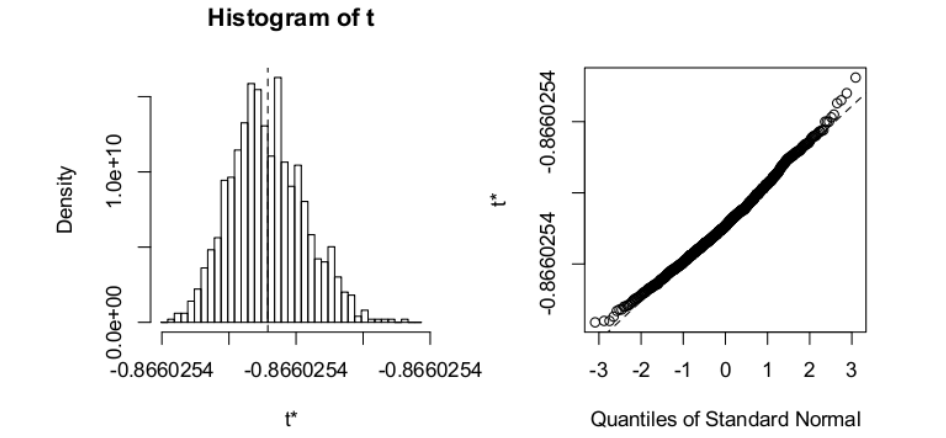
**Histograma de densidad y gráfica de cuantiles**




g16

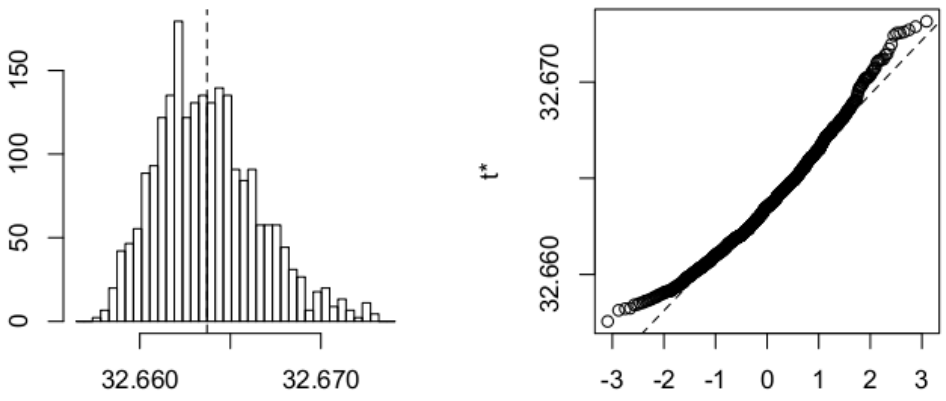



g17

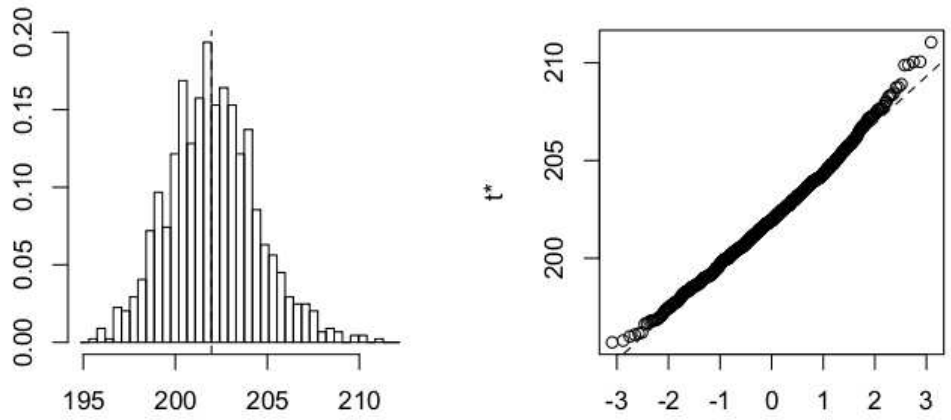



g18

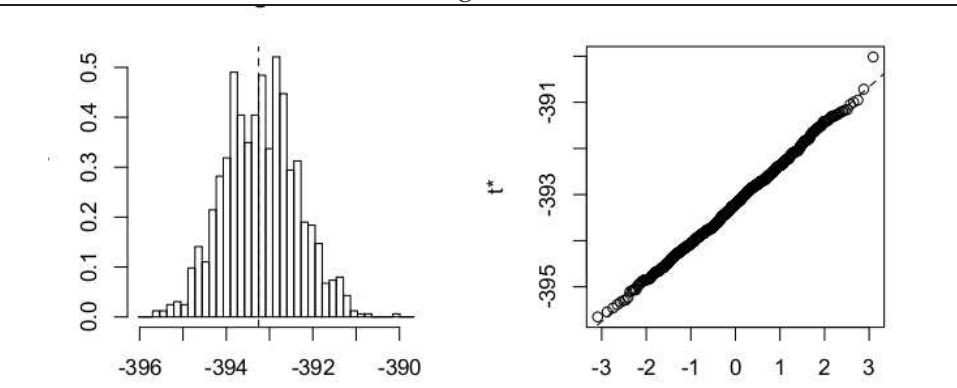
**Histograma de densidad y gráfica de cuantiles**




g19




g21




g23

# Bibliografía


[1] Ali, M. M., and Kajee-Bagdadi, Z. A local exploration-based differential evolution algorithm for constrained global optimization. *Applied Mathematics and Computation 208* (2009), 31–48.

[2] Bäck, T., Schütz, M., and Khuri, S. *Evolution strategies: An alternative evolution computation method.*, vol. 1063/1996. Springer Berlin / Heidelberg, Lecture Notes in Computer Science, 1996.

[3] Bambha, N. K., Bhattacharyya, S. S., Teich, J., and Zitzler, E. Systematic integration of parameterized local search. *evolutionary algorithms, IEEE Transactions on Evolutionary Computation 8*, 2 (2004), 137–155.

[4] Beasley, D., Bull, D., and Martin, R. Reducing epistasis in combinatorial problems by expansive coding. *In Stephanie Forrest, editor, Proceedings of the Fifth International Conference on Genetic Algorithms* (1993), 400–407.

[5] Berg, M., Cheong, O., van Kreveld, M., and Overmars, M. *Computational Geometry, Algorithms and applications*, 3 ed. Department of Computer Science KAIST Gwahangno Daejeon Korea, 2008.

[6] Bertsekas, D. P. *Constrained Optimization and Lagrange multiplier methods*. Massachusetts Institute of Technology, At hena Scientific, Belmont , Massachusetts, Inc., in 1982.

[7] Cannon, W. D. *The Wisdom of the Body.* W. W. Norton and Company, Inc., New York, 1963.

[8] Caruana, R., and Schaffer, J. D. Representation and hidden bias: Gray vs. binary coding for genetic algorithms. *In Proceedings of the Fifth International Conference on Machine Learning* (1988), 132–161.

[9] Chiu, K., Shirley, P., and ChangyawWang. Multi-jittered sampling. in graphics gems. *AP Professional IV* (1994), 370–374.

[10] Coello Coello, C. A. Theoretical and numerical constraint handling techniques used with evolutionary algorithms: A survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering 191* (2002), 1245–1287.

[11] Cohen, P. R. *Empirical Methods for Artificial Intelligence*. The MIT Press Cambridge, Massachusetts, Cambridge, MA, USA, 1995.

[12] Davis., L. Adapting operator probabilities in genetic algorithms. *In J. David Schaffer, editor, Proceedings of the Third International Conference on Genetic Algorithms* (1989), 61–69.

[13] Davis., L. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York, 1991.

[14] Davison, A., and Hinkley, D. *Bootstrap Methods and their Application*. Cambridge Series in Statistical and Probabilistic Mathematics, Cambridge, University Press, 2005.

[15] Deb., K. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering* (2000), 186(2/4):311–338.

[16] Deb, K. *Optimization for Engineering Design: Algorithms and Examples*. Prentice-Hall of India Pvt. Ltd., 2002.

[17] Eberhart, R. C., and Kennedy, J. A new optimizer using particle swarm theory. *Proceedings of the Sixth International Symposium on Micro Machine and Human Science* (1995), 39–43.

[18] Edwin, K., Chong, P., and Stanislaw, H. Z. *An Introduction to optimization*. Editorial Wiley-Interscience Series in Discrete Mathematics and Optimization, New York, 2001.

[19] Efron, B. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics 7*, 1 (1979), 1–26.

[20] Eshelman, L. J., Mathias, K. E., and Schaffer, J. D. Crossover operator biases: Exploiting the population distribution. In *Proceedings of the Seventh International Conference on Genetic Algorithms* (1997), pp. 354–361.

[21] Eshelman, L. J., and Schaffer, J. D. *Real-coded Genetic Algorithms and Interval-Schemata.* In L. Darrell Whitley, editor, Foundations of Genetic Algorithms 2, Morgan Kaufmann Publishers, San Mateo, California, 1993.

[22] Fogel, D. B. *Evolutionary Computation. The Fossil Record. Selected Readings on the History of Evolutionary Algorithms*. The Institute of Electrical and Electronic Engineers, New York, 1998.

[23] Fogel, D. B., and Stayton, L. C. On the effectiveness of crossover in simulated evolutionary optimization. *BioSystems 32* (1994), 171–182.

[24] Fogel, L. J. *Artificial Intelligence through Simulated Evolution*. JohnWiley, New York, 1966.

[25] Fogel, L. J., Owens, A. J., and Walsh, M. J. Artificial intelligence through a simulation of evolution. *InM.Maxfield, A. Callahan, and L. J. Fogel, editors, Biophysics and Cybernetic Systems: Proceedings of the Second Cybernetic Sciences Symposium* (1965), 131–155.

[26] Glover, F. Tabu search. *ORSA Journal on Computing 1*, 3 (1989), 190–206.

[27] Goldberg, D. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Longman Publishing Co, Boston, MA, USA, 1989.

[28] Gómez, A., and Bodmann, B. Neighborhood search considering continuos space search. *Tendencias em Matemática Aplicada e Computacional 3*, 2 (2002), 111–119.

[29] Hannan, B. L., Zhang, M. X., and Zhou, Y. A memetic co-evolutionary differential evolution algorithm for constrained optimization. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on Issue* (2007), pp. 2996 – 3002.

[30] Hansen, P., Jaumard, J., Mladenovíc, N., and Parreira, A. *Variable neighbourhood search for maximum weight satisfiability problem.* Les Cahiers du GERAD, 2000.

[31] Hansen, P., Mladenovic, N., and Moreno, P. J. Búsqueda de entorno variable. *Revista Iberoamericana de Inteligencia Artificial 19* (2003), 77–92.

[32] Herrera, F., and Lozano, M. Two-loop real-coded genetic algorithms with adaptive control of mutation step sizes. *Applied Intelligence 13* (2000), 187–204.

[33] Herrera, F., Lozano, M., and Verdegay, J. L. Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artificial Intelligence Review 12*, 4 (1998), 265–319.

[34] Hinterding, R., and Michalewicz, Z. Your brains and my beauty: Parent matching for constrained optimisation. *In Proceedings of the 5th International Conference on Evolutionary Computation* (1998), 810–815.

[35] Hoffman, A. *Arguments on Evolution: A Paleontologist's Perspective.* Oxford University Press, New York, 1989.

[36] Holland., J. H. Outline for a logical theory of adaptive systems. *Journal of the Association for Computing Machinery* (1962), 9:297–314.

[37] Holland, J. H. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, Michigan, 1975.

[38] Homaifar, A., Lai, H., and Qi, X. Constrained optimization via genetic algorithms. *Simulation.* (1994), 62(4):242–254.

[39] Hooke, R., and Jeeves, T. A. Direct search solution of numerical and statistical problems. *Journal of the ACM 8* (1961), 212 – 229.

[40] James, C. B. Genetics and random keys for sequencing and optimization. *ORSA Journal on Computing* (1994), 6(2):154–160.

[41] Joines, J., and Houck., C. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with gas. *In David Fogel, editor, Proceedings of the first IEEE Conference on Evolutionary Computation* (1994), 579–584.

[42] Jong, K. D. An analysis of the behavior of a class of genetic adaptive systems. *Doctoral dissertation* (1975).

[43] jun Shi, Y., fei Teng, H., and qiang Li, Z. Cooperative co-evolutionary differential evolution for function optimization. In *ICNC (2), Lecture Notes in Computer Science* (2005), vol. 3611/2005, pp. 1080–1088.

[44] Kita, H., and Yamamura, M. A functional specialization hypothesis for designing genetic algorithms. In *IEEE International Conference on Systems, Man, and Cybernetics* (Piscataway, New Jersey, 1999), IEEE Press, pp. 579–584.

[45] Kolda, T., Lewis, R., and Torczon, V. Optimization by direct search: new perspective on classical and modern methods. *SIAM Review 45*, 3 (2003), 385–482.

[46] Kowalczyk., R. Constraint consistent genetic algorithms. *In Proceedings of the 1997 IEEE Conference on Evolutionary Computation* (1997), 343–348.

[47] Koziel, S., and Michalewicz., Z. A decoder-based evolutionary algorithm for constrained parameter optimization problems. *In T. Bäck, A. E. Eiben, M. Schoenauer, and H.-P. Schwefel, editors, Proceedings of the 5th Parallel Problem Solving from Nature (PPSN V)* (1998), 231–240.

[48] Koziel, S., and Michalewicz., Z. Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evolutionary Computation* (1999), 7(1):19–44.

[49] Krasnogor, N., and Smith, J. Emergence of profitable search strategies based on a simple inheritance mechanism. In *International Genetic and Evolutionary Computation Conference (GECCO2001)* (San Francisco, CA, 2001), Morgan Kaufmann Publishers, pp. 432–439.

[50] Lozano, M., Herrera, F., Krasnogor, N., and Molina, D. Real-coded memetic algorithms with crossover hill-climbing. *Evolutionary Computation 12*, 3 (2004), 273–302.

[51] Lucasius, C. B., and G., K. Applications of genetic algorithms in chemometrics. *Proc. of the Third International Conference on Genetic Algorithms, J. David Schaffer (Ed.)* (1989), 170–176.

[52] Luo, C., and Yu, B. Low dimensional simplex evolution–a hybrid heuristic for global optimization. In *Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing* (DC, USA, 2007), vol. 2, IEEE Computer Society Washington, pp. 470–474.

[53] McKay, M. D., Beckman, R. J., and Conover, W. J. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics 1*, 42 (2000), 55–61.

[54] Mendel, G. J. Versuche ber pflanzen-hybriden. *In Verhandlungen des naturforschenden Vereines in Brunn 1* (1985), 3–47.

[55] Mezura Montes, E., Clerc, M., Suganthan, P. N., Coello Coello, C. A., Liang, K. D. J. J., and Runarsson, T. P. Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization. In *Technical report* (September 2006.).

[56] Michalewicz., Z. Genetic algorithms + data structures = evolution programs. *Springer-Verlag, second edition* (1992).

[57] Michalewicz, Z. A survey of constraint handling techniques in evolutionary computation methods. *In J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, editors, Proceedings of the 4th Annual Conference on Evolutionary Programming* (1995), 135–155.

[58] Michalewicz, Z., and Nazhiyath., G. Genocop iii: A coevolutionary algorithm for numerical optimization with nonlinear constraints. *In David B. Fogel, editor, Proceedings of the Second IEEE International Conference on Evolutionary Computation* (1995), 647–651.

[59] Michalewicz, Z., and Schoenauer., M. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation* (1996), 4(1):1–32.

[60] Miettinen, K., Makela, M., and Toivanen, J. Numerical comparison of some penalty based constraint handling techniques in genetic algorithm. *Journal of Global Optimization 27* (2003), 427–446.

[61] Montgomery, D. C. *Design and analysis of experiments*. John Wiley, USA, 2001.

[62] Moscato, P. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Technical Report C3P Report 826* (1989).

[63] Mühlenbein, H., and Schlierkamp-Voosen, D. Predictive models for the breeder genetic algorithm i. continuous parameter optimization. *Evolutionary Computation 1* (1993), 25–49.

[64] Nelder, J. A., and Mead, R. A simplex method for function minimization. *The Computer Journal 7* (1965), 308–313.

[65] Nguyen, Q. H., Ong, Y. S., Lim, M. H., and Krasnogor, N. A study on the design issues of memetic algorithm. In *Proceedings of the 2007 IEEE Congress on Evolutionary Computation* (2007), IEEE, pp. 2390–2397.

[66] Noman, N., and Iba, H. Accelerating differential evolution using an adaptive local search. *IEEE Trans. Evolutionary Computation 12*, 1 (2008), 107–125.

[67] Paredis., J. Co-evolutionary constraint satisfaction. *In Proceedings of the 3rd Conference on Parallel Problem Solving from Nature* (1994), 231–240.

[68] Potter, M., and Jong, K. D. A cooperative co-evolutionary approach to function optimization. In *Proceedings of the third parallel problem solving from nature* (Jerusalem, Israel, 1994).

[69] Powell, D., and Skolnick, M. M. Using genetic algorithms in engineering design optimization with non-linear constraints. *In Stephanie Forrest, editor, Proceedings of the Fifth International Conference on Genetic Algorithms* (1993), 424–431.

[70] Price, K. *An introduction to Differential Evolution. In New Ideas in Optimization*. D. Corne, M. Dorigo and F. Glover, Eds. MacGraw-Hill, 1999.

[71] Pólya, G. *How to Solve It: A New Aspect of Mathematical Method*. Penguin Science, stamford, uk, United Kingdom, 1990.

[72] Rao, S. *Engineering Optimization: Theory and Practice*. John Wiley and Sons, 1996.

[73] Ray, T., Kang, T., and Chye., S. K. An evolutionary algorithm for constrained optimization. *In Darrell Whitley, David Goldberg, Erick Cantú-Paz, Lee Spector, Ian Parmee, and Hans-Georg Beyer* (2000), 771–777.

[74] Reeves, C. B. *Modern Heuristic Techniques for Combinatorial Problems.* John Wiley and Sons, Inc. New York, NY, USA, Coventry Univ., Coventry, UK, 1993.

[75] Richardson, J. T., Palmer, M. R., Liepins, G., and Hilliard, M. Some guidelines for genetic algorithms with penalty functions. *In J. David Schaffer, editor, Proceedings of the Third International Conference on Genetic Algorithms* (1989), 191–197.

[76] Runarsson, T. P., and Yao, X. Stochastic ranking for constrained evolutionary optimization. *IEEE Trans. Evolutionary Computation 4*, 3 (2000), 284–294.

[77] Runarsson, T. P., and Yao., X. Search biases in constrained evolutionary optimization. *IEEE Transactions on Systems, Man and Cybernetics 35* (2005), 2:233–243.

[78] Sawaragi, Y., Nakayama, H., and Tanino, T. *Theory of Multiobjective Optimization*. (vol. 176 of Mathematics in Science and Engineering), Orlando, FL: Academic Press Inc. ISBN: 0126203709., 1985.

[79] Schoenauer, M., and Michalewicz., Z. Evolutionary computation at the edge of feasibility. *In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, Proceedings of the Fourth Conference on Parallel Problem Solving from Nature (PPSN IV)* (1996), 245–254.

[80] Schoenauer, M., and Xanthakis., S. Constrained ga optimization. *In Stephanie Forrest, editor, Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA-93)* (1993), 573–580.

[81] Schwefel, H.-P. Numerical optimization of computer models. *Wiley, Chichester, UK* (1981).

[82] Shi, and Eberhart., R. A modified particle swarm optimizer. *In Proceedings of the IEEE Congress on Evolutionary Computation* (1998), 69–73.

[83] Smith, J., and Fogarty, T. Operator and parameter adaptation in genetic algorithms. *Soft Computing - A Fusion of Foundations, Methodologies and Applications 1*, 2 (june, 1997), 81–87.

[84] Spears, W. Crossover or mutation? *In Whitley, L.D.* (1993), 221–238.

[85] Spendley, G. R. H. W., and Himsworth, F. R. Sequencial application of simplex designs in optimization and evolutionary operation. *Technometrics 4*, 4 (November 1962), 441–461.

[86] Steuer, R. *Multiple Criteria Optimization: Theory, Computations, and Application*. John Wiley and Sons, Inc. ISBN: 047188846X, 1986.

[87] Storn, R., and Price, K. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. In *Technical Report, TR-95-012* (Berkeley, CA, 1995).

[88] Storn, R., and Price., K. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization 11*, 4 (1997), 341–359.

[89] Surry, P., and Rsdcliffe, N. Real representations, foundations of genetic algorithms 4. In *Morgan Kaufman* (San Francisco, 1996), pp. 343–363.

[90] Syswerda, G. Uniform crossover in genetic algorithms. *In Proceedings of the 3rd International Conference on Genetic Algorithms* (1989), 2–9.

[91] Syswerda, G. Schedule optimization using genetic algorithms. In *In Lawrence Davis, editor, Handbook of Genetic Algorithms* (Van Nostrand Reinhold, New York, New York, 1991), pp. 332–349.

[92] Tsutsui, S., Yamamura, M., and Higuchi, T. Multi-parent recombination with simplex crossover in real coded genetic algorithms. *Genetic Evol. Comput. Conf. (GECCO'99)* (1999), 657–664.

[93] Turing, A. M. Computing machinery and intelligence. *Mind 59* (1950), 433–460.

[94] Velázquez Reyes, J. Propuesta de evoluci´on diferencial para optimización de espacios restringidos. *Master's thesis* (2006).

[95] Vries, H. D. Species and varieties: Their origin by mutation. *Open Court* (1906).

[96] Wolpert, D., and Macready, W. G. No free lunch theorems for optimization. *IEEE Trans. Evolution Computation 1* (1997), 67–82.

[97] Wright, A. H. *Genetic Algorithms for Real Parameter Optimization*. In Gregory J. E. Rawlins, editor, Foundations of Genetic Algorithms, Kaufmann Publishers, San Mateo, California, 1991.