Centro de Investigación y de Estudios
Avanzados del Instituto Politécnico Nacional

Departamento de Computación

Algoritmo híbrido para resolver problemas de optimización con restricciones

Tesis que presenta

Adriana Menchaca Méndez

para obtener el Grado de

Maestro en Ciencias

en la Especialidad de

Computación

Director de la Tesis

Dr. Carlos A. Coello Coello

México, D. F. **Noviembre 2008**

Resumen

En el mundo en que vivimos existe una enorme cantidad de problemas de optimización con restricciones prácticamente en todas las áreas del conocimiento. En esta tesis se plantea una mejora a los algoritmos con manejo de restricciones. Esto en sí es una aportación importante debido a su extensa aplicabilidad.

El modelo matemático de un problema de optimización con restricciones está conformado por una función objetivo, f, y un conjunto de funciones de restricciones de igualdad y desigualdad, h y g, respectivamente. Dichas funciones son funciones escalares de las variables del problema: $f : \mathbb{R}^n \to \mathbb{R}$, $h : \mathbb{R}^n \to \mathbb{R}$ y $g : \mathbb{R}^n \to \mathbb{R}$. El subconjunto de todos los puntos que satisfacen las restricciones del problema es llamado región factible.

El manejo de restricciones es una tarea díficil, en la que las técnicas de programación matemática tienen varias limitaciones, por ejemplo, cuando la función objetivo y/o las restricciones son no diferenciables o discontinuas o no se pueden expresar de manera algebraica, o cuando la región factible es disjunta. Los Algoritmos Evolutivos (AEs) han podido resolver funciones que tienen estas características. Sin embargo, no cuentan con un mecanismo explícito para manejo de restricciones. El problema principal de las técnicas para el manejo de restricciones en los AEs consiste en lograr el balance correcto entre dar prioridad a la minimización del valor de la función objetivo o darle prioridad a la minimización de la violación a las restricciones.

En esta tesis, se propone un nuevo criterio de selección entre soluciones candidatas a un problema de optimización con restricciones y se incorpora al algoritmo de Evolución Diferencial (ED). El algoritmo resultante es llamado *Evolución Diferencial para Problemas de optimización con Restricciones* (EDPR). Dicho algoritmo es comparado contra dos de los AEs más conocidos en la literatura especializada: la jerarquización estocástica y Diversity-DE.

Finalmente, teniendo como objetivo mejorar los resultados obtenidos con el algoritmo EDPR se propone un algoritmo híbrido basado en el algoritmo EDPR y el método de Nelder-Mead llamado *Híbrido de Evolución Diferencial y el método Simplex para Problemas de optimización con Restricciones* (HEDSPR). La idea de este híbrido surge del hecho de que los AEs son capaces de lidiar con problemas de optimización complejos que las técnicas clásicas no pueden resolver. Sin embargo, los AEs

tienen un costo computacional mucho mayor que las técnicas clásicas. Por tal motivo, se buscó usar el método de Nelder-Mead para acelerar la convergencia a soluciones de buena calidad. Los resultados obtenidos con este algoritmo son comparados contra los resultados del EDPR, evidenciando las ventajas que el híbrido propuesto nos puede ofrecer en algunos casos.

Abstract

In the world in which we live, there exist numerous optimization problems, practically in every application domain. Such problems are subject to a number of constraints, since we always aim to find the best solution given a certain set of circunstances. For that reason, during the last few years, several optimization methods have been developed, in order to deal with such different types of problems. However, an improvement to the existing algorithms means an important contribution (due to their wide applicability).

The mathematical model for an optimization problem with constraints is denoted by an objetive function and a set of equality and inequality constraint functions, h y g, respectively. These functions are scalar functions of the variables of the problem: $f : \mathbb{R}^n \rightarrow \mathbb{R}$, $h : \mathbb{R}^n \rightarrow \mathbb{R}$ y $g : \mathbb{R}^n \rightarrow \mathbb{R}$. The subset of all points that satisfy all constraints is called feasible region.

Constraint handling is a difficult task in which the mathematical programming techniques have several limitations, for instance, when the objective function and/or the constraints are non-differentiable or discontinuous or when they cannot be expressed in algebraic form, or when the feasible region is disjoint. Evolutionary algorithms (EAs) have been able to solve functions with these features. However, EAs do not have an explicit mechanism for handling constraints. The main problem of constraint handling techniques used with EAs consists of achieving the proper balance between giving priority to the minimization of the objective function value or to the minimization of the constraints violation.

In this thesis, we propose a new selection criterion for candidate solutions to a constrained optimization problem, and we incorporate to differential evolution (DE) algorithm. The resulting EA is called "Differential Evolution for Constrained Optimization Problems" (DECOP). Such algorithm is compared against two of the most well-known EAs from the specialized literature: stochastic ranking and Diversity-DE.

Finally, and having as a goal the improvement of the results obtained by our DECOP, we propose a hybrid algorithm based on DECOP and the Nelder-Mead method. This approach is called "Hybrid of Differential Evolution and the Simplex Method for Constrained Optimization Problems" (HDESMCO). Its core idea departs from the fact that EAs are capable of dealing with complex optimization problems that the

classical techniques cannot solve. However, EAs have a much higher computational cost than the classical techniques. Therefore, the intention was to use the Nelder-Mead method to accelerate convergence towards good quality solutions. The results obtained by HDESMCO are compared with respect to the results obtained by DE-COP, making evident what are the advantages that the proposed hybrid approach can offer in some cases.

Agradecimientos

Agradezco profundamente a mi Papá, a mi Mamá, a mis hermanos, a mi tía Moni y a mi tío Edmundo ya que gracias a su cariño, apoyo y enseñanzas, en todos los ámbitos, he logrado ser la persona que hoy soy y de la cual me siento orgullosa.

A Armin y a todos mis amigos, en especial a Miriam y Nidia, pues gracias a ellos he tenido muchos momentos de felicidad e inspiración.

Agradezco sinceramente al Dr. Carlos A. Coello Coello por sus enseñanzas y apoyo pues fueron una parte fundamental para poder concluir esta etapa.

Al Dr. Luis Gerardo de la Fraga y al Dr. Francisco Rodríguez Henríquez por el tiempo que dedicaron a la revisión de esta tesis y porque a través de sus comentarios he obtenido un mejor trabajo.

A todos los Drs. que me impartieron clase durante esta etapa por compartirnos sus conocimientos.

A mis compañeros y a Sofi por su apoyo y amistad pues gracias a ellos tuve una estancia amena.

Al CINVESTAV por permitirme formar parte de esta gran institución.

Al CONACyT por brindarnos un apoyo económico, permitiendo que muchas personas podamos concluir los estudios de maestría.

Esta tesis fue derivada del proyecto CONACyT titulado “Técnicas Avanzadas de Optimización Evolutiva Multiobjetivo” (Ref-45683-Y) cuyo responsable técnico es el Dr. CArlos A. Coello Coello.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Optimización es el acto de obtener el mejor resultado posible dadas ciertas circunstancias. No existe ningún método de optimización para resolver todo tipo de problemas, por lo que se han desarrollado, a lo largo de los años, diferentes métodos para resolver distintas clases de problemas.

En esta tesis nos enfocamos a la resolución de problemas que consisten en encontrar el mínimo de una función de varias variables, las cuales están sujetas a un conjunto de restricciones. Aunque existen diversas técnicas de optimización en la literatura de programación matemática (por ejemplo, basadas en búsquedas directas, en gradientes y en aproximaciones polinomiales [2]) existen problemas para los cuales dichas técnicas resultan inadecuadas debido, por ejemplo, a que la función objetivo no es diferenciable, el espacio de búsqueda es muy accidentado, o la función es discontinua, entre otras posibles razones. Es entonces cuando resulta conveniente utilizar las heurísticas. Para fines de esta tesis, consideramos que una heurística es una técnica de búsqueda y optimización que produce aproximaciones a la solución de problemas complejos en tiempos razonablemente cortos, pero sin garantizar optimilidad.

Los algoritmos evolutivos (AEs) son un ejemplo de las heurísticas y han demostrado ser técnicas muy competitivas en problemas de alta complejidad. Sin embargo, los AEs en su propuesta original, no cuentan con un mecanismo explícito para el manejo de restricciones. Por tal motivo, en los últimos años se ha realizado una amplia investigación para incorporar mecanismos a los AEs que les permitan resolver problemas de optimización con restricciones [3].

1.1. Antecedentes

En el mundo en que vivimos existe una enorme cantidad de problemas de optimización prácticamente en todas las áreas del conocimiento. Por este motivo, desde hace mucho tiempo se han desarrollado diversos métodos de optimización, para lidiar con diversos tipos de problemas.

Los primeros métodos de optimización se remontan a los años 1600. Gracias a Isaac Newton y Gottfried Wilhelm von Leibniz se desarrollaron los métodos de cálculo diferencial; Johann Bernoulli, Leonhard Euler, Joseph-Louis Lagrange y Karl Weierstrass contribuyeron con los fundamentos del cálculo de variaciones; Lagrange desarrolló un método para problemas con restricciones conocido como multiplicadores de Lagrange; Augustin-Louis Cauchy desarrolló el método del descenso empinado, el cual se aplica a problemas sin restricciones [2].

Sin embargo, no fue sino hasta mediados del siglo XX, cuando aparecieron las computadoras digitales, que se implementaron los algoritmos para optimización existentes hasta esos días, impulsándose el desarrollo de nuevos métodos, originándose así varias áreas bien definidas dentro de la optimización.

George Dantzig en 1947 desarrolló el método numérico simplex, para problemas lineales [4]; Richard Bellman en 1957 desarrolló el principio de optimalidad para problemas de programación dinámica [5]; Harold W. Kuhn y Albert W. Tucker en 1951 desarrollaron las condiciones necesarias y suficientes para la solución a un problema de optimización [6], las cuales motivaron a una amplia investigación en optimización no lineal; C. W. Carroll, Anthony V. Fiacco & Garth P. McCormick propusieron las funciones de penalización para resolver problemas con restricciones [7]; R. J. Duffin, C. Zener y E. L. Peterson desarrollaron la programación geométrica [8]; Ralph E. Gomory contribuyó en la programación entera; George Dantzig y Abraham Charnes & William Wager Cooper desarrollaron técnicas de programación estocástica; Charnes y Cooper en 1961 desarrollaron uno de los primeros métodos de optimización multiobjetivo [9]; John von Neumann en 1928 desarrolló los fundamentos de la teoría de juegos la cual, desde entonces, se ha aplicado a diversos problemas, sobre todo en economía [10].

Las heurísticas evolutivas surgen en la segunda mitad del siglo XX con la finalidad de resolver problemas que las técnicas clásicas no pueden, debido sobre todo, al enorme tamaño del espacio de búsqueda o al comportamiento de la función objetivo. Ejemplos de ellas son el recocido simulado [11], los algoritmos genéticos [12], las redes neuronales [13] y, más recientemente, la evolución diferencial [14], los cúmulos de partículas [15], etc.

Debido a la enorme cantidad de problemas de optimización que existen y a los rescursos limitados disponibles, para poder resolverlos, ha sido necesario desde hace ya varios años diseñar algoritmos más eficientes y confiables que sean capaces de encontrar buenas soluciones en lapsos de tiempo razonables. Actualmente, existe ya una gran variedad de algoritmos propuestos que han logrado superar a los anteriores o han resuelto problemas que no podrían ser atacados con las técnicas conocidas hasta ese momento. Sin embargo, es necesario preguntarnos: ¿Podemos mejorar los algoritmos ya existentes que resuelven problemas de optimización con restricciones? En caso de ser posible, esto extiende de manera muy importante su aplicabilidad, dado que los problemas del mundo real suelen tener restricciones.

1.2. Objetivos

El objetivo principal de esta tesis ha sido diseñar e implementar un algoritmo evolutivo hibridizado con una técnica de programación matemática para resolver problemas de optimización con restricciones. La idea de este híbrido surge a partir de que los algoritmos evolutivos son capaces de lidiar con problemas de optimización complejos que las técnicas clásicas no pueden resolver.

Para poder llevar a cabo el objetivo antes descrito, se decidió diseñar, implementar y validar una técnica para el manejo de restricciones usada en un algoritmo evolutivo. En esta tesis se trabaja con el algoritmo de evolución diferencial [14]. Dicha técnica, como se verá más adelante, obtiene resultados competitivos con respecto a dos de las técnicas evolutivas para optimización con restricciones más conocidas hasta el momento: jerarquización estocástica [16] y Diversity-DE [1].

Finalmente, con el objetivo de mejorar los resultados ya obtenidos, se decidió diseñar, implementar y validar un esquema que hibridice el algoritmo propuesto (Evolución Diferencial para Problemas con Restricciones) con el método de Nelder-Mead. Dicho algoritmo, como se verá más adelante, logra superar en algunas de las funciones de prueba utilizadas al algoritmo de Evolución Diferencial para Problemas con Restricciones.

1.3. Contribuciones

Las principales contribuciones de esta tesis son las siguientes:

1. Se propone un criterio de selección nuevo, aplicable entre dos soluciones correspondientes a un problema de optimización con restricciones.

2. Se propone un algoritmo evolutivo para la resolución de problemas de optimización con restricciones basado en el algoritmo de evolución diferencial [14]. Dicho algoritmo es llamado *Evolución Diferencial para Problemas con Restricciones* (EDPR).

3. Se propone un algoritmo híbrido que combina el algoritmo EDPR y el método propuesto por Nelder y Mead [17] en 1965, para la resolución de problemas de optimización sin restricciones. Dicho algoritmo es llamado *Híbrido de Evolución Diferencial y el método Simplex para Problemas con Restricciones* (HEDSPR).

4. Los dos algoritmos propuestos (EDPR y HEDSPR) son implementados y validados utilizando un conjunto de veintidós funciones de prueba comúnmente adoptadas en la literatura de optimización evolutiva con restricciones. Los resultados obtenidos se comparan con los producidos por dos algoritmos evolutivos del estado del arte: la jerarquización estocástica [16] y Diversity-DE [1].

1.4. Organización de la tesis

El resto de esta tesis está organizada de la siguiente forma:

Capítulo 2. Optimización: Se explican los conceptos básicos de optimización y se describe detalladamente una técnica de programación matemática no lineal llamada método simplex no lineal o de Nelder-Mead. Posteriormente, se da una breve introducción a la computación evolutiva y se describe detalladamente el algoritmo de evolución diferencial.

Capítulo 3. Manejo de restricciones en los AEs: En este capítulo se estudian cuatro técnicas para el manejo de restricciones en los algoritmos evolutivos: Funciones de penalización, jerarquización estocástica, un mecanismo co-evolutivo y un mecanismo basado en diversidad.

Capítulo 4. Algoritmos híbridos: Se estudian diversos algoritmos híbridos para la resolución de problemas de optimización con y sin restricciones. La mayoría de ellos hibridizan un algoritmo evolutivo con el método simplex para resolver problemas sin restricciones. Para problemas con restricciones se estudian dos algoritmos que hibridizan el algoritmo de evolución diferencial en un caso con otro algoritmo evolutivo denominado cúmulo de partículas y en otro con técnicas de programación matemática.

Capítulo 5. Propuesta de dos AEs para resolver problemas de optimización con restricciones: En este capítulo se presenta la propuesta de un criterio de selección entre dos posibles soluciones a un problema de optimización sin restricciones. Dicho criterio se utiliza en el algoritmo de evolución diferencial obteniendo el algoritmo EDPR. Finalmente, se describe detalladamente la propuesta de un algoritmo híbrido del algoritmo EDPR y el método de Nelder-Mead que llamamos HEDSPR.

Capítulo 6. Resultados experimentales: Se muestra un análisis estadístico de los resultados obtenidos por los algoritmos EDPR y HEDSPR. Se comparan ambos algoritmos con dos de los algoritmos que han obtenido mejores resultados en la resolución de problemas de optimización con restricciones: jerarquización estocástica y el algoritmo Diversity-DE.

Capítulo 7. Conclusiones y trabajo futuro: En este último capítulo, se presentan las conclusiones obtenidas con base en los estudios realizados y los resultados obtenidos. Posteriormente, se presenta una posible línea de investigación a seguir en el futuro, como continuación de este trabajo.

Apéndice A. Funciones de prueba: Se presenta el conjunto de funciones de prueba que se utilizaron para la validación de los algoritmos EDPR y HEDSPR.

Apéndice B. Gráficas de convergencia: Se presentan las gráficas de convergencia, tanto de la función objetivo como de la función de violación de las restricciones, correspondientes a la solución mediana, de cada una de las funciones de prueba utilizadas, de los experimentos realizados.

Capítulo 2

Optimización

En diseño, construcción y mantenimiento de sistemas de ingeniería, los ingenieros deben tomar muchas decisiones, tanto tecnológicas como administrativas en las diferentes etapas, teniendo como objetivo principal minimizar el esfuerzo requerido o maximizar el beneficio obtenido. En el sentido más amplio del término, la optimización puede aplicarse para resolver cualquier problema de ingeniería. Sin embargo, no existe ningún método de optimización capaz de resolver todo tipo de problemas por lo que se han desarrollado diferentes métodos desde hace ya varios años [2].

Los métodos de optimización también son llamados técnicas de programación matemática y son estudiados generalmente como una parte de la investigación de operaciones. La investigación de operaciones es una rama de las matemáticas que se ocupa de la aplicación de técnicas y métodos científicos a problemas de toma de decisiones, con el objetivo de establecer la mejor solución posible, es decir, la solución óptima.

Los métodos de optimización pueden ser clasificados en *técnicas de programación matemática*, *técnicas estocásticas* y *métodos estadísticos* [2]. Es importante aclarar que ésta no es una clasificación única. Las técnicas de programación matemática son útiles para encontrar el mínimo de una función de varias variables sujetas a un conjunto de restricciones. Las técnicas estocásticas son usadas para resolver problemas descritos por un conjunto de variables aleatorias, las cuales tienen una distribución de probabilidad conocida. Los métodos estadísticos se utilizan para analizar datos experimentales y construir modelos empíricos que obtengan una representación, lo más precisa posible, de una situación real.

Por otro lado, existen las heurísticas, cuya finalidad es proporcionar aproximaciones a la solución de problemas de alta complejidad en tiempos razonablemente cortos, aunque sin garantizar que dichas soluciones sean óptimas. En esta tesis estudiamos la hibridización de una heurística (la evolución diferencial) con un método de programación matemática (el método de Nelder-Mead).

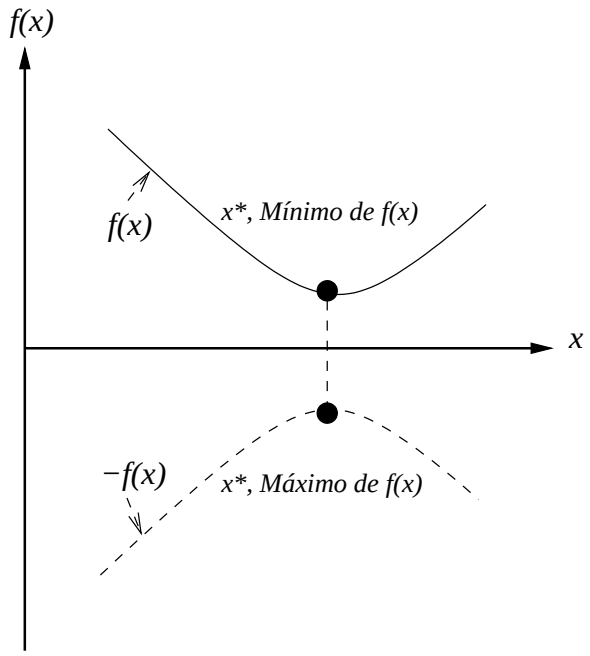


Figura 2.1: El mínimo de la función $f(x)$ es igual al máximo de la función $-f(x)$.

2.1. Definición de un problema de optimización

Un problema de optimización se puede formular como sigue: Encontrar los valores de las n variables $[x_1, x_2, ..., x_n]$, denotadas por el vector $\vec{x}$, que satisfacen m condiciones de desigualdad y p condiciones de igualdad (restricciones) y optimizan (minimizan o maximizan) la función objetivo $f(\vec{x})$. Debido a que el problema de minimizar $f(\vec{x})$ es equivalente al de maximizar $-f(\vec{x})$, ver figura 2.1, el problema de optimización general puede ser escrito como sigue:

$$\begin{aligned} &\text{Encontrar } \vec{x} = [x_1, x_2, ..., x_n]^T \text{ que minimice } f(\vec{x}) \text{ sujeto a:} \\ &g_i(\vec{x}) \leq 0, \qquad i = 1, 2, ..., m \\ &h_j(\vec{x}) = 0, \qquad j = 1, 2, ..., p \end{aligned} \tag{2.1}$$

donde $\vec{x}$ es un vector $n-$dimensional llamado *vector de diseño*, $f(\vec{x})$ es la *función objetivo*, y $g_i(\vec{x}) \leq 0$ y $h_j(\vec{x})$ son las *restricciones de desigualdad e igualdad*, respectivamente. El problema declarado en (2.1) es llamado **problema de optimización con restricciones**. Un **problema de optimización sin restricciones** se define de la siguiente forma:

$$\text{Encontrar } \vec{x} = [x_1, x_2, ..., x_n]^T \text{ que minimice } f(\vec{x}) \tag{2.2}$$

2.1.1. Vector de diseño

Los sistemas de ingeniería son definidos por un conjunto de cantidades, algunas de las cuales son vistas como variables durante el proceso de diseño. En general, ciertas cantidades son fijas y se conocen como **parámetros pre-asignados**. El resto son tratadas como variables y son llamadas **variables de decisión** o **variables de diseño**; x_i, $i = 1, 2, ..., n$. Dichas variables son representadas por un **vector columna de diseño** $\vec{x} = [x_1, x_2, ..., x_n]^T$.

Si no existen restricciones sobre las n variables, entonces el espacio cartesiano $n-$dimensional (donde cada eje coordenado representa una variable de diseño) es llamado **espacio de diseño** o **espacio de búsqueda,** S. En este trabajo de tesis se consideran únicamente problemas donde $\vec{x} \in \mathbb{R}^n$.

2.1.2. Restricciones de diseño

En la mayoría de los problemas, las *variables de diseño* no pueden ser elegidas de manera arbitraria, sino que deben satisfacer ciertos requerimientos. $g_i(\vec{x})$ y $h_j(\vec{x})$, con $i = 1, ..., m$ y $j = 1, ..., p$, son funciones escalares $g_i : \mathbb{R}^n \to \mathbb{R}$ y $h_j : \mathbb{R}^n \to \mathbb{R}$ llamadas restricciones de desigualdad y de igualdad, respectivamente.

Las restricciones que se deben satisfacer 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**.

Región factible y sus límites

El vector de diseño $\vec{x}$ puede ser visto como un punto en el espacio $n-$dimensional. Considerando únicamente restricciones de desigualdad, el subconjunto de todos los puntos que satisfacen la ecuación $g_i(\vec{x}) = 0$ forman una hipersuperficie (subespacio $(n-1)$-dimensional), en el espacio de diseño, llamada **superficie de la restricción**. Al conjunto de todas las superficies de las restricciones se le conoce como **superficie compuesta de las restricciones** [2].

El subconjunto de todos los puntos que satisfacen las restricciones del problema es llamado **región factible** y, por lo tanto, cualquier punto dentro de ésta es llamado **punto factible**. Un punto factible debe caer en la intersección de las superficies correspondientes a todas las restricciones de igualdad y, simultáneamente, sobre los lados adecuados de las superficies correspondientes a las restricciones de desigualdad. De lo contrario, se trata de un **punto infactible**. El subconjunto de todos los puntos infactibles es llamado **región infactible**.

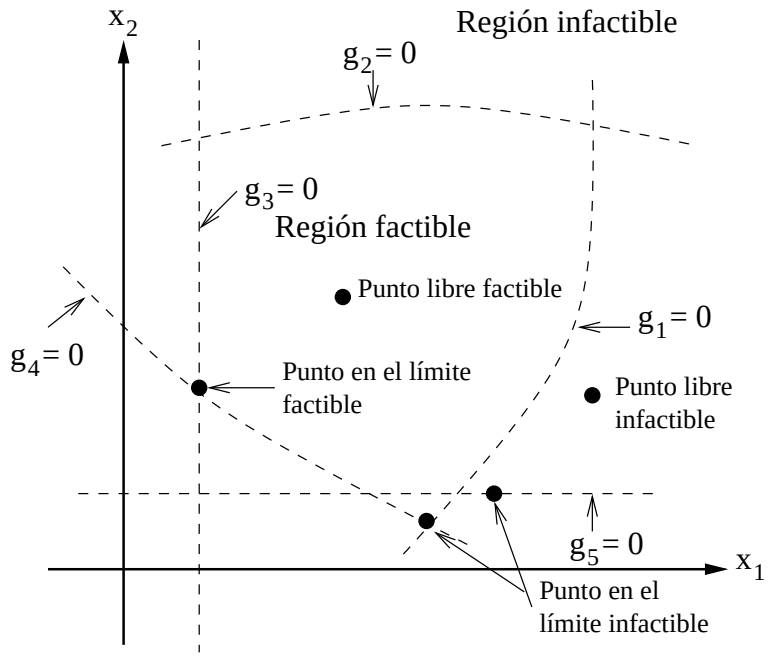


Figura 2.2: Ejemplo hipotético para un problema con dos variables de diseño y únicamente con cinco restricciones de desigualdad.

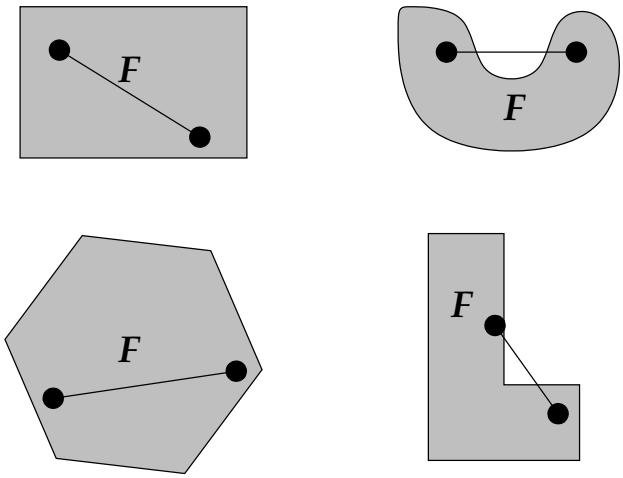
La región factible $F \subseteq S$ se define como sigue:

$$F = \{\vec{x} \in \mathbb{R}^n | \; g_i(\vec{x}) \leq 0, \; i = 1, ..., m \text{ y } h_j(\vec{x}) = 0, \; j = 1, ..., p\} \tag{2.3}$$

Se dice que un *punto está en el límite* si cae al menos sobre una superficie, correspondiente a cualquier restricción; de otra forma es llamado *punto libre*. Por lo anterior, se pueden identificar los siguientes cuatro tipos de puntos:

- punto libre factible
- punto libre infactible
- punto en el límite factible
- punto en el límite infactible

En la figura 2.2 se muestra un ejemplo en el que se ilustran los diferentes tipos de puntos antes mencionados, así como las superficies de las restricciones, la región factible y la región infactible. Las restricciones correspondientes a las superficies sobre las cuales un punto en el límite cae son llamadas **restricciones activas**. Se dice que una restricción, en el punto $\vec{x}$, es:



(a) Conjuntos convexos

(b) Conjuntos no convexos

Figura 2.3: Ejemplos de conjuntos convexos y no convexos.

satisfecha $\iff g_i(\vec{x}) \leq 0$ o $h_j(\vec{x}) = 0$
activa $\iff g_i(\vec{x}) = 0$ o $h_j(\vec{x}) = 0$
inactiva $\iff g_i(\vec{x}) < 0$
violada $\iff g_i(\vec{x}) > 0$ o $h_j(\vec{x}) \neq 0$

Finalmente, es importante definir **convexidad**, pues en capítulos posteriores se utilizará dicho término. El conjunto F es convexo si para toda $\vec{a}, \vec{b} \in F$ y para toda $\theta \in [0, 1]$:

$$(1 - \theta) \cdot \vec{a} + \theta \cdot \vec{b} \in F \tag{2.4}$$

Es decir, F es convexo si para dos puntos cualesquiera $\vec{a}$ y $\vec{b}$ en el conjunto, el segmento rectilíneo que une estos puntos está también dentro del conjunto, ver figura 2.3.

2.1.3. Función objetivo

Los procedimientos de diseño convencional tienen como objetivo encontrar un diseño aceptable o adecuado que satisfaga los requerimientos del problema. Por lo regular, existe más de un diseño aceptable y el propósito de la optimización es elegir el mejor, a partir de un conjunto de alternativas disponibles. El criterio que se utiliza para optimizar es expresado como una función escalar de las variables de diseño, $f : \mathbb{R}^n \rightarrow \mathbb{R}$, y es conocida como **función objetivo**. La elección de la función objetivo es dada por la naturaleza del problema.

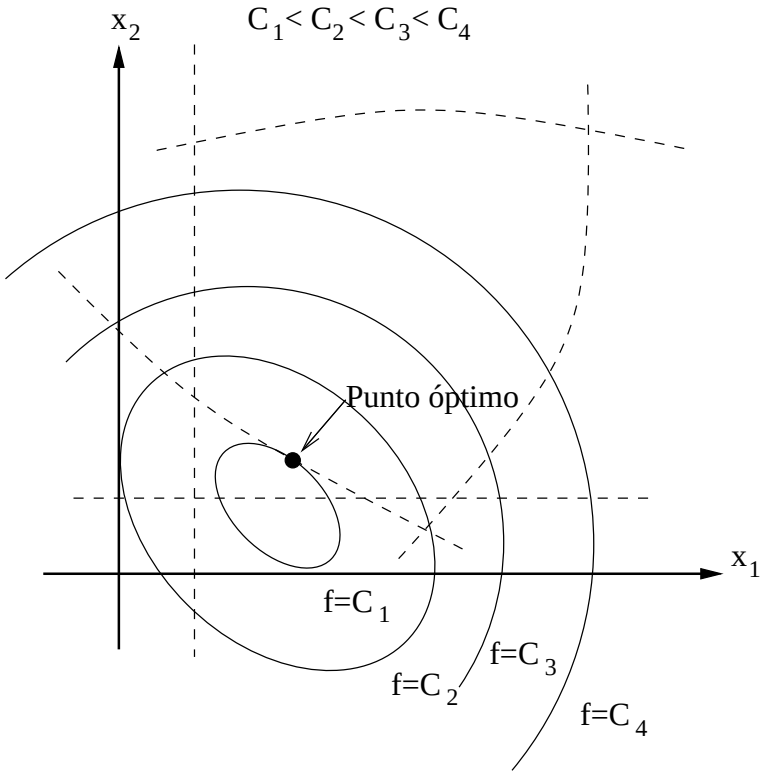


Figura 2.4: Ejemplo hipotético para un problema de dos variables, en el que se muestran los contornos de la función objetivo.

Cuando un problema de optimización envuelve múltiples funciones objetivo es conocido como *problema de programación multiobjetivo*. Cabe mencionar, que en esta tesis sólo se estudiarán problemas que involucren una sola función objetivo.

Superficies de la función objetivo

La posición de todos los puntos que satisfacen $f(x) = C$, donde C es una constante, forman una hipersuperficie en el espacio de diseño, y a cada valor de C le corresponde un miembro diferente de una familia de superficies. Dichas superficies, son llamadas *superficies de la función objetivo*, y se ilustran en la figura 2.4.

Dado que las superficies de la función objetivo pueden ser dibujadas sobre las superficies formadas por las restricciones, el punto óptimo se puede determinar sin mayor dificultad. Sin embargo, el problema surge cuando el número de variables de diseño es mayor a dos o tres, ya que la visualización de las superficies es compleja y, por lo tanto, la visualización ya no resulta posible.

2.1.4. Tipos de solución

La solución al problema de la ec. 2.1 es llamado **mínimo**. Un mínimo $f(\vec{x}^*)$ es llamado **mínimo global** si y sólo si para todo punto factible $\vec{x}$, $f(\vec{x}^*) \leq f(\vec{x})$. Un mínimo $f\left(\vec{x}^L\right)$ es llamado **mínimo local** si y sólo si existe un vecindario factible de $\vec{x}^L$ (definido como $\left\|\vec{x}^L - \vec{x}\right\| \leq \delta$, donde $\vec{x}$ es factible y $\delta > 0$) tal que para todo $\vec{x}$ en el vecindario, $f(\vec{x}_L) \leq f(\vec{x})$ [18].

2.2. Técnicas clásicas de optimización

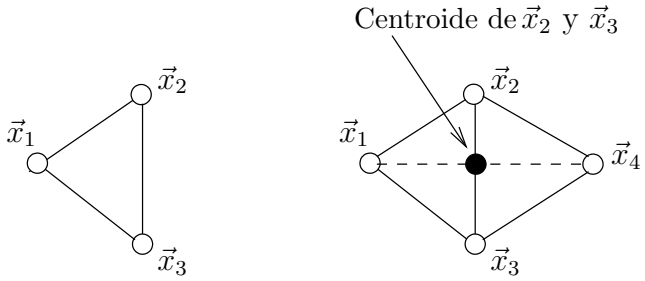
Existen diversas técnicas para solucionar diferentes tipos de problemas de optimización. Los métodos clásicos de cálculo diferencial pueden ser usados para encontrar el máximo o el mínimo de una función de varias variables sin restricciones. Estos métodos presuponen que la función es dos veces diferenciable con respecto a cada variable de decisión y que todas las derivadas son continuas. Para problemas que tienen sólo restricciones de igualdad se puede aplicar el método de los multiplicadores de Lagrange. Si el problema tiene restricciones de desigualdad, las condiciones de Kuhn-Tucker son usadas para localizar el punto óptimo. Sin embargo, es importante hacer énfasis en que estos métodos se pueden aplicar sólo si tanto la función objetivo como las restricciones son diferenciables.

Las técnicas de programación lineal, cuadrática, no lineal, geométrica o entera son usadas para resolver una clase particular de problemas (el nombre indica el tipo de problemas que pueden resolver). Las técnicas de programación no lineal son métodos más generales y son los estudiados en esta tesis.

2.2.1. El método de Nelder y Mead de programación no lineal

En general, los métodos de minimización sin restricciones son iterativos y requieren de un punto inicial para iniciar la búsqueda. Según la estrategia que utilizan para generar el nuevo punto, se pueden clasificar en dos categorías: *métodos de búsqueda directa* y *métodos basados en derivadas*. Los métodos de búsqueda directa requieren únicamente evaluar la función objetivo, sin embargo, no son adecuados para resolver problemas con varias variables. Los métodos basados en derivadas, además de evaluar la función objetivo necesitan la primera y, en algunos casos, la segunda derivada de la función objetivo. Por lo anterior, son más eficientes que las técnicas de búsqueda directa, sin embargo, los métodos basados en derivadas sólo se pueden aplicar a problemas donde la función objetivo sea diferenciable.

La desventaja de estas técnicas es que dependen del punto inicial de búsqueda y, por lo tanto, no garantizan converger al óptimo global. Por ejemplo, en funciones multi-modales pueden converger a óptimos locales.



(a) Simplex viejo: $\vec{x}_1$, $\vec{x}_2$ y $\vec{x}_3$

(b) Simplex nuevo: $\vec{x}_2$, $\vec{x}_3$ y $\vec{x}_4$

Figura 2.5: Creación de un nuevo simplex a partir de otro en dos dimensiones.

El método simplex de búsqueda directa fue propuesto por Spendley, Hext y Himsworth en 1962 [19], y posteriormente mejorado por Nelder y Mead en 1965 [17].

Se le llama **simplex** a la figura geométrica formada por $n+1$ puntos en un espacio n-dimensional. Por ejemplo, en dos dimensiones, el simplex es un triángulo y en tres dimensiones es un tetraedro. Cuando los puntos son equidistantes se dice que el **simplex es regular**.

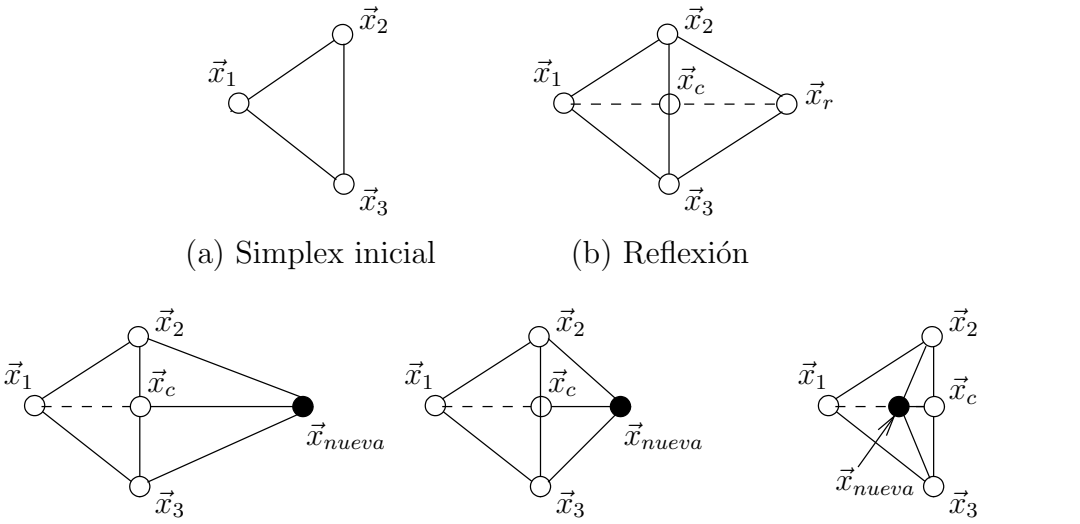
La siguiente ecuación se puede utilizar para generar los vértices de un simplex regular de tamaño α a partir de un punto único$\vec{x}_0$. Si $\alpha = 1$ entonces obtenemos un simplex regular con lados de longitud unitaria.

$$\vec{x}_i = \vec{x}_0 + \delta_1 u_i + \sum_{j=1, j\neq i}^{n} \delta_2 u_j, \qquad i = 1, 2, ..., n \tag{2.5}$$

donde $\delta_1 = \frac{\alpha}{n\sqrt{2}}\left(\sqrt{n+1} + n - 1\right)$, $\delta_1 = \frac{\alpha}{n\sqrt{2}}\left(\sqrt{n+1} - 1\right)$ y u_j es el vector unitario a lo largo del $j-$ésimo eje coordenado.

La idea básica de este método es comparar el valor de la función objetivo de los $n+1$ vértices de un simplex general para, posteriormente, mover el simplex gradualmente, aplicando operadores de *reflexión*, *expansión* o *contracción*, hacia el punto óptimo de manera iterativa. Lo anterior se logra debido a que el algoritmo explota la propiedad de que puede generarse un nuevo simplex sobre cualquier cara del simplex previo, proyectando cualquier vértice elegido en una distancia apropiada a través del centroide de los vértices restantes del viejo simplex, ver figura 2.5. El nuevo simplex se forma entonces reemplazando el viejo vértice por el punto proyectado que se acaba de generar.

Es importante tener en cuenta que el simplex inicial no debe formar un hipercubo ($n-$dimensional) de volumen cero.



(a) Simplex inicial (b) Reflexión

(c) Expansión (d) Contracción externa (e) Contracción interna

Figura 2.6: Ejemplo hipotético del funcionamiento de los operadores utilizados por el algoritmo de Nelder-Mead en dos dimensiones, donde $\vec{x}_1 = \vec{x}_h$.

Reflexión Sea $\vec{x}_h$ el vértice correspondiente al valor más alto de la función objetivo, $f(\vec{x}_h) = \text{máx}_{i=1 \text{ hasta } n+1} f(\vec{x}_i)$, es decir es el peor punto. Entonces podemos obtener el punto $\vec{x}_r$ haciendo una reflexión del punto $\vec{x}_h$ hacia la cara opuesta tratando de obtener un punto con un valor más pequeño en la función objetivo (ver figura 2.6 (b)) de la siguiente manera:

$$\vec{x}_r = (1+\alpha)\,\vec{x}_c - \alpha\vec{x}_h, \tag{2.6}$$

donde $\vec{x}_c$ es el centroide de todos los puntos $\vec{x}_i$, excepto $i = h$:

$$\vec{x}_c = \frac{1}{n}\sum_{i=1, i\neq h}^{n+1} \vec{x}_i, \tag{2.7}$$

y $\alpha > 0$ es el **coeficiente de reflexión** definido como:

$$\alpha = \frac{\text{distancia entre } \vec{x}_r \text{ y } \vec{x}_c}{\text{distancia entre } \vec{x}_h \text{ y } \vec{x}_c}. \tag{2.8}$$

Expansión Si en el proceso de reflexión se obtiene un punto tal que $f(\vec{x}_r) < f(\vec{x}_l)$; donde $\vec{x}_l$ es el mejor punto, $f(\vec{x}_l) = \text{mín}_{i=1 \text{ hasta } n+1} f(\vec{x}_i)$, entonces se espera que el valor de la función objetivo decrezca si nos movemos a lo largo de la dirección que

va de $\vec{x}_c$ a $\vec{x}_r$. Por lo tanto, se realiza una expansión de $\vec{x}_r$ (ver figura 2.6 (c)) de la siguiente forma:

$$\vec{x}_{nueva} = (1+\gamma)\,\vec{x}_c - \gamma\vec{x}_h, \tag{2.9}$$

donde $\gamma > 1$ es llamado **coeficiente de expansión**, definido como:

$$\gamma = \frac{\text{distancia entre } \vec{x}_{nueva} \text{ y } \vec{x}_c}{\text{distancia entre } \vec{x}_h \text{ y } \vec{x}_c}. \tag{2.10}$$

Contracción Si en el proceso de reflexión se genera un punto $\vec{x}_r$ tal que $f(\vec{x}_r) \geq f(\overrightarrow{x}_h)$ entonces se considera que la reflexión condujo al simplex a una mala región del espacio de búsqueda y por lo tanto se realiza una *contracción interna* en la dirección que va de $\vec{x}_c$ a $\vec{x}_r$ (ver figura 2.6 (e)) de la manera siguiente:

$$\vec{x}_{nueva} = (1-\beta)\,\vec{x}_c + \beta\vec{x}_h, \tag{2.11}$$

donde β es llamado **coeficiente de contracción** $(0 \leq \beta \leq 1)$ y se define como:

$$\beta = \frac{\text{distancia entre } \vec{x}_{nueva} \text{ y } \vec{x}_c}{\text{distancia entre } \vec{x}_h \text{ y } \vec{x}_c}. \tag{2.12}$$

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 2.6 (d), de la siguiente forma:

$$\vec{x}_{nueva} = (1+\beta)\,\vec{x}_c - \beta\vec{x}_h. \tag{2.13}$$

El escenario por omisión es el punto reflejado, $\vec{x}_{nueva} = \vec{x}_r$. Finalmente, el nuevo punto reemplaza al peor punto en el simplex, $\vec{x}_h = \vec{x}_{nueva}$, y el algoritmo continúa con el nuevo simplex. El método presupone que ha convergido cuando la desviación estándar del valor de la función objetivo para los $n+1$ vértices del simplex actual es tan pequeña como se describe a continuación:

$$Q = \left[\sum_{i=1}^{n+1} \frac{(f(\vec{x}_i) - f(\vec{x}_c))^2}{n+1}\right]^{\frac{1}{2}} \leq \varepsilon. \tag{2.14}$$

En el algoritmo 1 se muestra el funcionamiento completo de este método. Nelder y Mead reportan buenos resultados al utilizar los siguientes valores: $\alpha = 1$, $\beta = \frac{1}{2}$ y $\gamma = 2$.

Algoritmo 1: Método de Nelder-Mead

entrada: $\gamma > 1$ (factor de expansión), $\beta \in (0, 1)$ (factor de contracción) y una tolerancia ϵ

salida : Mejor solución encontrada

repeat

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 $f(x_r) \geq f(x_h)$ **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: $x_{nueva} \leftarrow (1 + \beta)x_c - x_h$;

end

end

end

Calcular $f(x_{nueva})$;

$x_h \leftarrow x_{nueva}$;

Calcular $Q \leftarrow \left[\sum_{i=1}^{n+1} \frac{(f(x_i) - f(x_c))^2}{n+1} \right]^{\frac{1}{2}}$;

until *Cumplir criterio de terminación:* $Q < \epsilon$;

2.3. Computación evolutiva

A lo largo de la historia varios científicos y pensadores contribuyeron al entendimiento del proceso evolutivo. El paradigma Neo-Darwiniano es la combinación de la teoría evolutiva propuesta por Charles Darwin, el seleccionismo de August Weismann y la genética de Gregor Mendel. Este paradigma establece que la vida en nuestro planeta puede ser explicada a través de un conjunto de procesos estocásticos que actúan en las poblaciones y especies. Dichos procesos son: *reproducción*, *mutación*, *competencia* y *selección*.

Desde los años 1930s, la evolución natural fue vista como un proceso de aprendizaje. Por ejemplo, W. D. Cannon, plantea en su libro *The Wisdom of the Body* [20] que el proceso evolutivo es parecido al aprendizaje por ensayo y error que se manifiesta en los humanos. Alan Turing reconoció una conexión obvia entre la evolución y el aprendizaje de máquina en su artículo titulado "Computing Machinery and Intelligence" [21].

Por lo anterior, no es sorprendente que exista una área dentro de las ciencias de la computación llamada **computación evolutiva**, la cual engloba una serie de técnicas inspiradas en los principios de la teoría Neo-Darwiniana de la evolución natural.

2.3.1. Algoritmos evolutivos

Actualmente, existe una cantidad considerable de AEs los cuales son aplicados para resolver una gran variedad de problemas. Sin embargo, la mayoría de ellos tienen características en común. Los principales componentes de los AEs son:

1. Definición de las características (o rasgos) de los individuos, que es el problema de cómo representarlos
2. Función de aptitud (función objetivo o alguna variante de ésta)
3. Mecanismo de selección de padres
4. Operadores de recombinación y mutación

Una vez que se cuenta con el modelo matemático del problema a resolverse se realiza lo siguiente: Se genera una **población inicial**, la cual es un conjunto de posibles soluciones al problema y cada solución tiene asignada una **aptitud**. La aptitud es calculada evaluando la función objetivo (o alguna variante de ésta) en dicha solución. Con base en los valores de aptitud, se **seleccionan** algunos individuos de la población para que funjan como padres de los nuevos individuos. Los nuevos individuos se generan aplicando operadores de **recombinación** y **mutación** a los padres ya

seleccionados. Finalmente, tanto las soluciones actuales (población actual) como las soluciones generadas **compiten** por un lugar en la nueva población.

El proceso antes descrito, es repetido hasta cumplir con un **criterio de terminación**, que puede ser, por ejemplo:

- la población ha evolucionado un determinado número de generaciones,
- en la población existe poca diversidad, o
- en la población actual ya se encuentra una solución adecuada para el problema.

Algoritmo 2: Algoritmo evolutivo

entrada: Tamaño de la población (N) y parámetros involucrados en el criterio de terminación
salida : Población final
$g \leftarrow 0$;
Generar la población inicial: $P_g = \{\vec{x_1}, \vec{x_2}, ..., , \vec{x_N}\}$;
Obtener la aptitud de cada individuo: $f(\vec{x_1}), f(\vec{x_2}), ..., f(\vec{x_N})$;
while *NO se cumpla criterio de terminación* **do**
  Seleccionar padres: $padres \subset P_g$;
  Aplicar operador de recombinación: $P'_g \leftarrow recombinar(padres)$;
  Aplicar operador de mutación: $P''_g \leftarrow mutar(P'_g)$;
  Obtener la aptitud de cada individuo en P''_g;
  Obtener la nueva población: $P_{g+1} \leftarrow seleccionar(P''_g \cup P_g)$;
  $g \leftarrow g + 1$;
end

Es importante notar que los elementos que componen a un AE son *estocásticos*. Al aplicar el operador de selección los individuos más aptos tienen una probabilidad mayor de ser elegidos para formar parte del conjunto de padres que generarán nuevos individuos; conforme disminuye el valor de aptitud la probabilidad de que ese individuo sea seleccionado disminuye. El operador de recombinación elige aleatoriamente los segmentos de información que son transferidos a los descendientes. Finalmente, el operador de mutación altera aleatoriamente la información del individuo.

El algoritmo 2 muestra un esquema general de un AE, donde g nos indica la generación actual, *padres* es un subconjunto de la población actual generado al elegir individuos de la población actual con base en su aptitud, P_g representa a la población actual, P'_g es el conjunto de soluciones generadas al aplicar el operador de recombinación a los padres seleccionados previamente, P''_g es el conjunto de soluciones generadas al aplicar el operador de mutación a las soluciones previamente encontradas (P'_g).

2.3.2. Paradigmas de la computación evolutiva

Por razones sobre todo históricas, se consideran como los principales paradigmas de la computación evolutiva a tres técnicas:

- Programación evolutiva
- Estrategias evolutivas
- Algoritmos genéticos

Sin embargo, actualmente se han propuesto otras heurísticas bio-inspiradas, las cuales no necesariamente se limitan a simular la evolución de especies. Ejemplos de éstas son:

- Programación genética
- Optimización con colonia de hormigas
- Sistema inmune artificial
- Algoritmos meméticos
- Optimización con cúmulos de partículas
- Evolución diferencial

Para fines de esta tesis únicamente se estudiará a profundidad la denominada evolución diferencial.

2.3.3. Evolución diferencial

El algoritmo de evolución diferencial (ED) fue propuesto por Rainer Storn y Kenneth Price a mediados de los 1990s y surge a partir de los intentos de Kenneth Price por ajustar polinomios de Chebyshev. La idea principal es utilizar la diferencia entre vectores para perturbar a otro vector de la población. ED es una heurística para minimizar funciones no lineales y no diferenciables sobre espacios continuos. Este método ha demostrado converger más rápido y con más certeza que muchos otros métodos de optimización global, requiere de pocas variables de control, es robusto, fácil de implementar y de paralelizar [14].

Las variables de decisión son codificadas como variables reales y mutadas con operaciones aritméticas simples. Durante la mutación se aplica un operador de cruza el cual va a decidir qué variables serán perturbadas.

Población

El algoritmo de ED mantiene dos poblaciones, ambas con N_p individuos. Cada individuo es representado por un vector que tiene N variables, donde N es el número de variables que tiene el problema. Cada variable almacena un valor real. El algoritmo se ejecutará durante $g_{\text{máx}}$ evoluciones. La población actual es representada por P_x y se define de la siguiente forma:

$$\begin{array}{ll} P_{\mathbf{x},g} = (\vec{x}_{i,g}), & i = 0, 1, ..., N_p - 1, \qquad g = 0, 1, ..., g_{\text{máx}} \\ \vec{x}_{i,g=} (x_{j,i,g}), & j = 0, 1, ..., N, \end{array} \tag{2.15}$$

donde g indica el número de generación, i el índice que tiene el individuo en la población y j el índice de la variable del individuo i.

ED realiza mutaciones a individuos de la población elegidos aleatoriamente para generar nuevos individuos y así crear una población intermedia, la cual está definida de la siguiente forma:

$$\begin{array}{ll} P_{\mathbf{v},g} = (\vec{v}_{i,g}), & i = 0, 1, ..., N_p - 1, \qquad g = 0, 1, ..., g_{\text{máx}} \\ \vec{v}_{i,g=} (v_{j,i,g}), & j = 0, 1, ..., N. \end{array} \tag{2.16}$$

Posteriormente, cada individuo de la población actual, $P_{\mathbf{x},g}$, es combinado con uno de la población $P_{\mathbf{v},g}$ y se produce una nueva población de prueba denotada como sigue:

$$\begin{array}{ll} P_{\mathbf{u},g} = (\vec{u}_{i,g}), & i = 0, 1, ..., N_p - 1, \qquad g = 0, 1, ..., g_{\text{máx}} \\ \vec{u}_{i,g=} (u_{j,i,g}), & j = 0, 1, ..., N. \end{array} \tag{2.17}$$

Población inicial

Para poder generar la población inicial es necesario conocer los límites inferiores y superiores de cada variable del problema. El vector $\vec{b}_L$ contiene el límite inferior de cada variable y vector $\vec{b}_U$ el límite superior de cada variable. De esta forma, se puede asignar un valor a cada variable de los vectores que representan a los individuos, respetando dichos límites. Lo anterior lo podemos hacer de la siguiente forma:

$$\mathbf{x}_{j,i,0} = \mathbf{rand}\,(0, 1) \cdot (b_{j,U} - b_{j,L}) + b_{j,L}, \tag{2.18}$$

donde $\mathbf{rand}\,(0, 1)$ es una función de distribución uniforme que regresa un número aleatorio en el rango de $[0, 1]$. Se debe generar un número aleatorio para cada variable

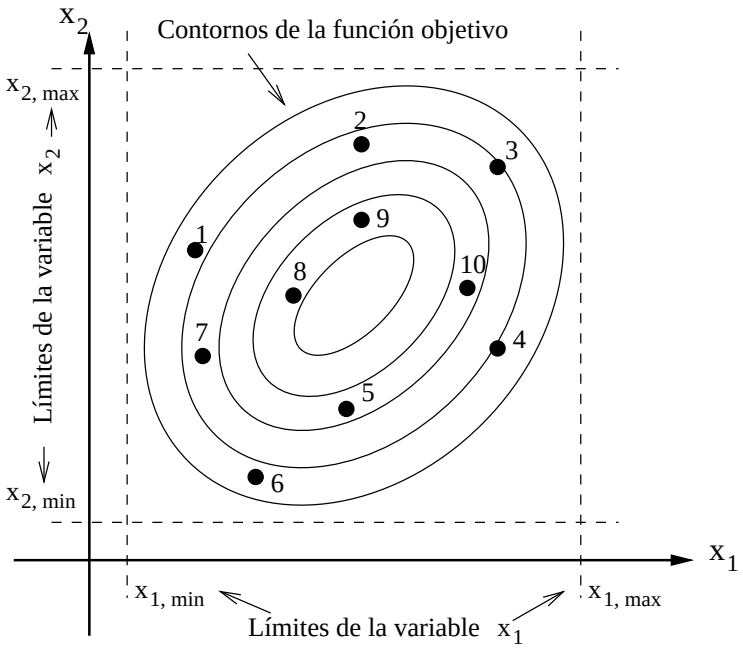


Figura 2.7: Población inicial.

que se desea inicializar. Es importante mencionar que incluso cuando el problema maneja variables discretas es necesario inicializar los vectores con valores reales, ya que el algoritmo internamente sólo trabaja con variables reales.

En la figura 2.7 se muestra un ejemplo hipotético para un problema con dos variables de decisión. Es importante considerar que la población inicial debe cubrir uniformemente el espacio de búsqueda.

Mutación

Una vez inicializada la población, ED muta y recombina la población para producir una población de N_p vectores de prueba. La **mutación diferencial** a partir de tres vectores elegidos aleatoriamente y un factor de escalamiento genera un nuevo vector, de la siguiente forma:

$$\vec{v}_{i,g} = \vec{x}_{r0,g} + F \cdot (\vec{x}_{r1,g} - \vec{x}_{r2,g}) \,. \tag{2.19}$$

El factor $F \in (0, 1+)$ es un factor que controla la amplitud en la que la población se desenvuelve. Si bien no se tiene un límite superior, los valores son rara vez mayores que 1.0.

El índice del vector base, $r0$, puede ser determinado de diferentes formas. Por ejemplo, se pueden elegir tres números aleatorios diferentes de i para $r0$, $r1$ y $r2$ por

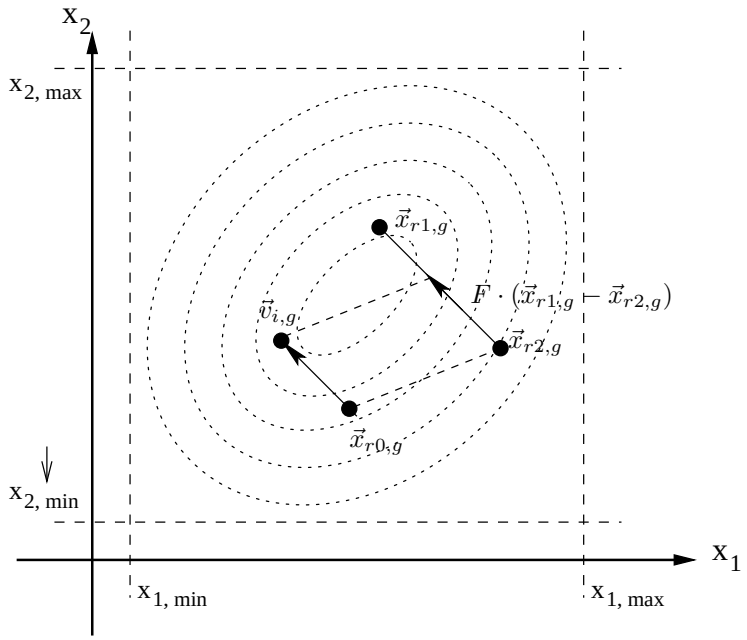


Figura 2.8: **Mutación diferencial**. El vector diferencial ponderado, $F \cdot (\vec{x}_{r1,g} - \vec{x}_{r2,g})$, es sumado al vector base, $\vec{x}_{r0,g}$, para producir el vector mutado, $\vec{v}_{i,g}$.

cada mutación que se realice. En la figura 2.8 se muestra un ejemplo hipotético del proceso de mutación en dos dimensiones.

Cruza

Para completar la estrategia de búsqueda, ED también aplica una **cruza uniforme**. Algunas veces se le denomina recombinación discreta. En particular, ED cruza cada vector de la población actual con un vector mutado, de la siguiente forma:

$$\vec{u}_{i,g} = u_{j,i,g} = \begin{cases} v_{j,i,g} \text{ si } (\mathbf{rand}\,(0,1) \leq Cr \ o \ j = j_{\mathbf{rand}}) \\ x_{j,i,g} \text{ en otro caso} \end{cases} \tag{2.20}$$

La probabilidad de cruza, $Cr \in [0,1]$, es definida por el usuario y controla la fracción de variables que serán copiadas del individuo mutado y las variables que se copiarán del individuo actual. En la figura 2.9 se muestra un ejemplo hipotético, en dos dimensiones, aplicando una cruza uniforme.

Selección

Para decidir qué individuos estarán en la población de la siguiente generación se utiliza el siguiente criterio:

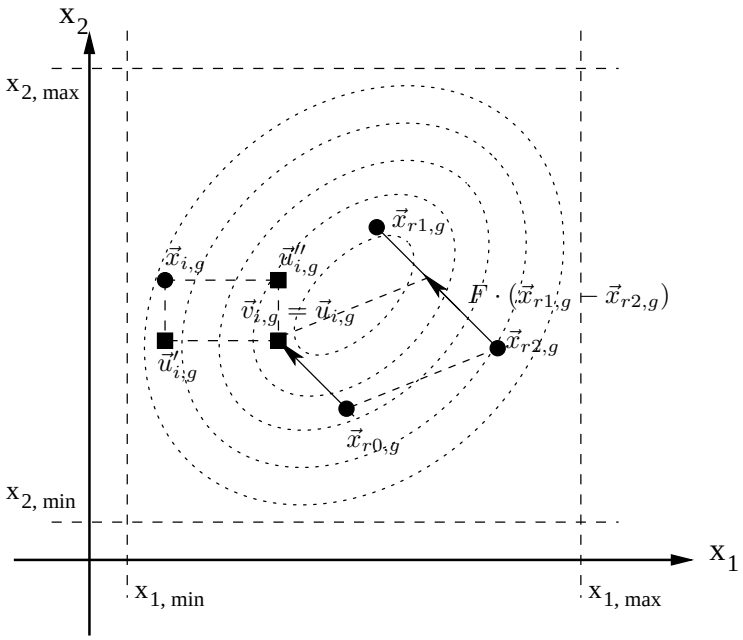


Figura 2.9: Al realizar una cruza uniforme, los posibles vectores de prueba son: $\vec{u}'_{i,g}$, $\vec{u}''_{i,g}$, $\vec{x}'_{i,g}$ y $\vec{v}'_{i,g}$.

$$\vec{x}_{i,g+1} = \begin{cases} \vec{u}_{i,g} \text{ si } f(\vec{u}_{i,g}) \leq f(\vec{x}_{i,g}) \\ \vec{x}_{i,g} \text{ en otro caso} \end{cases} \tag{2.21}$$

Una vez que se ha construido la nueva población, el proceso de mutación, recombinación y selección es repetido hasta que el óptimo sea localizado, o se cumpla un criterio de terminación.

El algoritmo 3 muestra el procedimiento completo de la técnica de ED.

Variantes de ED

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:

Algoritmo 3: Evolución Diferencial

entrada: D (número de variables que tiene el problema), N_p (tamaño de la población), Cr (probabilidad de cruza y g_{max} (generaciones máximas)
salida : Población final
Generar la población inicial;
repeat
  `/*Se genera la población de prueba */`
  for $i \leftarrow 1$ **to** N_p **do**
    repeat
      $r0 \leftarrow floor(rand(0,1) * N_p)$;
    until $r0 \neq i$;
    repeat
      $r1 \leftarrow floor(rand(0,1) * N_p)$;
    until $r1 \neq r0$ AND $r1 \neq i$;
    repeat
      $r2 \leftarrow floor(rand(0,1) * N_p)$;
    until $r2 \neq r1$ AND $r2 \neq r0$ AND $r2 \neq i$;
    $j_{rand} \leftarrow floor(D * rand(0,1))$;
    `/*Se crea al nuevo individuo, aplicando los operadores de mutación y cruza */`
    for $j \leftarrow 1$ **to** D **do**
      if $rand(0,1) \leq Cr$ OR $j = j_{rand}$ **then**
        $u_{j,i} \leftarrow x_{j,r0} + F * (x_{j,r1} - x_{j,r2})$;
      else
        $u_{j,i} \leftarrow x_{j,i}$;
      end
    end
    `/*Se aplica el operador de selección */`
    if $f(\mathbf{U}_i) \leq f(\mathbf{X}_i)$ **then**
      $\mathbf{X}_i \leftarrow \mathbf{U}_i$;
    end
  end
until *No se exceda* g_{max} ;

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.

De acuerdo al estudio comparativo realizado en [22] se observa que los modelos *DE/rand/1/bin* y *DE/best/1/bin* muestran los mejores rendimientos en términos de resultados y de número de evaluaciones de la función objetivo.

Capítulo 3

Manejo de restricciones en los AEs

Muchos de los problemas de optimización que existen en el mundo real están sujetos a un conjunto de restricciones. Las técnicas de programación matemática existentes para la resolución de los mismos cuentan con varias limitaciones, por ejemplo cuando los problemas cuentan con algunas de las siguientes características: la función objetivo y/o las restricciones son no diferenciables o discontinuas, cuando la región factible es disjunta o cuando la función objetivo y/o las restricciones no pueden ser expresadas de manera algebraica. De hecho, las condiciones de Kuhn-Tucker para optimilidad no son válidas para el problema general de optimización no lineal. Estas condiciones son una generalización del método de los multiplicadores de Lagrange y se aplican cuando se tienen restricciones de desigualdad.

Los Algoritmos Evolutivos (AEs) han podido resolver gran variedad de problemas que cuentan con las características antes mencionadas. Sin embargo, los AEs no cuentan con un mecanismo explícito para manejo de restricciones. Debido a ello, 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.

3.1. Funciones de penalización

Las funciones de penalización es uno de los mecanismos que se utiliza más comúnmente para manejar restricciones en los AEs. La idea de esta técnica es transformar un problema con restriciones en uno sin restricciones al agregar a la función objetivo los factores de penalización tanto para restricciones de igualdad como para las de desigualdad y, de esta forma, si las restricciones son violadas, entonces el valor de la función objetivo se ve afectado por un costo ("la penalización").

En programación matemática, se consideran dos tipos de funciones de penalización [3]:

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 los límites. Lo anterior con la finalidad de guiar la búsqueda únicamente al 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.

Las funciones de penalización son de la siguiente forma:

$$\phi(\vec{x}) = f(\vec{x}) \pm \left[\sum_{i=1}^{m} r_i G_i + \sum_{i=1}^{p} c_j L_j \right] \tag{3.1}$$

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 es:

$$G_i = \text{máx}\left[0, g_i(\vec{x})\right]^{\beta}, \tag{3.2}$$

$$L_j = \left|h_j(\vec{x})\right|^{\gamma}, \tag{3.3}$$

donde, β y γ son uno o dos, normalmente. Además, una restricción de igualdad se puede transformar en una de desigualdad de la siguiente forma:

$$\left|h_j(\vec{x})\right| - \epsilon \leq 0 \tag{3.4}$$

donde ϵ establece la tolerancia permitida.

El uso de esta técnica tiene una seria desventaja: el resultado depende de los factores de penalización y éstos son difíciles de determinar. Se debe cuidar que los factores de penalización estén equilibrados con respecto a los valores de la función objetivo y además se debe determinar qué tanto se quiere penalizar a una solución infactible. Si se eligen valores muy pequeños, el algoritmo sólo considerará el valor de la función objetivo, pues no penalizará suficientemente a las soluciones infactibles, y por lo tanto puede no llegar a la zona factible. Si se eligen valores muy grandes, el algoritmo llegará a la zona factible, pero no podrá oscilar entre la zona factible y la infactible lo cual impedirá llegar al óptimo en los casos en que éste se encuentre en la frontera entre estas dos zonas.

Richardson et al. [23] definieron algunas reglas para diseñar una función de penalización. Sin embargo, éstas no están libres de problemas. Por ejemplo, mencionan que 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, pero no queda claro qué tanto se deben explorar las regiones infactibles.

Se han propuesto diversas variantes de la función de penalización. A continuación se describen las más populares en la literatura especializada.

3.1.1. Pena de muerte

Consiste en desechar (volviéndolo a generar) o asignar una aptitud de cero a un individuo infactible y tomar el valor de la función objetivo para los individuos factibles. Esta variante no es recomendable ya que habrá un estancamiento en la búsqueda cuando en la población inicial no exista ningún individuo factible. Michalewicz [24] 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.

3.1.2. Penalizaciones estáticas

En esta variante los factores de penalización no cambian durante el proceso evolutivo, lo cual no es recomendable ya que conforme el proceso evolutivo avanza varía la porción explorada de la región infactible y es posible que ya no interese explorar dicha región en las etapas finales de la búsqueda. Además, los factores son generalmente dependientes del problema y en algunos casos se debe definir un valor alto para los coeficientes de penalización. Un ejemplo de esta técnica se presenta en [25].

3.1.3. Penalizaciones dinámicas

En este caso los factores de penalización se ajustan conforme avanza el proceso evolutivo. Normalmente, los factores de penalización crecen conforme trascurren

las generaciones. Se cree que este tipo de esquema es mejor que las penalizaciones estáticas. Sin embargo, los problemas asociados a las penalizaciones estáticas también aplican a las penalizaciones dinámicas ya que es igualmente díficil derivar buenas funciones de penalización y producir buenos factores de penalización en ambos casos (estáticos y dinámicos). Un ejemplo de esta variante se presenta en [26].

3.1.4. Penalizaciones adaptativas

Usa funciones de penalización que se retroalimentan del proceso evolutivo por lo que regula los factores de una manera más inteligente tratando de evitar que todos los puntos sean factibles o todos sean infactibles. Un ejemplo de esta técnica se presenta en [27].

3.2. 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 [28]. 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. Un aspecto interesante de esta técnica es que no requiere la definición de factores de penalización.

Este mecanismo jerarquiza λ 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 díficil, se define una probabilidad P_f 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_f. Los autores determinaron que se logran los mejores resultados, en varios problemas, cuando $0.4 < P_f < 0.5$. 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

entrada: Población de individuos que se desea jerarquizar
salida : Población de individuos jerarquizados
for $i \leftarrow 1$ **to** N **do**
  for $j \leftarrow 1$ **to** $\lambda - 1$ **do**
    if $\psi(\vec{x}_j) = \psi(\vec{x}_{j+1}) = 0$ OR $U(0,1) < P_f$ **then**
      if $f(\vec{x}_j) > f(\vec{x}_{j+1})$ **then**
        Intercambiar$(\vec{x}_j, \vec{x}_{j+1})$;
      end
    else
      if $\psi(\vec{x}_j) > \psi(\vec{x}_{j+1})$ **then**
        Intercambiar$(\vec{x}_j, \vec{x}_{j+1})$;
      end
    end
  end
  if No se realizo intercambio **then**
    Terminar();
  end
end

Los autores presentan una versión mejorada de su algoritmo en [16]. 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 del tipo evolución diferencial, en caso de que las variables generadas después del proceso de mutación estén fuera del rango permitido.

Los autores reportan resultados realizando únicamente 350,000 evaluaciones de la función objetivo. La validación se realiza con trece funciones de prueba que corresponden a las primeras trece funciones del apéndice A.

3.3. Mecanismo co-evolutivo (Algoritmo CODE)

En [29] se presenta un algoritmo llamado *A Co-evolutionary differential evolution algorithm (CODE)*. CODE trabaja con dos poblaciones las cuales evolucionan de manera independiente. La primera población evoluciona para encontrar la mejor solución sin tomar en cuenta las restricciones. La segunda población evoluciona buscando minimizar la violación a las restricciones sin importar el valor de la función objetivo. Ambas poblaciones evolucionan de acuerdo al algoritmo de la evolución diferencial (ED). Los autores justifican el uso de ED con las siguientes razones:

- Utiliza un operador diferencial simple, para crear posibles soluciones, el cual trabaja con números reales de manera natural.

- ED tiene memoria, ya que el conocimiento de buenas soluciones es retenido en la población actual, mientras que en el Algoritmo Genético (AG), el conocimiento previo del problema es destruido cuando cambia la población, debido a que dicha técnica suele emplear un reemplazo total de la población.

- ED es computacionalmente menos costoso, en términos de memoria y tiempo de CPU, que otras metaheurísticas.

- La precisión de la búsqueda local es mejor en ED que en los AGs y que en la optimización mediante cúmulo de partículas (PSO) [30].

- De acuerdo a comparaciones que se han hecho con otros algoritmos, como Annealed Nelder and Mead strategy (ANM), Adaptive simulated annealing (ASA), Evolution strategies (ESs) y Standard genetic algorithm (SGA), se ha observado que ED es más eficiente y efectivo [30].

En CODE, después de cierto número de generaciones se combinan ambas poblaciones y se dividen nuevamente en dos grupos. Debido a que una solución infactible puede estar mucho más cerca del valor óptimo que algunas soluciones factibles, los autores de este método definen un umbral, δ, para determinar el límite entre ambas poblaciones. Una solución se introduce en el primer grupo si la suma de las violaciones de todas las restricciones es menor o igual que δ; de lo contrario se introduce en el segundo grupo. Para encontrar el óptimo global que satisface las restricciones, δ debe ir decreciendo durante el proceso evolutivo.

Los autores realizaron pruebas con cinco funciones de prueba propuestas por Michalewicz [31, 32] y concluyeron que CODE es superior a otros algoritmos basados en funciones de penalización. También muestran tablas comparando los resultados con dos algoritmos: COGA (Algoritmo Genético Co-evolutivo) [33, 34] y GA-PF (Algoritmo Genético con funciones de penalización) en las que se observa que CODE obtiene mejores resultados en cuanto a precisión y costo computacional. Sin embargo, es importante mencionar que el algoritmo denominado jerarquización estocástica obtiene mejores resultados que CODE.

3.4. Mecanismo de diversidad

En [1] se introduce un mecanismo de diversidad al algoritmo ED para resolver problemas de optimización con restricciones. Este mecanismo tiene dos objetivos principalmente:

1. Permitir soluciones infactibles, con un valor promisorio de la función objetivo, en la población, e

2. Incrementar la probabilidad de que un individuo genere un individuo mejor a él mientras se promueve la colaboración de toda la población para generar mejores individuos

El mecanismo propuesto trabaja de la siguiente forma: Cada solución que se encuentra en la población genera n_0 nuevas soluciones, usando los operadores de mutación y cruza del algoritmo ED. Entre las n_0 soluciones se elige la mejor de acuerdo al criterio propuesto por Deb [35]: Si las dos soluciones son factibles, se elige la que tenga el mejor valor en la función objetivo. Si se tiene una solución factible y una solución infactible se elige la solución factible. Si ambas soluciones son infactibles se elige la que tenga una menor violación en las restricciones. Posteriormente, se compara la solución actual con la mejor solución obtenida usando un parámetro llamado *tasa de selección*, S_r , el cual determina si se compara con base en el valor de la función objetivo o con base en la violación de las restricciones. El procedimiento completo se muestra en el algoritmo 5, donde N es el número de variables que tiene el problema y N_p es el tamaño de la población.

Al realizar experimentos con trece funciones de prueba (las primeras trece del apéndice A) se puede observar que el algoritmo Diversity-DE, produce resultados competitivos, basándose en calidad y robustez, con respecto a una de las mejores técnicas: jerarquización estocástica [28]. De acuerdo a los resultados obtenidos, se puede pensar que el hecho de permitir soluciones infactibles que tienen un buen valor en la función objetivo proporciona mejores resultados cuando se utiliza como motor de búsqueda ED en lugar de una estrategia evolutiva. También se puede observar que Diversity-DE es superior a otros algoritmos basados en ED.

Es importante mencionar que Diversity-DE trabaja adecuadamente en diversos tipos de problemas: altamente restringidos, con una dimensionalidad alta o baja, con restricciones de diferentes tipos (lineales, no lineales, de desigualdad, de igualdad) que generan zonas factibles muy pequeñas e incluso disjuntas. Además de que tiene un costo computacional menor, con respecto al número de evaluaciones de la función objetivo, que el método de jerarquización estocástica.

Algoritmo 5: Procedimiento de Diversity-DE

entrada: Cr (probabilidad de cruza), S_r (tasa de selección), n_0 (número de soluciones que genera cada individuo) y g_{max} (generaciones máximas)
salida : Población final
$G \leftarrow 0$;
Crear la población inicial $\vec{x}_G^i \ \forall i,\ i = 1, ..., N_p$;
Evaluar $f(\vec{x}_G^i) \ \forall i,\ i = 1, ..., N_p$;
for $G \leftarrow 1$ **to** g_{max} **do**
  $F \leftarrow rand[0.3, 0.9]$;
  for $i \leftarrow 1$ **to** N_p **do**
    for $k \leftarrow 1$ **to** n_0 **do**
      Seleccionar aleatoriamente $r1 \neq r2 \neq r3 \neq i$;
      $j_{rand} \leftarrow$ **randint**$(1, D)$;
      for $j \leftarrow 1$ **to** N **do**
        if $rand_j[0, 1) < CR \ OR \ j = j_{rand}$ **then**
          $child_j \leftarrow x_{j,G}^{r3} + F(x_{j,G}^{r1} - x_{j,G}^{r2})$;
        else
          $child_j \leftarrow x_{j,G}^{i}$;
        end
      end
      if $k > 1$ **then**
        if *child es mejor que* $\vec{u}_{G+1}^i$ *(basado en los tres criterios de selección)* **then**
          $\vec{u}_{G+1}^i \leftarrow child$;
        end
      else
        $\vec{u}_{G+1}^i \leftarrow child$;
      end
    end
    if $flip(S_r)$ **then**
      if $f(\vec{u}_{G+1}^i) \leq f(\vec{x}_G^i)$ **then**
        $\vec{x}_{G+1}^i \leftarrow \vec{u}_{G+1}^i$;
      else
        $\vec{x}_{G+1}^i \leftarrow \vec{x}_G^i$;
      end
    else
      if $\vec{u}_{G+1}^i$ *es mejor que* $\vec{x}_G^i$ *(basado en los tres criterios de selección)* **then**
        $\vec{x}_{G+1}^i \leftarrow \vec{u}_{G+1}^i$;
      else
        $\vec{x}_{G+1}^i \leftarrow \vec{x}_G^i$;
      end
    end
  end
  $G \leftarrow G + 1$;
end

Capítulo 4

Algoritmos híbridos

En los últimos años, en un intento por mejorar la eficiencia de las técnicas existentes, para resolver problemas de optimización, se han propuesto varios algoritmos híbridos. Se entiende por algoritmo híbrido a un algoritmo que es producto de algoritmos de distinta naturaleza. Estos algoritmos han combinado algunas técnicas de optimización como: algoritmos genéticos (AGs) y el método de Nelder-Mead [36], cúmulo de partículas (PSO) y el método de Nelder-Mead [37], evolución diferencial (ED) y el método de Nelder-Mead [38], cúmulo de partículas y evolución diferencial [39], evolución diferencial y el método de actualización de multiplicadores [40], algoritmo cultural y DPSO (híbrido de evolución diferencial y cúmulo de partículas) [41], por mencionar algunos.

4.1. Optimización sin restricciones

4.1.1. Algoritmo genético hibridizado con el método de Nelder-Mead

En [36] se propone un algoritmo llamado "Continuous Hybrid Algorithm (CHA)" para optimización de funciones continuas multimodales. Este algoritmo es un híbrido de un algoritmo genético llamado "Continuous Genetic Algorithm (CGA)" y la modificación del método simplex propuesta por Nelder y Mead [17]. CGA utiliza codificación real y trabaja de la siguiente forma:

- Se genera la población inicial cubriendo homogéneamente el espacio de búsqueda. Para esto, definen un vecindario para cada individuo y no permiten que haya más de un individuo por vecindario.
- La aptitud de cada individuo, $Fit\,(i)$, depende tanto del valor de la función objetivo, para ese individuo, como del valor máximo de la función objetivo en la población.

- El cálculo de la probabilidad de que un individuo sea seleccionado, p_i, va a depender únicamente de su aptitud y a partir de ésta se calcula la probabilidad acumulada de cada individuo a través de las generaciones, $p_{c_i} = p_{c_{i-1}} + p_i$ con $p_{c_o} = 0$.
- El *operador de selección* trabaja como una ruleta y el cálculo de la porción de ruleta que corresponde a cada individuo se hace con base en la probabilidad acumulada.
- El *operador de recombinación* selecciona dos individuos y un número aleatorio (entre 0 y el número de variables), j, y se van a modificar únicamente las variables del individuo que están almacenadas después del punto j.
- El *operador de mutación* selecciona una variable del individuo aleatoriamente y la modifica.
- Finalmente, utiliza un *operador de reinserción*, como estrategia elitista, que en caso de que el mejor individuo de la población de hijos tenga una aptitud menor al mejor individuo de la población de padres, entonces sustituye al peor individuo de la población de hijos por el mejor individuo de la población de padres.

Se ha probado que la modificación del método simplex propuesta por Nelder y Mead [17] ha sido útil en muchas aplicaciones. Ésta sólo necesita hacer evaluaciones de la función objetivo sin requerir el gradiente. Así mismo, una vez generado el simplex inicial realiza a lo más dos evaluaciones de la función objetivo en cada iteración, y exhibe convergencia superlineal sin requerir el cálculo de la matriz Hessiana o el gradiente.

CHA tiene dos etapas principales: *Diversificación e intensificación*. La **etapa de diversificación** está basada en CGA y su objetivo es explorar el espacio de búsqueda y encontrar una región promisoria para que posteriormente se inicie la **etapa de intensificación**, basada en [17], cuyo objetivo es explotar la región promisoria para encontrar el óptimo con la mayor precisión posible. De acuerdo a los resultados que reportan sus autores, CHA tiene resultados similares o mejores que otros métodos u otras versiones de algoritmos genéticos en funciones que tienen menos de 10 variables, pero con un menor costo computacional.

4.1.2. Híbrido de cúmulo de partículas y el método de Nelder-Mead

En [37] se presenta un algoritmo híbrido para localizar óptimos globales orientado a funciones multimodales continuas llamado HSMPSO. Este algoritmo utiliza la modificación del método simplex propuesta por Nelder y Mead [17] como un operador en

PSO. La *heurística PSO* trabaja de la siguiente manera: Una población de soluciones potenciales es usada para hacer la búsqueda en el espacio de soluciones como todas las técnicas evolutivas. Sin embargo, la población no es manipulada de acuerdo a operadores inspirados en el proceso evolutivo sino que la dinámica de la población simula el movimiento de una bandada de pájaros en busca de alimento. Durante el proceso de búsqueda, los individuos intercambian información y se pueden ver beneficiados de los descubrimientos y experiencias anteriores de sus demás compañeros.

HSMPSO funciona como sigue: Después de un número fijo de iteraciones de PSO se aplica un operador basado en el método de Nelder-Mead a cada partícula. En caso de que una partícula llegue a la solución buscada con cierta precisión dada, se termina la búsqueda; de lo contrario, se vuelve a ejecutar PSO en la población durante un determinado número de iteraciones. Por lo tanto, HSMPSO utiliza PSO para explorar el espacio de búsqueda y el método de Nelder-Mead para explotar cierta región de éste.

De acuerdo a los experimentos realizados, los autores de este algoritmo híbrido reportan que éste presenta buenos resultados para funciones multimodales continuas, comparado con otros algoritmos, excepto en presencia de alta dimensionalidad. Por lo tanto, proponen como trabajo futuro mejorar la convergencia en problemas con alta dimensionalidad y extender el algoritmo a optimización multiobjetivo con restricciones.

4.1.3. Híbrido de evolución diferencial y el método de Nelder-Mead

En [38] se diseña un algoritmo evolutivo, con codificación real, para optimización global. Dicho algoritmo utiliza la modificación del método simplex propuesta por Nelder y Mead en [17] y lo combinan con la heurística de evolución diferencial. La heurística ED, al igual que PSO, trabaja con una población de posibles soluciones. Sin embargo, un nuevo individuo es generado a partir de la combinación lineal de dos individuos elegidos aleatoriamente de la población, con la finalidad de estimar el gradiente en dicha zona. En el artículo mencionan dos posibles diseños del algoritmo híbrido. Sin embargo, uno de ellos no obtuvo buenos resultados ya que la velocidad de convergencia era demasiado lenta, especialmente en problemas con dimensión alta. Por este motivo, se propone una segunda alternativa, en la que se introduce una técnica para disminuir la dimensionalidad, a la que llamaron "low dimensional simplex evolution".

Una vez que se reduce la dimensión, el algoritmo propuesto por Nelder y Mead no se ve como un optimizador ya que no converge ni a un mínimo local ni a un mínimo global. Para solucionar este problema mantienen un conjunto con más puntos durante toda la búsqueda. De acuerdo a experimentos que realizaron definen un problema de alta dimensionalidad a partir de $n \geq 5$. El esquema del híbrido es el siguiente: Se

utiliza el método de Nelder-Mead con dimensiones pequeñas para producir individuos y éstos sobreviven de acuerdo a una regla de selección natural.

El algoritmo "low dimensional simplex evolution" maneja una población de N puntos, $\mathbf{X}(t)$, la cual va evolucionando. La evolución se hace reemplazando los puntos malos de la población actual por puntos mejores en cada generación. Esto se logra de la siguiente manera: Para cada punto de la población, $X_i(t)$ donde $i = 1, ..., N$, se seleccionan $m+1$ puntos aleatoriamente para formar el $m-$simplex. Luego, el punto $X_i(t)$ se intenta mejorar usando los operadores de reflexión y contracción. En caso de no ser mejorado, un individuo se reemplaza tratando de mejorar la diversidad. En el algoritmo 6 se muestra el procedimiento completo.

Los resultados que reportan es que el algoritmo "low dimensional simplex evolution" mejora a la evolución diferencial la mayoría de las veces con respecto al promedio del número de evaluaciones de la función objetivo y además existen casos en que el porcentaje de éxito también es superado.

4.1.4. Híbrido de un algoritmo cultural, evolución diferencial y cúmulo de partículas

En [41] se propone un algoritmo cultural basado en DPSO como una técnica de optimización global. Los algoritmos culturales tienen dos componentes principales: Una población y un espacio de creencias. La población es el conjunto de individuos que son posibles soluciones al problema y el espacio de creencias es un repositorio de información donde los individuos pueden almacenar sus experiencias para que otros individuos aprendan de ellas. La población y el espacio de creencias están relacionados a través de un protocolo de comunicación, el cual establece las reglas sobre los individuos que pueden contribuir al espacio de creencias y la forma en que el espacio de creencias puede contribuir a la generación de nuevos individuos.

DPSO es un algoritmo híbrido que incorpora un mecanismo de variación diferencial al algoritmo PSO, con la finalidad de mantener diversidad en la población. Lo anterior debido a que PSO tiene la desventaja de perder diversidad si la partícula líder cae en un óptimo local, lo que ocasiona que todas las demás partículas converjan a dicho óptimo local.

Las fuentes de conocimiento que están en el espacio de creencias del algoritmo son diseñadas de acuerdo a la evolución de la población en DPSO. Se tienen cuatro fuentes de conocimiento: Situacional, normativo, topográfico e histórico. El *conocimiento situacional* consiste del mejor ejemplar encontrado a lo largo del proceso evolutivo. El *conocimiento normativo* contiene los intervalos de las variables de velocidad donde se han encontrado buenas posiciones. El *conocimiento topográfico* es un mapa del paisaje de aptitudes del problema durante el proceso evolutivo. Este mapa consiste de un conjunto de celdas; cada celda almacena la aptidud de la mejor partícula encontrada

Algoritmo 6: Procedimiento "low dimensional simplex evolution"

entrada: N_p (tamaño de la población), m (tamaño del simplex), factores de expansión y contracción (α, β)
salida : Población final
$G \leftarrow 0$;
Crear la población inicial $\vec{x}_G^i \ \forall i,\ i = 1, ..., N_p$;
Evaluar $f(\vec{x}_G^i) \ \forall i,\ i = 1, ..., N_p$;
repeat
  for $i \leftarrow 1$ **to** N_p **do**
    Elegir aleatoriamente $m+1$ individuos de la población actual: $\vec{s}_G^j \ \forall j$, $j = 1, ..., m+1$;
    Determinar el mejor individuo del subconjunto elegido: $\vec{x}_G^b$;
    Determinar el peor individuo del subconjunto elegido: $\vec{x}_G^w$;
    Calcular el centroide: $\vec{x}_G^c \leftarrow \frac{1}{m} \sum_{j=1, j\neq w}^{m+1} \vec{s}_G^j$;
    Realizar la reflexión: $\vec{x}_G^r \leftarrow \vec{x}_G^c + \alpha \cdot (\vec{x}_G^c - \vec{x}_G^w)$;
    if $f(\vec{x}_G^r) < f(\vec{x}_G^i)$ **then**
      $\vec{x}_{G+1}^i \leftarrow \vec{x}_G^r$;
    else
      Se realiza una contracción: $\vec{x}_G^C \leftarrow \vec{x}_G^c + \beta \cdot (\vec{x}_G^w - \vec{x}_G^c)$;
      if $f(\vec{x}_G^C) < f(\vec{x}_G^i)$ **then**
        $\vec{x}_{G+1}^i \leftarrow \vec{x}_G^C$;
      else
        if $f(\vec{x}_G^i) \geq \frac{1}{N_p} \sum_{k=1}^{N_p} f(\vec{x}_G^k)$ **then**
          if $f(\vec{x}_G^b) < f(\vec{x}_G^i)$ **then**
            $\vec{x}_{G+1}^i \leftarrow \vec{x}_G^i + 0.618 \cdot (\vec{x}_G^b - \vec{x}_G^i)$;
          else
            $\vec{x}_{G+1}^i \leftarrow \vec{x}_G^i + 0.382 \cdot (\vec{x}_G^i - \vec{x}_G^w)$;
          end
        end
      end
    end
  end
  $G \leftarrow G + 1$;
until *Cumplir criterio de terminación* ;

en dicha celda. Finalmente, el *conocimiento histórico* contiene información acerca de las secuencias de cambios ambientales en términos de la posición y la velocidad del óptimo en el espacio de búsqueda. Lo que se almacena en este último conocimiento es el promedio de los cambios en posición y velocidad.

Las diferentes fuentes de conocimiento se utilizan para influir en la variación del operador DPSO, buscando mejorar la capacidad de búsqueda y el costo computacional del algoritmo. De acuerdo a los experimentos realizados se observa que el algoritmo propuesto obtiene buenos resultados y además conserva las características de robustez y convergencia rápida para problemas de optimización real. Sin embargo, los autores mencionan que es necesario hacer un análisis del impacto de cada fuente de conocimiento durante el proceso evolutivo y además realizar un estudio más profundo de otras técnicas evolutivas de optimización para intentar mejorar el algoritmo ya propuesto.

4.2. Optimización con restricciones

4.2.1. DEPSO: Híbrido de cúmulo de partículas con un operador de evolución diferencial

En [39] se propone un híbrido de PSO con un operador de evolución diferencial. Este operador realiza mutaciones en forma de campana siguiendo una distribución Gaussiana. A este híbrido lo llaman DEPSO y trabaja alternando el operador original de PSO y el operador de evolución diferencial. El operador de evolución diferencial se aplica en generaciones impares y el operador PSO en generaciones pares.

Los autores justifican que debido a que tanto PSO como ED utilizan pocos puntos no es posible emplear directamente algunas técnicas para el manejo de restriciones como lo es la jerarquización estocástica, ya que éstas utilizan poblaciones grandes. Por este motivo, DEPSO hace el manejo de restricciones aplicando el criterio propuesto por Deb [35]: Si las dos soluciones son factibles, se elige la que tenga el mejor valor en la función objetivo. Si se tiene un solución factible y una solución infactible se elige la solución factible. Si ambas soluciones son infactibles se elige la que tenga una menor violación en las restricciones.

Al realizar experimentos con diferentes funciones de prueba reportan que DEPSO obtiene muy buenos resultados en problemas con variables enteras y lo atribuyen al manejo de las mutaciones en forma de campana. Sin embargo, su desempeño no es muy bueno para problemas en los que la zona factible es muy pequeña, por ejemplo aquellos que tienen restricciones de igualdad. Por lo anterior, proponen como investigación a futuro emplear una memoria extendida con un conjunto de puntos a los que se les puedan aplicar otras técnicas para el manejo de restricciones tales como la jerarquización estocástica.

4.2.2. Híbrido de evolución diferencial con un método de actualización de multiplicadores

En [40] se diseña un híbrido con evolución diferencial y el método de actualización de multiplicadores para resolver problemas de optimización con restricciones. Además presentan un esquema adaptativo para los parámetros de penalización, que incluyen en el híbrido, con la finalidad de que se puedan usar parámetros de penalización pequeños sin afectar los resultados de la búsqueda. Esto último es muy interesante debido a que el funcionamiento del algoritmo no depende mucho de los parámetros de entrada.

Al algoritmo propuesto lo llaman "hybrid differential evolution (HDE)" y lo presentan como una extensión del algoritmo ED. El operador de mutación que se usa en HDE es muy similar a ED en cuanto a que usan la diferencia entre dos individuos elegidos aleatoriamente como dirección de búsqueda. La diferencia está en que ED utiliza un factor fijo que multiplica al vector resultante de la diferencia para obtener al nuevo individuo y HDE usa un factor aleatorio con la finalidad de obtener una mayor diversidad en la población. El operador de selección, al igual que en ED, es una competencia entre padres e hijos.

HDE cuenta con dos operadores más, uno de aceleración y otro de migración. El **operador de aceleración** se usa para aumentar la velocidad de convergencia. Al diseñar este operador tomaron en cuenta que cuando se usa ED para resolver problemas de optimización, la aptitud no se mejora en cada generación sino que es hasta después de cierto número de generaciones que se puede observar una mejora. Por lo anterior, proponen aplicar el método del descenso empinado para mejorar al individuo con mayor aptitud y conducirlo hacia una mejor solución. Sin embargo, este operador puede repercutir en la diversidad y hacer que se presente convergencia prematura. Por este motivo, incorporan el **operador de migración** el cual aplica un algoritmo de clusters a la población y posteriormente regenera los individuos con base en el mejor individuo de la población actual. El operador de migración sólo es aplicado si al medir la diversidad en la población se observa que no satisface la tolerancia permitida.

Con respecto al manejo de restricciones mencionan que los métodos de penalización son los más comunes. Sin embargo, Powell [42] observó que los métodos de penalización tienen una seria desventaja, ya que si los factores para la violación de restricciones no están equilibrados entonces esto dificulta más el poder llegar a buenas soluciones. El método de los multiplicadores de Lagrange, también para el manejo de restricciones, mejora significativamente a los métodos de penalización. Sin embargo, puede llegar a una solución infactible si los factores de penalización son pequeños. Por lo anterior, decidieron utilizar el método de convergencia global de Powell [42] junto con un método de actualización de los multiplicadores con la finalidad de forzar a que se converja a una solución factible. Al algoritmo ya con el manejo de restricciones lo llaman HDE-MUM-APP.

Los resultados que obtienen con HDE-MUM-APP los comparan con otros híbridos propuestos y se observa que HDE-MUM-APP obtiene mejores resultados la mayoría de las veces. Sin embargo, se puede observar que para algunas funciones, HDE-MUM-APP diverge.

Capítulo 5

Propuesta de un AE para resolver problemas de optimización con restricciones

Sabemos que el algoritmo de ED ha demostrado converger más rápido y con más certeza que muchos otros métodos de optimización global. Además, requiere de pocas variables de control, es robusto, fácil de implementar y paralelizable [14]. Sin embargo, ED en su forma natural no cuenta con un mecanismo para el manejo de restricciones y aunque ya se han hecho propuestas, algunas de las cuales se describen en el capítulo 3, es necesario buscar métodos que obtengan mejores resultados. Esto, en virtud de que, como ya se ha mencionado, existe una enorme cantidad de problemas de optimización con restricciones en el mundo real y por lo tanto una mejora a la ED en este sentido extiende de manera muy importante su aplicabilidad.

En este capítulo se propone un mecanismo para el manejo de restricciones y se incorpora a la heurística ED con la finalidad de mejorar los resultados obtenidos por los algoritmos actuales. El algoritmo obtenido es llamado *"Evolución Diferencial para Problemas Restringidos (EDPR)"*. Finalmente, se propone añadir un operador basado en el método de Nelder-Mead al algoritmo EDPR con la finalidad de acelerar la convergencia a buenas soluciones.

5.1. Técnica propuesta para el manejo de restricciones en ED

Como se mencionó en la sección 2.3.3, existen diversas variantes del algoritmo ED. Sin embargo, en [22] se realiza un estudio comparativo de dichas variantes concluyendo que el modelo **DE/rand/1/bin** muestra muy buenos rendimientos en términos de resultados y de número de evaluaciones. Por ello, se trabajará con esta variante de ED en esta tesis. El procedimiento completo de dicha variante se muestra en el algoritmo 3 de la página 25.

Debido a que ED es una heurística para minimizar funciones sin restricciones, el operador de selección que utiliza se basa únicamente en el valor de la función objetivo, usando el siguiente criterio:

$$\vec{x}_{i,g+1} = \begin{cases} \vec{u}_{i,g} & \text{si } f(\vec{u}_{i,g}) \leq f(\vec{x}_{i,g}) \\ \vec{x}_{i,g} & \text{en otro caso} \end{cases} \tag{5.1}$$

Como ya se estudió en el capítulo 3, el problema principal de las técnicas para el manejo de restricciones es lograr el balance correcto entre tener prioridad en minimizar el valor de la función objetivo o tener prioridad en minimizar la violación a las restricciones. La técnica de jerarquización estocástica [28] y el mecanismo de diversidad propuesto en [1] han obtenido excelentes resultados, por lo que, se decidió analizarlos a fondo para proponer un cambio que pudiera producir mejoras en los resultados.

En [28], al jerarquizar λ posibles soluciones, se propone el uso de una probabilidad P_f que va a determinar el balance entre tener prioridad con respecto al valor de la función objetivo o tener prioridad con respecto a la violación de las restricciones. Al comparar pares de soluciones, la probabilidad de comparar con respecto al valor de la función objetivo es 1 si ambas soluciones son factibles; en otro caso, es P_f.

En [1] se promueve la colaboración de toda la población para generar mejores individuos. Lo anterior se logra haciendo que cada solución que se encuentra en la población genere n_0 nuevas soluciones. Entre las n_0 soluciones se elige la mejor de acuerdo al criterio propuesto por Deb [35]:

- Si las dos soluciones son factibles, se elige la que tenga el mejor valor en la función objetivo.
- Si se tiene un solución factible y una solución infactible se elige la solución factible.
- Si ambas soluciones son infactibles se elige la que tenga una menor violación en las restricciones.

Finalmente, también se incorpora un elemento probabilístico llamado *tasa de selección*, S_r, similar al de la jerarquización estocástica y se compara la solución encontrada (la mejor de las n_0 soluciones) con la solución actual para determinar cuál de ellas permanecerá en la siguiente generación.

Podemos notar que tanto en la técnica de jerarquización estocástica como en el mecanismo de diversidad no se considera el caso en el que las soluciones a comparar

tengan el mismo valor en la violación a las restricciones. En ese caso, la comparación se realizaría basándonose únicamente en el valor de la función objetivo. Nuestra propuesta para el manejo de las restricciones en ED se presenta a continuación.

Debido a la naturaleza del algoritmo ED, únicamente se modifica el operador de selección utilizando el siguiente criterio:

- Si ambas soluciones tienen el mismo valor en la violación a las restricciones se elige la que tenga un mejor valor en la función objetivo.
- De lo contrario, de acuerdo a una probabilidad definida se decide si la comparación se hace de acuerdo a la violación a las restricciones o al valor de la función objetivo.

En el algoritmo 7 se muestra la modificación de ED siguiendo nuestra propuesta. La función objetivo es denotada por f y la función de violación a las restricciones es denotada por ψ.

5.2. Operador de mutación basado en el método de Nelder-Mead

La idea de este operador surge a partir de que sabemos que los AEs son capaces de lidiar con problemas de optimización complejos que las técnicas clásicas no pueden resolver. Sin embargo, los AEs tienen un costo computacional mucho mayor que las técnicas clásicas. Por tal motivo, se pretende usar una técnica de programación matemática para acelerar la convergencia a soluciones de buena calidad.

En la sección 2.2.1, se mencionaron los *métodos de búsqueda directa* y los *métodos basados en derivadas*. Los métodos basados en derivadas son más eficientes que los métodos de búsqueda directa; sin embargo, requieren de la primera y, en algunos casos, de la segunda derivada de la función objetivo. Por ello, sólo se pueden aplicar a problemas donde la función objetivo sea diferenciable. Por lo anterior, se decidió diseñar un operador basado en un método de búsqueda directa.

En un inicio se pensó en utilizar el método complex propuesto por Box en [43]. Este método está basado en el método de Nelder-Mead pero incluye un esquema sencillo de manejo de restricciones. Box propone que para cada punto generado se verifique si es factible. En caso de ser infactible, dicho punto se contrae hacia el centro de los puntos generados previamente hasta que éste se vuelva factible. El método complex no requiere de continuidad pero tiene la desventaja de requerir que la región factible sea convexa, debido a la forma en que se hace el manejo de las restricciones, ver figura 5.1. Además, es necesario que el conjunto de puntos inicial contenga únicamente puntos

Algoritmo 7: Evolución Diferencial para Problemas Restringidos (EDPR)

entrada: D (número de variables que tiene el problema), N_p (tamaño de la población), Cr (probabilidad de cruza), P_f (probabilidad de comparar con respecto a f) y g_{max} (generaciones máximas)
salida : Población final
Generar la población inicial;
repeat
  `/*Se genera la población de prueba */`
  for $i \leftarrow 1$ **to** N_p **do**
    repeat
      $r0 \leftarrow floor(rand(0,1) * N_p)$;
    until $r0 \neq i$;
    repeat
      $r1 \leftarrow floor(rand(0,1) * N_p)$;
    until $r1 \neq r0$ AND $r1 \neq i$;
    repeat
      $r2 \leftarrow floor(rand(0,1) * N_p)$;
    until $r2 \neq r1$ AND $r2 \neq r0$ AND $r2 \neq i$;
    $j_{rand} \leftarrow floor(D * rand(0,1))$;
    `/*Se crea al nuevo individuo, aplicando los operadores de mutación y cruza */`
    for $j \leftarrow 1$ **to** D **do**
      if $rand(0,1) = Cr$ OR $j = j_{rand}$ **then**
        $u_{j,i} \leftarrow x_{j,r0} + F * (x_{j,r1} - x_{j,r2})$;
      else
        $u_{j,i} \leftarrow x_{j,i}$;
      end
    end
    `/*Se aplica el operador de selección propuesto */`
    if $\psi(\vec{u}_i) = \psi(\vec{x}_i)$ AND $f(\vec{u}_i) \leq f(\vec{x}_i)$ **then**
      $\vec{x}_i \leftarrow \vec{u}_i$;
    else
      if $rand(0,1) < P_f$ **then**
        if $f(\vec{u}_i) < f(\vec{x}_i)$ **then**
          $\vec{x}_i \leftarrow \vec{u}_i$;
        end
      else
        if $\psi(\vec{u}_i) < \psi(\vec{x}_i)$ **then**
          $\vec{x}_i \leftarrow \vec{u}_i$;
        end
      end
    end
  end
until *No se exceda* g_{max} ;

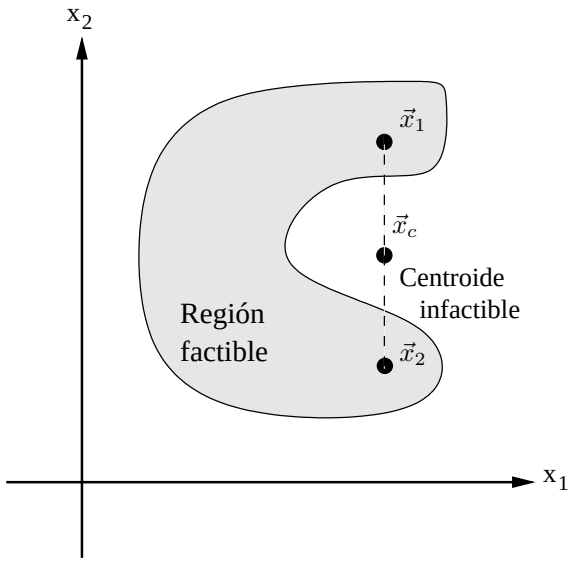


Figura 5.1: Ejemplo hipotético de un caso en el que el centroide es infactible. En este caso, no se logrará factibilidad al contraer cualquier punto hacia él.

factibles, lo cual resulta sumamente díficil en la práctica, cuando la zona factible no es convexa.

Varios autores han propuesto cambios al algoritmo básico de Box como por ejemplo, reglas más elaboradas para la contracción y expansión, uso de pesos para los centroides, modificaciones para acomodar los centroides que son infactibles en la generación de los puntos iniciales, usar una interpolación cuadrática en vez de un muestreo aleatorio. Sin embargo, estas propuestas no han mejorado de manera significativa al algoritmo original.

Debido a las desventajas antes mencionadas del método complex se decidió utilizar la modificación del método simplex propuesta por Nelder y Mead [17] e incorporar una técnica para el manejo de restricciones. Se eligió este método debido a que diversos estudios comparativos indican que es muy confiable y eficiente. Además, una vez generado el simplex inicial, sólo realiza una o dos evaluaciones de la función objetivo (en caso de aplicar los operadores de expansión o contracción) en cada iteración. En 1969, Box y Draper llegaron a afirmar que este algoritmo es el "más eficiente de todas las técnicas secuenciales" [44].

El problema que existe al utilizar el método simplex es que no es recomendable para problemas de varias variables debido a la dificultad de poder generar buenas direcciones de búsqueda. Existen varias propuestas de AEs hibridizados con el método simplex pero en los resultados reportados se observan malos resultados en problemas de alta dimensionalidad. En el capítulo 4 se describieron algunas de estas propuestas.

Entre los algoritmos estudiados está el propuesto en [38] llamado "Low Dimensional Simplex Evolution (LDSE)" el cual combina la heurística ED con la modificación

del método simplex propuesta por Nelder y Mead en [17] obteniendo muy buenos resultados. En éste, los autores proponen introducir una técnica para disminuir la dimensión en el simplex. Una vez que se reduce la dimensión, el algoritmo propuesto por Nelder y Mead no se ve como un optimizador ya que no converge ni a un mínimo local ni a un mínimo global. Para solucionar este problema mantienen un conjunto con más puntos durante toda la búsqueda. De acuerdo a experimentos que realizaron definen un problema de alta dimensionalidad a partir de $n \geq 5$.

LDSE maneja una población de N puntos, $\mathbf{X}(t)$, la cual va evolucionando. La evolución se hace reemplazando los puntos malos de la población actual por puntos mejores en cada generación. Esto se logra de la siguiente manera: Para cada punto de la población, $X_i(t)$ donde $i = 1, ..., N$, se seleccionan $m+1$ puntos aleatoriamente para formar el $m-$simplex. Luego, el punto $X_i(t)$ se intenta mejorar usando los operadores de reflexión y contracción. En caso de no ser mejorado, se reemplaza tratando de mejorar la diversidad. El procedimiento completo se muestra en el algoritmo 6 en la página 39. Los autores reportan mejoras a la ED la mayoría de las veces con respecto al promedio del número de evaluaciones de la función objetivo y además reportan casos en los que el porcentaje de éxito también es superado.

Basándonos en el algoritmo LDSE proponemos añadir un operador al algoritmo EDPR, al cual llamaremos **operador simplex**. Dicho operador, trabaja de manera muy similar a LDSE, con la diferencia de que vamos a incluir un esquema para el manejo de restricciones. Dicho esquema es muy similar al utilizado en el algoritmo EDPR, con la diferencia de que no va a incluir un elemento probabilístico. En este caso, se va a dar prioridad a las soluciones factibles, y en caso de tener dos soluciones con el mismo valor de violación a las restricciones, se elige con base en el valor de la función objetivo. El esquema antes descrito, se utiliza tanto en las comparaciones requeridas por el método de Nelder-Mead como en las comparaciones requeridas para el ordenamiento del conjunto de posibles soluciones. El procedimiento completo del operador simplex se muestra en el algoritmo 8.

Finalmente, se propone aplicar el operador simplex con cierta frecuencia en EDPR a partir de cierta generación, en lugar de los operadores de mutación y cruza originales. De esta forma, obtenemos un algoritmo híbrido del algoritmo evolutivo EDPR y el método de Nelder-Mead. El procedimiento completo se muestra en el algoritmo 9.

Con respecto al criterio de terminación, se presupone que el AE ha convergido en la generación G, siempre y cuando, tanto la desviación estándar de la función objetivo como la de las soluciones, sean menores que un valor suficientemente pequeño, ε y δ respetivamente, como sigue:

$$\sum_{i=1}^{N_p} \left(f(\vec{x}_i) - \bar{f}\right)^2 \leq \varepsilon \qquad \text{y} \qquad \sum_{i=1}^{N_p} \left\|\vec{x}_i - \vec{\bar{x}}\right\|^2 \leq \delta \tag{5.2}$$

Algoritmo 8: Operador simplex

entrada: Dimensión del simplex: m, población actual: X_i^G donde $i = 1, 2, ..., N_p$, punto al que se va aplicar el operador: $\vec{x}_i$
salida : Nuevo punto

Elegir aleatoriamente m puntos de la población actual y junto con $\vec{x}_i$ formar el $m-$simplex;
Encontrar $\vec{x}_h$ (el peor punto) y $\vec{x}_b$ (el mejor punto);
Calcular el centroide: $\vec{x}_c \leftarrow \frac{1}{m} \sum_{i=1, i \neq h}^{m+1} \vec{x}_i$;
Calcular el punto reflejado: $\vec{x}_r \leftarrow \vec{x}_c + \gamma \cdot \left(\vec{x}_c - \vec{x}_h\right)$;
`/*Utilizando el criterio propuesto en la sección 5.2 */`
if $\vec{x}_r$ *es mejor a* $\vec{x}_i$ **then**
  $\vec{x}_i \leftarrow \vec{x}_r$;
else
  Realizar una contracción: $\vec{x}_{nueva} \leftarrow \vec{x}_c + \beta \cdot \left(\vec{x}_h - \vec{x}_c\right)$;
  `/*Utilizando el criterio propuesto en la sección 5.2 */`
  if $\vec{x}_{nueva}$ *es mejor que* $\vec{x}_i$ **then**
    $\vec{x}_i \leftarrow \vec{x}_{nueva}$;
  else
    if $\vec{x}_b$ *es mejor que* $\vec{x}_i$ **then**
      $\vec{x}_{nueva} \leftarrow \vec{x}_i + 0.618 \cdot \left(\vec{x}_b - \vec{x}_i\right)$;
    else
      $\vec{x}_{nueva} \leftarrow \vec{x}_i + 0.382 \cdot \left(\vec{x}_i - \vec{x}_h\right)$;
    end
    `/*Utilizando el criterio propuesto en la sección 5.2 */`
    if $\vec{x}_{nueva}$ *es mejor que* $\vec{x}_i$ **then**
      $\vec{x}_i \leftarrow \vec{x}_{nueva}$;
    end
  end
end

El criterio de terminación descrito en la ec. 5.2 será utilizado tanto para el algoritmo EDPR como para el algoritmo híbrido propuesto.

Algoritmo 9: EDPR hibridizado con el método simplex

entrada: N_p (Tamaño de la población), g_{max} (generaciones máximas), N (número de variables), Cr (probabilidad de cruza), P_f (probabilidad de comparar con respecto al valor de la función objetivo), $f_{opSimplex}$ (frecuencia con la que se va a aplicar el operador simplex), $pGenApSim$ (generación en la que se empieza a aplicar el operador simplex) y m (dimensión del simplex generado en el operador simplex)

salida : $\vec{x}^*$ (mejor solución encontrada), $f(\vec{x}^*)$ y $contGen$ (número de generaciones requeridas)

$contGen \leftarrow 0$;
Generar la población inicial;
repeat
  if $contGen$ mód $f_{opSimplex} \neq 0$ *OR* $contGen < pGenApSim$ **then**
    Obtener la siguiente generación a partir de los operadores de mutación, cruza y selección del algoritmo EDPR;
  else
    Obtener la siguiente generación aplicando el operador simplex propuesto;
  end
  $contGen \leftarrow contGen + 1$;
until $contGen < g_{max}$ *AND No se cumpla criterio de terminación* ;
Obtener el mejor punto ($\vec{x}^*$) de la población (siguiendo el criterio propuesto en la sección 5.2);

Capítulo 6

Resultados experimentales

En este capítulo se muestran los resultados obtenidos con los dos algoritmos propuestos en el capítulo 5: Evolución Diferencial para Problemas con Restricciones (EDPR) y Algoritmo Híbrido de Evolución Diferencial y el método Simplex para Problemas con Restricciones (HEDSPR). Para validar dichos algoritmos, se utilizaron veintidós funciones de prueba estándar de la literatura de optimización global con restricciones propuestas en [45].

Finalmente, se realiza un estudio comparativo del algoritmo HEDSPR y dos de los algoritmos que han obtenido mejores resultados en la resolución de problemas de optimización con restricciones en la literatura especializada: Jerarquización estocástica [16] y el método de diversidad presentado en [1].

6.1. Funciones de prueba

En [45] se presentan los problemas de optimización con restricciones que adoptamos en esta tesis, con el siguiente formato:

$$\begin{aligned} &\text{Minimizar } f(\vec{x}),\ \vec{x} = [x_1, x_2, ..., x_n]^T \text{ sujeto a:} \\ &\\ &g_i(\vec{x}) \leq 0,\ i = 1, ..., q \\ &h_j(\vec{x}) \leq 0,\ j = q+1, ..., m \end{aligned} \tag{6.1}$$

Las restricciones de igualdad usualmente son transformadas en restricciones de desigualdad de la forma siguiente:

$$|h_j(\vec{x})| - \epsilon \leq 0, \text{ para } j = q+1, ..., m \tag{6.2}$$

Una solución es considerada como factible si $\forall i\ g_i(\vec{x}) \leq 0$ y $\forall j\ |h_j(\vec{x})| - \epsilon \leq 0$. Para esta tesis se fijó ϵ en 0.0001.

En el apéndice A se muestran a detalle cada una de las veintidós funciones utilizadas y en la tabla 6.1 se muestra un resumen de las características de las mismas.

	n	Tipo de función	ρ	DL	DN	IL	IN	a
g01	13	cuadrática	0.0111 %	9	0	0	0	6
g02	20	no lineal	99.9971 %	0	2	0	0	1
g03	10	polinomial	0.0 %	0	0	0	1	1
g04	5	cuadrática	52.123 %	0	6	0	0	2
g05	4	cúbica	0.0 %	2	0	0	3	3
g06	2	cúbica	0.0066 %	0	2	0	0	2
g07	10	cuadrática	0.0003 %	3	5	0	0	6
g08	2	no lineal	0.8560 %	0	2	0	0	0
g09	7	polinomial	0.5121 %	0	4	0	0	2
g10	8	lineal	0.001 %	3	3	0	0	6
g11	2	cuadrática	0.0 %	0	0	0	1	1
g12	3	cuadrática	4.7713 %	0	1	0	0	0
g13	5	no lineal	0.0 %	0	0	0	3	3
g14	10	no lineal	0.0 %	0	0	3	0	3
g15	3	cuadrática	0.0 %	0	0	1	1	2
g16	5	no lineal	0.0204 %	4	34	0	0	4
g17	6	no lineal	0.0 %	0	0	0	4	4
g18	9	cuadrática	0.0 %	0	13	0	0	6
g19	15	no lineal	33.4761 %	0	5	0	0	0
g21	7	lineal	0.0 %	0	1	0	5	6
g23	9	lineal	0.0 %	0	2	3	1	6
g24	2	lineal	79.6556 %	0	2	0	0	2

Tabla 6.1: Características de las 22 funciones utilizadas. n es el número de variables de decisión, $\rho = \frac{|F|}{|S|}$ es la tasa estimada entre la región factible y el espacio de búsqueda[2], DL es el número de restricciones de desigualdad lineales, DN es el número de restricciones de desigualdad no lineales, IL es el número de restricciones de igualdad lineales, IN es el número de restricciones de igualdad no lineales y a es el número de restricciones activas en el óptimo.

[2]Este valor se obtuvo generando un millón de soluciones aleatorias para cada problema, y dividiendo la cantidad de soluciones factibles obtenidas entre el total.

6.2. Diseño experimental e interpretación de los datos

Los algoritmos utilizados para resolver problemas de optimización generan una solución candidata a un problema dado, cada vez que se ejecutan, teniendo como objetivo encontrar una solución que corresponda con el valor óptimo global del problema que se está resolviendo.

Por lo tanto, el experimento consiste en efecutar 100 ejecuciones independientes, con diferente semilla inicial para el generador de números aleatorios, del algoritmo propuesto para cada una de las veintidós funciones de prueba. Una vez realizadas las ejecuciones para una determinada función, obtenemos 100 soluciones, las cuales conformarán nuestro conjunto de datos experimentales Ω. Posteriormente, se aplica una análisis estadístico al conjunto Ω y se presentan los siguientes datos, para cada función:

- La mejor, la mediana y la peor solución
- La media aritmética y la desviación estándar
- Valor de la función de violación de las restricciones, v, en la solución mediana:

$$v = \sum_{i=1}^{q} G_i(\vec{x}) + \sum_{i=q+1}^{m} H_i(\vec{x}), \tag{6.3}$$

donde:

$$\begin{aligned} G_i(\vec{x}) &= \begin{cases} g_i(\vec{x}) & \text{Si } g_i(\overrightarrow{x}) > 0 \\ 0 & \text{Si } g_i(\vec{x}) \leq 0, \text{ y} \end{cases} \\ H_j(\vec{x}) &= \begin{cases} |h_j(\vec{x})| & \text{Si } |h_j(\vec{x})| - \epsilon > 0 \\ 0 & \text{Si } |h_j(\vec{x})| - \epsilon \leq 0 \end{cases} \end{aligned} \tag{6.4}$$

- Tasa de factibilidad, tasa de éxito y costo.
- Gráficas de convergencia: De la solución mediana se realizan dos gráficas: una muestra la convergencia hacia el valor óptimo. Para ello graficamos en el eje de las x el número de evaluaciones de la función objetivo y en el eje de las y el valor de $\log_{10}(f(\vec{x}) - f(\vec{x}^*))$. La otra gráfica muestra la convergencia hacia la zona factible, para lo cual graficamos en el eje de las x el número de evaluaciones de la función objetivo y en el eje de las y el valor de $\log_{10}(\vec{v})$.

Sean x_1, x_2,..., x_n, los datos de una muestra ordenada, entonces se define **la mediana** como sigue:

$$\begin{aligned} M_e &= x_{(n+1)/2}, \text{ si } n \text{ es impar} \\ M_e &= \tfrac{1}{2}\left(x_{n/2} + x_{(n/2)+1}\right), \text{ si } n \text{ es par} \end{aligned} \tag{6.5}$$

En la ec. 6.5 observamos que si n es impar entonces la mediana será la observación central de los valores, una vez que éstos han sido ordenados en orden creciente o decreciente. Si n es par entonces la mediana es el promedio de las dos observaciones centrales.

La **media aritmética** o **promedio**, de una cantidad finita de números, es igual a la suma de todos ellos dividida entre el número de sumandos. Se puede decir que la media aritmética es la cantidad total de la variable distribuida a partes iguales entre cada observación. Usualmente, se denota como $\overline{X}$ si se trata de la media de una muestra de datos y con la letra μ si se trata de la media de una población. Dados n números x_1, x_2, ..., x_n, la media aritmética se define como:

$$\mu = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i \tag{6.6}$$

Para conocer a fondo un conjunto de datos, no es suficiente conocer las medidas de tendencia central, sino que es necesario conocer también la desviación que presentan los datos con respecto a la media aritmética de dicha distribución, con la finalidad de lograr una visión de los mismos más acorde con la realidad, a la hora de describirlos e interpretarlos. La desviación estándar es una medida de dispersión usada en estadística que nos indica cuánto tienden a alejarse los valores puntuales del promedio en una distribución. Una desviación estándar grande indica que los puntos están lejos de la media, y una desviación pequeña indica que los datos están agrupados cerca de la media.

La **desviación estándar** está estrechamente relacionada con la varianza ya que se define una a partir de la otra. 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; y, si por el contrario, sólo consideramos una muestra de la población, obtenemos una varianza muestral. En las ecs. 6.7 y 6.8 se muestra la definición matématica para la varianza poblacional y la desviación estándar, respectivamente.

$$\sigma^2 = \frac{1}{N} \cdot \sum_{i=1}^{N} (X_i - \mu)^2 \tag{6.7}$$

$$\sigma = \sqrt{\sigma^2} \tag{6.8}$$

La **tasa de factibilidad** es la relación entre el número de ejecuciones factibles y el total de las ejecuciones. Una ejecución es considerada como factible si la solución generada es factible. En la ec. 6.9 se muestra dicha relación.

$$tasa\ de\ factibilidad = \frac{\text{número de ejecuciones factibles}}{\text{total de ejecuciones}} \tag{6.9}$$

La **tasa de éxito** es la relación entre el número de ejecuciones exitosas y el total de ejecuciones. Una ejecución es considerada como exitosa si la solución encontrada $\vec{x}$ satisface $f(\vec{x}) - f(\vec{x}^*) \leq 0.0001$. En la ec. 6.10 se muestra dicha relación.

$$tasa\ de\ éxito = \frac{\text{número de ejecuciones exitosas}}{\text{total de ejecuciones}} \tag{6.10}$$

Finalmente, el **costo** se define como sigue:

$$costo = \frac{media\left(\text{número de evaluaciones de las ejecuciones exitosas}\right) \cdot \left(\text{total de ejecuciones}\right)}{\left(\text{número de ejecuciones exitosas}\right)} \tag{6.11}$$

Nótese que la ec. 6.11 es una medida de costo que nos indica cuantas evaluaciones de la función objetivo le cuesta al algoritmo para llegar a un cierto número de ejecuciones exitosas.

6.3. Parámetros de control utilizados

El algoritmo EDPR utiliza los siguientes parámetros: F (factor del rango en que se desenvuelve la población), Cr (probabilidad de cruza), N_p (tamaño de la población), y G_{max} (generaciones máximas) para el algoritmo de ED y P_f (probabilidad de comparar con respecto al valor de la función objetivo) para el manejo de restricciones.

En la literatura especializada, el factor F se encuentra en el intervalo $(0, 1+)$ [14]. Sin embargo, en [46] se determinó de forma experimental que el comportamiento del algoritmo no tiene efectos significativos al generarlo de manera aleatoria en el intervalo $[0.3, 0.9]$ para cada generación. Por lo anterior, en las pruebas realizadas en esta tesis no se establece un valor predeterminado para el factor F.

Para cada función de prueba se utiliza una recombinación discreta binomial, también conocida como recombinación uniforme, y la probabilidad de cruza, $Cr \in [0, 1]$, se fijó en $Cr = 0.99$ ya que se obtiene un comportamiento aceptable para la mayoría de las funciones de prueba. Con respecto al tamaño de la población, se observó un buen comportamiento del algoritmo al utilizar una población de 60 individuos. Ambos valores se determinaron a partir de una búsqueda exhaustiva. El número máximo de generaciones se fijó de tal forma que el algoritmo realizara a lo más 180,000. Para ello, se fijó G_{max} en 3000.

Finalmente, el parámetro P_f también fue determinado realizando una búsqueda exhaustiva. Recordemos que este parámetro indica la probabilidad de comparar a los individuos de la población con respecto al valor de la función objetivo y, por lo tanto, está en el rango $[0, 1]$. P_f depende del tipo de restricciones del problema. Sin embargo, en las pruebas realizadas con las veintidós funciones, se observaron buenos resultados en el intervalo $[0.0, 0.3]$. Por ello, se decidió generar un número aleatorio en dicho intervalo para determinar el valor de P_f que se utiliza en cada generación.

El algoritmo HEDSPR, además de adoptar los parámetros utilizados por el algoritmo EDPR, utiliza los siguientes: γ (factor de expansión), β (factor de contracción), m (dimensión del simplex), $fopSimplex$ (frecuencia con la que se aplica el operador basado en el método simplex) y $pgenApSim$ (generación a partir de la cual se empieza a aplicar el operador simplex). Todos ellos son utilizados por el operador simplex que es incorporado.

Los parámetros γ y β se fijaron en 1.3 y 0.5 respectivamente, debido a que en la literatura se reportan buenos resultados con dichos valores. Los parámetros restantes se fijaron en $m = 2$, $pgenApSim = 1$ y $fopSimplex = 10$. Dichos valores se determinaron tras realizar numerosos experimentos.

Es importante aclarar que debido al funcionamiento del operador simplex, el algoritmo HEDSPR puede realizar más de 60 evaluaciones de la función objetivo en las generaciones que usen dicho operador ya que por cada individuo se pueden realizar hasta tres evaluaciones. Por lo tanto, usando los valores antes mencionados se realizan 216,000 evaluaciones de la función objetivo en el peor caso.

6.4. Resultados obtenidos con los algoritmos EDPR y HEDSPR

Los resultados se agrupan de la siguiente forma:

- La tabla 6.2 muestra los resultados obtenidos, tomando en cuenta la solución obtenida $(\vec{x}, f(\vec{x}))$, para los experimentos efectuados con los algoritmos EDPR y HEDSPR.

- La tabla 6.3 muestra los resultados obtenidos, tomando en cuenta el número de evaluaciones de la función objetivo requeridas para obtener la solución final $(\vec{x}, f(\vec{x}))$, para los experimentos efectuados con los algoritmos EDPR y HEDSPR. En este análisis estadístico sólo se consideraron las ejecuciones exitosas. En la primera columna se indica el nombre del algoritmo que obtuvo un mejor desempeño.

De acuerdo a los resultados presentados en las tablas 6.2 y 6.3 podemos observar que ambos algoritmos obtienen buenos resultados en la mayoría de las funciones de prueba, pues encuentran el óptimo en veintiún funciones de las veintidós utilizadas. Además, en la mayoría de los casos obtienen un porcentaje de éxito de 100 %. Es importante mencionar, que de acuerdo a las pruebas realizadas, los algoritmos EDPR y HEDSPR tienen problemas en las funciones $g01$ y $g02$. Sin embargo, este problema se resuelve al ajustar el valor del parámetro de cruza Cr y del parámetro usado para el manejo de restricciones, P_f. En la sección 6.6 se explica a detalle el ajuste de dichos parámetros. Con respecto al número de evaluaciones de la función objetivo, que realizan los algoritmos, podemos concluir que ambos obtienen buenos resultados. EDPR realiza un máximo de 180,000 evaluaciones y HEDSPR realiza un máximo de 216,000 evaluaciones.

En la tabla 6.3 podemos observar que EDPR es superado por HEDSPR en nueve de las veintidós funciones de prueba, en cuatro funciones obtienen resultados similares y en las nueve funciones restantes EDPR supera a HEDSPR. Es importante mencionar que aunque el operador simplex ayuda a mejorar significativamente el desempeño de EDPR (ya sea mejorando las tasas de factibilidad y éxito o disminuyendo el número de evaluaciones de la función objetivo) en nueve funciones en otras nueve afecta adversamente el desempeño del mismo. Este problema puede disminuir ajustando los parámetros m, $fopSimplex$ y $pgenApSim$, como se verá en la sección 6.6.

Estadísticas con respecto a la solución obtenida $(\vec{x}, f(\vec{x}))$			
ALGORITMO Máximo de evaluaciones de f		EDPR 180,000	HEDSPR 180,000-216,000
g01 -15.000000	mejor	**-15.000000**	**-15.000000**
	mediana	**-14.999999**	**-14.999999**
	peor	**-9.000000**	**-6.000000**
	v	**0.000000**	**0.000000**
	media	**-13.755468**	**-13.260937**
	Desv. Est.	**1.234172**	**1.702781**
g02 -0.803619	mejor	-0.793147	-0.617171
	mediana	-0.793147	-0.617171
	peor	-0.391592	-0.309712
	v	0.000000	0.000000
	media	-0.525320	-0.423918
	Desv. Est.	0.055183	0.043355

Estadísticas con respecto a la solución obtenida $(\vec{x}, f(\vec{x}))$			
ALGORITMO Máximo de evaluaciones de f		EDPR 180,000	HEDSPR 180,000-216,000
g03 -1.000500	mejor	**-1.000500**	**-1.000500**
	mediana	**-1.000500**	**-1.000495**
	peor	**-0.000000**	**-0.000000**
	v	**0.000000**	**0.000000**
	media	**-0.950472**	**-0.949563**
	Desv. Est.	**0.095047**	**0.094956**
g04 -30665.538672	mejor	**-30665.538672**	**-30665.538672**
	mediana	**-30665.538672**	**-30665.538672**
	peor	**-30665.538671**	**-30665.538671**
	v	**0.000000**	**0.000000**
	media	**-30665.538671**	**-30665.538672**
	Desv. Est.	**0.000000**	**0.000000**
g05 5126.496714	mejor	**5126.496714**	**5126.496714**
	mediana	**5126.496714**	**5126.496714**
	peor	**5126.496714**	**5126.496714**
	v	**0.000000**	**0.000000**
	media	**5126.496714**	**5126.496714**
	Desv. Est.	**0.000000**	**0.000000**
g06 -6961.813876	mejor	**-6961.813876**	**-6961.813876**
	mediana	**-6961.813875**	**-6961.813875**
	peor	**-7950.961894**	**-6961.813875**
	v	**0.000000**	**0.000000**
	media	**-6929.749586**	**-6961.813875**
	Desv. Est.	**103.053214**	**0.000000**
g07 24.306209	mejor	**24.306209**	**24.306209**
	mediana	**24.306209**	**24.306209**
	peor	**24.306288**	**24.306915**
	v	**0.000000**	**0.000000**
	media	**24.306211**	**24.306225**
	Desv. Est.	**0.000003**	**0.000025**
g08 -0.095825	mejor	**-0.095826**	**-0.095826**
	mediana	**-0.095826**	**-0.095826**
	peor	**-0.095826**	**-0.095826**
	v	**0.000000**	**0.000000**
	media	**-0.095826**	**-0.095826**
	Desv. Est.	**0.000000**	**0.000000**
g09 680.630057	mejor	**680.630057**	**680.630057**
	mediana	**680.630057**	**680.630057**
	peor	**680.630057**	**680.630057**
	v	**0.000000**	**0.000000**
	media	**680.630057**	**680.630057**
	Desv. Est.	**0.000000**	**0.000000**

Estadísticas con respecto a la solución obtenida $(\vec{x}, f(\vec{x}))$			
ALGORITMO Máximo de evaluaciones de f		EDPR 180,000	HEDSPR 180,000-216,000
g10 7049.248021	mejor	**7049.248021**	**7049.248021**
	mediana	**7049.248021**	**7049.248021**
	peor	**7049.248023**	**7049.248021**
	v	**0.000000**	**0.000000**
	media	**7049.248021**	**7051.265214**
	Desv. Est.	**0.000000**	**3.994042**
g11 0.749900	mejor	**0.749900**	**0.749900**
	mediana	**0.749900**	**0.749900**
	peor	**0.749900**	**0.749873**
	v	**0.000000**	**0.000000**
	media	**0.749900**	**0.749905**
	Desv. Est.	**0.000000**	**0.000010**
g12 -1.000000	mejor	**-1.000000**	**-1.000000**
	mediana	**-1.000000**	**-1.000000**
	peor	**-1.000000**	**-1.000000**
	v	**0.000000**	**0.000000**
	media	**-1.000000**	**-1.000000**
	Desv. Est.	**0.000000**	**0.000000**
g13 0.053942	mejor	**0.053942**	**0.053942**
	mediana	**0.053942**	**0.053942**
	peor	**1.000000**	**0.438803**
	v	**0.000000**	**0.000000**
	media	**0.294319**	**0.238675**
	Desv. Est.	**0.187494**	**0.192123**
g14 -47.764888	mejor	**-47.764888**	**-47.764888**
	mediana	**-47.764888**	**-47.764888**
	peor	**-44.664015**	**-47.622320**
	v	**0.000000**	**0.000000**
	media	**-47.623774**	**-47.647472**
	Desv. Est.	**0.231465**	**0.191210**
g15 961.715022	mejor	**961.715022**	**961.715022**
	mediana	**961.715022**	**961.715022**
	peor	**961.715022**	**961.715022**
	v	**0.000000**	**0.000000**
	media	**961.715022**	**961.715022**
	Desv. Est.	**0.000000**	**0.000000**
g16 -1.905155	mejor	**-1.905155**	**-1.905155**
	mediana	**-1.905155**	**-1.905155**
	peor	**-1.905155**	**-1.905155**
	v	**0.000000**	**0.000000**
	media	**-1.905155**	**-1.905155**
	Desv. Est.	**0.000000**	**0.000000**

Estadísticas con respecto a la solución obtenida $(\vec{x}, f(\vec{x}))$			
ALGORITMO Máximo de evaluaciones de f		EDPR 180,000	HEDSPR 180,000-216,000
g17 8853.539675	mejor	**8853.533875**	**8853.533875**
	mediana	**8853.533875**	**8853.533875**
	peor	**8853.533875**	**8927.591751**
	v	**0.000000**	**0.000000**
	media	**8866.031621**	**8877.299910**
	Desv. Est.	**20.774853**	**32.204891**
g18 -0.866025	mejor	**-0.866025**	**-0.866025**
	mediana	**-0.866025**	**-0.866025**
	peor	**-0.866017**	**-0.674981**
	v	**0.000000**	**0.000000**
	media	**-0.863330**	**-0.841190**
	Desv. Est.	**0.005099**	**0.043214**
g19 32.655593	mejor	**32.655593**	**32.655593**
	mediana	**32.655593**	**32.655593**
	peor	**32.655999**	**38.476427**
	v	**0.000000**	**0.000000**
	media	**32.655597**	**32.714055**
	Desv. Est.	**0.000008**	**0.115247**
g21 193.724510	mejor	**193.724510**	**193.724510**
	mediana	**193.724510**	**193.724511**
	peor	**0.000000**	**320.113536**
	v	**0.000000**	**0.000000**
	media	**269.000788**	**269.050345**
	Desv. Est.	**64.088882**	**63.271893**
g23 -400.055100	mejor	**-400.055100**	**-400.055100**
	mediana	**-400.055099**	**-400.055099**
	peor	**900.000000**	**-2100.000000**
	v	**0.000000**	**0.000000**
	media	**-439.033082**	**-358.885950**
	Desv. Est.	**365.412722**	**325.706370**
g24 -5.508013	mejor	**-5.508013**	**-5.508013**
	mediana	**-5.508013**	**-5.508013**
	peor	**-5.508013**	**-5.508013**
	v	**0.000000**	**0.000000**
	media	**-5.508013**	**-5.508013**
	Desv. Est.	**0.000000**	**0.000000**

Tabla 6.2: Resultados obtenidos con los algoritmo EDPR y HEDSPR, considerando la solución obtenida, $(\vec{x}, f(\vec{x}))$. v corresponde al valor de la función de restricciones de la solución mediana. Los valores en negritas indican que en dicha función el algoritmo encontró el óptimo.

Estadísticas con respecto al número de evaluaciones de la función objetivo			
ALGORITMO Máximo de evaluaciones de f		EDPR 180,000	HEDSPR 180,000-216,000
	mejor	27120	27787
	mediana	36840	36518
g01	peor	47760	49619
-15.000000	media	36841.224490	36500.191489
	Desv. Est.	2719.616826	3282.144862
EDPR	TF	100.000000	100.000000
	TE	49.000000	47.000000
	Costo	75186.172428	77659.981892
	mejor	-	-
	mediana	-	-
	peor	-	-
g02	media	-	-
-0.803619	Desv. Est.	-	-
	TF	100.000000	100.000000
	TE	0.000000	0.000000
	Costo	-	-
	mejor	180000	209922
	mediana	180000	211150
g03	peor	180000	211970
-1.000500	media	180000.000000	211109.381818
	Desv. Est.	0.000000	383.060496
EDPR	TF	100.000000	81.000000
	TE	94.000000	55.000000
	Costo	191489.361702	383835.239669
	mejor	**11160**	**11376**
	mediana	**14760**	**13689**
g04	peor	**17880**	**16209**
-30665.538672	media	**14772.000000**	**13640.880000**
	Desv. Est.	**903.840000**	**892.882400**
HEDSPR	TF	**100.000000**	**100.000000**
	TE	**100.000000**	**100.000000**
	Costo	**14772.000000**	**13640.880000**
	mejor	**99120**	**120421**
	mediana	**180000**	**210427**
g05	peor	**180000**	**210878**
5126.496714	media	**179191.200000**	**209224.310000**
	Desv. Est.	**1601.424000**	**2045.722000**
EDPR	TF	**100.000000**	**100.000000**
	TE	**100.000000**	**100.000000**
	Costo	**179191.200000**	**209224.310000**

Estadísticas con respecto al número de evaluaciones de la función objetivo			
ALGORITMO Máximo de evaluaciones de f		EDPR 180,000	HEDSPR 180,000-216,000
	mejor	8580	**8633**
	mediana	9660	**9922**
g06	peor	12900	**11392**
-6961.813876	media	9756.494845	**9921.790000**
	Desv. Est.	424.623233	**465.570000**
HEDSPR	TF	98.000000	**100.000000**
	TE	97.000000	**100.000000**
	Costo	10058.242109	**9921.790000**
	mejor	**127800**	92358
	mediana	**180000**	185644
	peor	**180000**	200500
g07	media	**176707.200000**	171424.659794
24.306209	Desv. Est.	**5626.416000**	28772.022106
	TF	**100.000000**	100.000000
	TE	**100.000000**	97.000000
	Costo	**176707.200000**	176726.453396
	mejor	**3480**	**3503**
	mediana	**3900**	**3903**
g08	peor	**4500**	**4604**
-0.095825	media	**3950.400000**	**3942.280000**
	Desv. Est.	**172.608000**	**165.839200**
HEDSPR	TF	**100.000000**	**100.000000**
	TE	**100.000000**	**100.000000**
	Costo	**3950.400000**	**3942.280000**
	mejor	**31680**	**27143**
	mediana	**36960**	**32126**
g09	peor	**48960**	**40101**
680.630057	media	**37305.600000**	**32178.090000**
	Desv. Est.	**1936.272000**	**1604.939000**
HEDSPR	TF	**100.000000**	**100.000000**
	TE	**100.000000**	**100.000000**
	Costo	**37305.600000**	**32178.090000**
	mejor	**180000**	203284
	mediana	**180000**	205372
g10	peor	**180000**	206858
7049.248021	media	**180000.000000**	205311.343434
	Desv. Est.	**0.000000**	629.538210
EDPR	TF	**100.000000**	100.000000
	TE	**100.000000**	99.000000
	Costo	**180000.000000**	207385.195388

Estadísticas con respecto al número de evaluaciones de la función objetivo			
ALGORITMO Máximo de evaluaciones de f		EDPR 180,000	HEDSPR 180,000-216,000
	mejor	**16620**	199344
	mediana	**20220**	199895
g11	peor	**43740**	200974
0.749900	media	**21384.600000**	199942.272727
	Desv. Est.	**2509.764000**	219.461629
EDPR	TF	**100.000000**	79.000000
	TE	**100.000000**	77.000000
	Costo	**21384.600000**	259665.289256
	mejor	**9780**	**8201**
	mediana	**12060**	**11066**
g12	peor	**15060**	**15024**
-1.000000	media	**12103.200000**	**10985.550000**
	Desv. Est.	**910.464000**	**940.817000**
HEDSPR	TF	**100.000000**	**100.000000**
	TE	**100.000000**	**100.000000**
	Costo	**12103.200000**	**10985.550000**
	mejor	35400	30483
	mediana	127260	144516
g13	peor	180000	202274
0.053942	media	121146.153846	137147.557692
	Desv. Est.	28789.349112	27697.411982
HEDSPR	TF	100.000000	100.000000
	TE	39.000000	52.000000
	Costo	310631.163708	263745.303254
	mejor	105840	91781
	mediana	180000	168467
	peor	180000	198999
g14	media	175647.272727	161484.557377
-47.764888	Desv. Est.	7134.545455	29387.327600
	TF	100.000000	100.000000
	TE	66.000000	61.000000
	Costo	266132.231405	264728.782585
	mejor	**18120**	**16434**
	mediana	**21360**	**19755**
g15	peor	**57720**	**71744**
961.715022	media	**23061.000000**	**23490.640000**
	Desv. Est.	**3623.220000**	**6423.162400**
EDPR	TF	**100.000000**	**100.000000**
	TE	**100.000000**	**100.000000**
	Costo	**23061.000000**	**23490.640000**

Estadísticas con respecto al número de evaluaciones de la función objetivo			
ALGORITMO Máximo de evaluaciones de f		EDPR 180,000	HEDSPR 180,000-216,000
	mejor	**27660**	**25923**
	mediana	**32340**	**28700**
g16	peor	**35100**	**32980**
-1.905155	media	**32266.200000**	**28869.880000**
	Desv. Est.	**1159.152000**	**1185.637600**
HEDSPR	TF	**100.000000**	**100.000000**
	TE	**100.000000**	**100.000000**
	Costo	**32266.200000**	**28869.880000**
	mejor	180000	195842
	mediana	180000	201892
g17	peor	180000	203193
8853.539675	media	180000.000000	201450.873016
	Desv. Est.	0.000000	1033.328294
EDPR	TF	100.000000	100.000000
	TE	77.000000	63.000000
	Costo	233766.233766	319763.290501
	mejor	142560	64899
	mediana	180000	83611
	peor	180000	207002
g18	media	178378.666667	111289.517241
-0.866025	Desv. Est.	3026.488889	44459.093143
	TF	99.000000	100.000000
	TE	90.000000	87.000000
	Costo	198198.518519	127918.985335
	mejor	126660	146218
	mediana	180000	201381
g19	peor	180000	201973
32.655593	media	177425.454545	198639.293478
	Desv. Est.	4159.669421	4885.911626
EDPR	TF	100.000000	100.000000
	TE	99.000000	92.000000
	Costo	179217.630854	215912.275520
	mejor	36720	31605
	mediana	43020	37720
g21	peor	54900	97315
193.724510	media	43227.857143	40211.102564
	Desv. Est.	3758.418367	4964.177515
HEDSPR	TF	69.000000	86.000000
	TE	28.000000	39.000000
	Costo	154385.204082	103105.391190

Estadísticas con respecto al número de evaluaciones de la función objetivo			
ALGORITMO Máximo de evaluaciones de f		EDPR 180,000	HEDSPR 180,000-216,000
g23 -400.055100 HEDSPR	mejor	89640	101597
	mediana	157140	155721
	peor	180000	203191
	media	155669.090909	160472.891304
	Desv. Est.	20813.663912	31267.620983
	TF	88.000000	92.000000
	TE	33.000000	46.000000
	Costo	471724.517906	348854.111531
g24 -5.508013 EDPR	mejor	**5100**	**5549**
	mediana	**6000**	**6190**
	peor	**6780**	**7094**
	media	**6036.600000**	**6173.040000**
	Desv. Est.	**242.064000**	**234.716000**
	TF	**100.000000**	**100.000000**
	TE	**100.000000**	**100.000000**
	Costo	**6036.600000**	**6173.040000**

Tabla 6.3: Resultados obtenidos con el algoritmo EDPR, considerando el número de evaluaciones de la función objetivo, f, requeridas para obtener la solución $(\vec{x}, f(\vec{x}))$. Donde TF es la tasa de factibilidad y TE es la tasa de éxito. Los valores en negritas indican que en dicha función el algoritmo obtuvo un porcentaje de éxito de 100 %.

6.5. Estudio comparativo entre la jerarquización estocástica, Diversity-DE, EDPR y HEDSPR

Tanto la jerarquización estocástica como Diversity-DE fueron validados por sus autores usando sólo las primeras trece funciones de las veintidós utilizadas en esta tesis. Los resultados presentados por ambas propuestas se muestran en la tabla 6.4. Estos algoritmos realizan 350,000 y 225,000 evaluaciones de la función objetivo, respectivamente, para encontrar los óptimos de las trece funciones.

ALGORITMO Evaluaciones de f		ISR 350,000	Diversity-DE 225,000	EDPR 180,000	HEDSPR 216,000
	mejor	-15.000	-15.000	-15.000	-15.000
g01	media	-15.000	-15.000	-13.755	-13.2609
-15.000	peor	-15.000	-15.000	-9.000	-6.000
	Desv. Est.	5.8E-14	1.0E-9	1.234	1.7027
	mejor	-0.803619	-0.803619	**-0.793147**	**-0.617171**
g02	media	-0.782715	-0.798079	**-0.525320**	**-0.423918**
-0.803619	peor	-0.723591	-0.751742	**-0.391592**	**-0.309712**
	Desv. Est.	2.2E-9	1.01E-2	**5.51E-2**	**4.33E-2**
	mejor	-1.001	-1.000	-1.0005	-1.0005
g03	media	-1.001	-1.000	-0.950472	-0.949563
-1.0005	peor	-1.001	-1.000	0	0
	Desv. Est.	8.2E-9	0	9.5E-2	9.49E-2
	mejor	-30665.539	-30665.539	-30665.538672	-30665.538672
g04	media	-30665.539	-30665.539	-30665.538672	-30665.538672
-30665.538672	peor	-30665.539	-30665.539	-30665.538672	-30665.538672
	Desv. Est.	1.1E-11	0	0	0
	mejor	5126.497	5126.497	5126.496714	5126.496714
g05	media	5126.497	5126.497	5126.496714	5126.496714
5216.496714	peor	5126.497	5126.497	5126.496714	5126.496714
	Desv. Est.	7.2E-13	0	0	0
	mejor	-6961.814	-6961.814	-6961.813876	-6961.813876
g06	media	-6961.814	-6961.814	-6929.749586	-6961.813875
-6961.813876	peor	-6961.814	-6961.814	-7950.961894	-6961.813875
	Desv. Est.	1.9E-12	0	1.03E+2	0
	mejor	24.306	24.306	24.306209	24.306209
g07	media	24.306	24.306	24.306211	24.306225
24.306209	peor	24.306	24.306	24.306288	24.306915
	Desv. Est.	6.3E-5	8.22E-9	3.0E-6	2.5E-5
	mejor	-0.095825	-0.095825	-0.095825	-0.095825
g08	media	-0.095825	-0.095825	-0.095825	-0.095825
-0.095825	peor	-0.095825	-0.095825	-0.095825	-0.095825
	Desv. Est.	2.7E-17	0	0	0
	mejor	680.630	680.630	680.630057	680.630057
g09	media	680.630	680.630	680.630057	680.630057
680.630057	peor	680.630	680.630	680.630057	680.630057
	Desv. Est.	3.2E-13	0	0	0
	mejor	7049.248	7049.248	7049.248021	7049.248021
g10	media	7049.250	7049.266	7049.248021	7049.248021
7049.248021	peor	7049.270	7049.617	7049.248021	7049.248021
	Desv. Est.	3.2E-3	4.45E-2	0	0
	mejor	0.75	0.75	0.7499	0.7499
g11	media	0.75	0.75	0.7499	0.7499
0.7499	peor	0.75	0.75	0.7499	0.7498
	Desv. Est.	1.1E-16	0	0	1E-5
	mejor	1.000	1.000	1.000	1.000
g12	media	1.000	1.000	1.000	1.000
1.000	peor	1.000	1.000	1.000	1.000
	Desv. Est.	1.2E-9	0	0	0
	mejor	0.053942	0.053941	0.053942	0.053942
g13	media	0.06677	0.069336	0.294319	0.238675
0.053942	peor	0.438803	0.438803	1.0	0.438803
	Desv. Est.	7.0E-2	7.58E-2	1.87E-1	1.92E-1

Tabla 6.4: Tabla comparativa de los algoritmos: jerarquización estocástica (ISR), Diversity-DE, EDPR y HEDSPR. Se toman los resultados reportados en [1] para los algoritmos de jerarquización estocástica y Diversity-DE. Los valores resaltados en negritas indican que en dicha función el algoritmo no logro localizar el óptimo.

A partir de los resultados de la tabla 6.4 podemos observar que los algoritmos EDPR y HEDSPR son competitivos con respecto a estas dos técnicas, ya que encuentran las soluciones óptimas en la mayoría de las funciones de prueba, con un número menor de evaluaciones de la función objetivo. En la función $g02$, tanto EDPR como HEDSPR tienen problemas para encontrar el óptimo y en la función $g01$ se llega al óptimo pero con una desviación estándar mayor. Sin embargo, estos problemas quedan solucionados al ajustar el parámetro de cruza Cr y el parámetro utilizado para el manejo de restricciones, P_f. Dicho ajuste se presenta en la sección 6.6.

Por último, es importante mencionar que el mecanismo de manejo de restricciones utilizado en los algoritmos propuestos es muy simple y, por lo tanto, mucho menos complejo que los utilizados en el algoritmo de jerarquización estocástica y el método de diversidad, contra los cuales ha sido comparado.

6.6. Ajuste empírico de los parámetros de control

Como se mencionó en secciones anteriores el algoritmo EDPR tiene dificultades en dos funciones de prueba, $g01$ y $g02$, al usar como parámetros $Cr = 0.99$ y $P_f = rand(0.0, 0.3)$. Sin embargo, este problema queda resuelto al ajustar dichos parámetros. Para ello, se realizaron una vez más las pruebas antes mencionadas, usando los valores de la tabla 6.5 para dichas funciones. Estos valores fueron determinados tras realizar numerosos experimentos.

En el resto de las funciones se utilizaron una vez más los valores antes mencionados ($Cr = 0.99$ y $P_f = rand(0.0, 0.3)$). Recordemos que en el caso del parámetro P_f, éste se genera para cada generación.

	g01	g02
Cr	0.6	0.4
P_f	0.35	0.0

Tabla 6.5: Valores de los parámetros Cr y P_f utilizados para las funciones $g01$ y $g02$.

Con respecto al algoritmo HEDSPR tras realizar varios experimentos se observó que el algoritmo obtiene mejores resultados en trece de las veintidós funciones al usar como parámetros los valores de la tabla 6.6.

	g01	g02	g03	g04	g05	g10	g15
m	3	4	2	3	4	2	2
$fopSimplex$	5	10	10	1	2	10	10
$pgenApSim$	500	2500	2000	1	2900	2000	200

	g16	g17	g18	g19	g23	g24
m	3	3	2	2	3	2
$fopSimplex$	1	5	10	10	2	10
$pgenApSim$	1	600	2000	2000	1000	50

Tabla 6.6: Valores de los parámetros m, $fopSimlex$ y $pgenApSim$ utilizados por el algoritmo HEDSPR.

6.7. Resultados obtenidos con el ajuste empírico de parámetros (Algoritmos: EDPR y HEDSPR)

Los resultados se agrupan de la siguiente forma:

- La tabla 6.7 muestra los resultados obtenidos, tomando en cuenta la solución obtenida $(\vec{x}, f(\vec{x}))$, para los experimentos efectuados con los algoritmos EDPR y HEDSPR sin utilizar el ajuste de parámetros y utilizándolo. Los espacios en blanco significan que en dicha función no se realizó un ajuste.

- La tabla 6.8 muestra los resultados obtenidos, tomando en cuenta el número de evaluaciones de la función objetivo requeridas para obtener la solución final $(\vec{x}, f(\vec{x}))$, para los experimentos efectuados con los algoritmos EDPR y HEDSPR sin utilizar el ajuste de parámetros y utilizándolo. Los espacios en blanco significan que en dicha función no se realizó un ajuste. En este análisis estadístico sólo se consideraron las ejecuciones exitosas. En la primera columna se indica el nombre del algoritmo que obtuvo un mejor desempeño.

En las tablas 6.7 y 6.8 se puede observar que una vez realizado el ajuste de parámetros se logra encontrar el óptimo en la función $g02$ y se mejora la tasa de éxito en la función $g01$. Además, el algoritmo HEDSPR supera al algoritmo EDPR en doce de las veintidós funciones de prueba, obtiene un desempeño similar en dos funciones, y en las ocho restantes es superado por el algoritmo EDPR. La tabla 6.8 nos deja visualizar con mayor claridad dicha mejora, ya sea en el número de evaluaciones de la función objetivo o en la tasa de éxito y factibilidad. Estas mejoras se ven reflejadas en el rendimiento del algoritmo.

Finalmente, analizamos tanto las funciones beneficiadas, por el operador simplex propuesto, como las funciones afectadas con el propósito de encontrar alguna relación en dichas funciones. Sin embargo, no se encontró una relación al analizar los siguientes aspectos:

- Tipo de función: lineal, cuadrática, cúbica, no lineal o polinomial
- Número de variables
- Tamaño de la región factible
- Diversidad existente en la población. Nótese que el operador algunas veces obtiene buenos resultados al aplicarse desde el inicio de la búsqueda y con una frecuencia alta e incluso llega a sustituir totalmente al operador de mutación utilizado por el algoritmo de evolución diferencial.

Por lo anterior, se piensa que su comportamiento está relacionado con la forma de la zona factible y que dicho problema puede solucionarse haciendo un análisis profundo de la forma en la se eligen los puntos que conformarán el *m-simplex*. Sin embargo, esto último ya no se comprobó en esta tesis.

Estadísticas con respecto a la solución obtenida $(\vec{x}, f(\vec{x}))$					
ALGORITMO Máximo de evaluaciones de f		EDPR 180,000	EDPR (Ajuste) 180,000	HEDSPR 180,000-216,000	HEDSPR (Ajuste) 180,000-216,000
g01 -15.000000	mejor	**-15.000000**	**-15.000000**	**-15.000000**	**-15.000000**
	mediana	**-14.999999**	**-15.000000**	**-14.999999**	**-15.000000**
	peor	**-9.000000**	**-15.000000**	**-6.000000**	**-15.000000**
	v	**0.000000**	**0.000000**	**0.000000**	**0.000000**
	media	**-13.755468**	**-15.000000**	**-13.260937**	**-15.000000**
	Desv. Est.	**1.234172**	**0.000000**	**1.702781**	**0.000000**
g02 -0.803619	mejor	-0.793147	-0.803504	-0.617171	**-0.803618**
	mediana	-0.793147	-0.803307	-0.617171	**-0.803616**
	peor	-0.391592	-0.764451	-0.309712	**-0.778310**
	v	0.000000	0.000000	0.000000	**0.000000**
	media	-0.525320	-0.800369	-0.423918	**-0.801279**
	Desv. Est.	0.055183	0.004724	0.043355	**0.003926**
g03 -1.000500	mejor	**-1.000500**		**-1.000500**	**-1.000500**
	mediana	**-1.000500**		**-1.000495**	**-1.000500**
	peor	**-0.000000**		**-0.000000**	**-0.705430**
	v	**0.000000**		**0.000000**	**0.000000**
	media	**-0.950472**		**-0.949563**	**-0.927505**
	Desv. Est.	**0.095047**		**0.094956**	**0.134292**
g04 -30665.538672	mejor	**-30665.538672**		**-30665.538672**	**-30665.538672**
	mediana	**-30665.538672**		**-30665.538672**	**-30665.538672**
	peor	**-30665.538671**		**-30665.538671**	**-30665.538671**
	v	**0.000000**		**0.000000**	**0.000000**
	media	**-30665.538671**		**-30665.538672**	**-30665.538672**
	Desv. Est.	**0.000000**		**0.000000**	**0.000000**
g05 5126.496714	mejor	**5126.496714**		**5126.496714**	**5126.496714**
	mediana	**5126.496714**		**5126.496714**	**5126.496714**
	peor	**5126.496714**		**5126.496714**	**5126.496714**
	v	**0.000000**		**0.000000**	**0.000000**
	media	**5126.496714**		**5126.496714**	**5126.496714**
	Desv. Est.	**0.000000**		**0.000000**	**0.000000**
g06 -6961.813876	mejor	**-6961.813876**		**-6961.813876**	
	mediana	**-6961.813875**		**-6961.813875**	
	peor	**-7950.961894**		**-6961.813875**	
	v	**0.000000**		**0.000000**	
	media	**-6929.749586**		**-6961.813875**	
	Desv. Est.	**103.053214**		**0.000000**	
g07 24.306209	mejor	**24.306209**		**24.306209**	**24.306209**
	mediana	**24.306209**		**24.306209**	**24.306209**
	peor	**24.306288**		**24.306915**	**24.306590**
	v	**0.000000**		**0.000000**	**0.000000**
	media	**24.306211**		**24.306225**	**24.306217**
	Desv. Est.	**0.000003**		**0.000025**	**0.000014**

Estadísticas con respecto a la solución obtenida ($\vec{x}, f(\vec{x})$)					
ALGORITMO Máximo de evaluaciones de f		EDPR 180,000	EDPR (Ajuste) 180,000	HEDSPR 180,000-216,000	HEDSPR (Ajuste) 180,000-216,000
	mejor	**-0.095826**		**-0.095826**	
	mediana	**-0.095826**		**-0.095826**	
g08	peor	**-0.095826**		**-0.095826**	
-0.095825	v	**0.000000**		**0.000000**	
	media	**-0.095826**		**-0.095826**	
	Desv. Est.	**0.000000**		**0.000000**	
	mejor	**680.630057**		**680.630057**	
	mediana	**680.630057**		**680.630057**	
g09	peor	**680.630057**		**680.630057**	
680.630057	v	**0.000000**		**0.000000**	
	media	**680.630057**		**680.630057**	
	Desv. Est.	**0.000000**		**0.000000**	
	mejor	**7049.248021**		**7049.248021**	**7049.248021**
	mediana	**7049.248021**		**7049.248021**	**7049.248021**
g10	peor	**7049.248023**		**7049.248021**	**7049.248021**
7049.248021	v	**0.000000**		**0.000000**	**0.000000**
	media	**7049.248021**		**7051.265214**	**7049.248021**
	Desv. Est.	**0.000000**		**3.994042**	**0.000000**
	mejor	**0.749900**		**0.749900**	
	mediana	**0.749900**		**0.749900**	
g11	peor	**0.749900**		**0.749873**	
0.749900	v	**0.000000**		**0.000000**	
	media	**0.749900**		**0.749905**	
	Desv. Est.	**0.000000**		**0.000010**	
	mejor	**-1.000000**		**-1.000000**	
	mediana	**-1.000000**		**-1.000000**	
g12	peor	**-1.000000**		**-1.000000**	
-1.000000	v	**0.000000**		**0.000000**	
	media	**-1.000000**		**-1.000000**	
	Desv. Est.	**0.000000**		**0.000000**	
	mejor	**0.053942**		**0.053942**	
	mediana	**0.053942**		**0.053942**	
g13	peor	**1.000000**		**0.438803**	
0.053942	v	**0.000000**		**0.000000**	
	media	**0.294319**		**0.238675**	
	Desv. Est.	**0.187494**		**0.192123**	
	mejor	**-47.764888**		**-47.764888**	
	mediana	**-47.764888**		**-47.764888**	
g14	peor	**-44.664015**		**-47.622320**	
-47.764888	v	**0.000000**		**0.000000**	
	media	**-47.623774**		**-47.647472**	
	Desv. Est.	**0.231465**		**0.191210**	
	mejor	**961.715022**		**961.715022**	**961.715022**
	mediana	**961.715022**		**961.715022**	**961.715022**
g15	peor	**961.715022**		**961.715022**	**961.715022**
961.715022	v	**0.000000**		**0.000000**	**0.000000**
	media	**961.715022**		**961.715022**	**961.715022**
	Desv. Est.	**0.000000**		**0.000000**	**0.000000**
	mejor	**-1.905155**		**-1.905155**	**-1.905155**
	mediana	**-1.905155**		**-1.905155**	**-1.905155**
g16	peor	**-1.905155**		**-1.905155**	**-1.905140**
-1.905155	v	**0.000000**		**0.000000**	**0.000000**
	media	**-1.905155**		**-1.905155**	**-1.905155**
	Desv. Est.	**0.000000**		**0.000000**	**0.000000**
	mejor	**8853.533875**		**8853.533875**	**8853.533875**
	mediana	**8853.533875**		**8853.533875**	**8853.533875**
g17	peor	**8853.533875**		**8927.591751**	**8927.591747**
8853.539675	v	**0.000000**		**0.000000**	**0.000000**
	media	**8866.031621**		**8877.299910**	**8867.057150**
	Desv. Est.	**20.774853**		**32.204891**	**21.792455**
	mejor	**-0.866025**		**-0.866025**	**-0.866025**
	mediana	**-0.866025**		**-0.866025**	**-0.866025**
g18	peor	**-0.866017**		**-0.674981**	**-0.866025**
-0.866025	v	**0.000000**		**0.000000**	**0.000000**
	media	**-0.863330**		**-0.841190**	**-0.866025**
	Desv. Est.	**0.005099**		**0.043214**	**0.000000**
	mejor	**32.655593**		**32.655593**	**32.655593**
	mediana	**32.655593**		**32.655593**	**32.655593**
g19	peor	**32.655999**		**38.476427**	**32.655593**
32.655593	v	**0.000000**		**0.000000**	**0.000000**
	media	**32.655597**		**32.714055**	**32.729282**
	Desv. Est.	**0.000008**		**0.115247**	**0.145897**
	mejor	**193.724510**		**193.724510**	
	mediana	**193.724510**		**193.724511**	
g21	peor	**0.000000**		**320.113536**	
193.724510	v	**0.000000**		**0.000000**	
	media	**269.000788**		**269.050345**	
	Desv. Est.	**64.088882**		**63.271893**	
	mejor	**-400.055100**		**-400.055100**	**-400.055100**
	mediana	**-400.055099**		**-400.055099**	**-400.055098**
g23	peor	**900.000000**		**-2100.000000**	**-2100.000000**
-400.055100	v	**0.000000**		**0.000000**	**0.000000**
	media	**-439.033082**		**-358.885950**	**-550.033923**
	Desv. Est.	**365.412722**		**325.706370**	**495.989145**

Estadísticas con respecto a la solución obtenida $(\vec{x}, f(\vec{x}))$					
ALGORITMO Máximo de evaluaciones de f		EDPR 180,000	EDPR (Ajuste) 180,000	HEDSPR 180,000-216,000	HEDSPR (Ajuste) 180,000-216,000
	mejor	**-5.508013**		**-5.508013**	**-5.508013**
	mediana	**-5.508013**		**-5.508013**	**-5.508013**
g24	peor	**-5.508013**		**-5.508013**	**-5.508013**
-5.508013	v	**0.000000**		**0.000000**	**0.000000**
	media	**-5.508013**		**-5.508013**	**-5.508013**
	Desv. Est.	**0.000000**		**0.000000**	**0.000000**

Tabla 6.7: Resultados obtenidos con los algoritmos EDPR y HEDSPR sin utilizar el ajuste empírico de parámetros y utilizandolo, considerando la solución obtenida, $(\vec{x}, f(\vec{x}))$. v corresponde al valor de la función de restricciones de la solución mediana. Los resultados en negritas indican que en dicha función el algoritmo HEDSPR encontró el óptimo.

Estadísticas con respecto al número de evaluaciones de la función objetivo					
ALGORITMO Máximo de evaluaciones de f		EDPR 180,000	EDPR (Ajuste) 180,000	HEDSPR 180,000-216,000	HEDSPR (Ajuste) 180,000-216,000
	mejor	27120	**51480**	27787	**38087**
	mediana	36840	**79140**	36518	**45550**
g01	peor	47760	**167040**	49619	**53703**
-15.000000	media	36841.224490	**82812.600000**	36500.191489	**45639.740000**
	Desv. Est.	2719.616826	**16727.016000**	3282.144862	**2984.509600**
HEDSPR	TF	100.000000	**100.000000**	100.000000	**100.000000**
	TE	49.000000	**100.000000**	47.000000	**100.000000**
	Costo	75186.172428	**82812.600000**	77659.981892	**45639.740000**
	mejor	-	-	-	182971
	mediana	-	-	-	183570
g02	peor	-	-	-	183996
-0.803619	media	-	-	-	183522.130952
	Desv. Est.	-	-	-	191.686791
HEDSPR	TF	100.000000	100.000000	100.000000	100.000000
	TE	0.000000	0.000000	0.000000	84.000000
	Costo	-	-	-	218478.727324
	mejor	180000		209922	189311
	mediana	180000		211150	190171
g03	peor	180000		211970	190868
-1.000500	media	180000.000000		211109.381818	190127.045455
	Desv. Est.	0.000000		383.060496	283.179752
EDPR	TF	100.000000		81.000000	97.000000
	TE	94.000000		55.000000	88.000000
	Costo	191489.361702		383835.239669	216053.460744
	mejor	**11160**		**11376**	**9536**
	mediana	**14760**		**13689**	**10255**
g04	peor	**17880**		**16209**	**11898**
-30665.538672	media	**14772.000000**		**13640.880000**	**10379.150000**
	Desv. Est.	**903.840000**		**892.882400**	**393.540000**
HEDSPR	TF	**100.000000**		**100.000000**	**100.000000**
	TE	**100.000000**		**100.000000**	**100.000000**
	Costo	**14772.000000**		**13640.880000**	**10379.150000**
	mejor	**99120**		**120421**	**118500**
	mediana	**180000**		**210427**	**185538**
g05	peor	**180000**		**210878**	**185665**
5126.496714	media	**179191.200000**		**209224.310000**	**184207.550000**
	Desv. Est.	**1601.424000**		**2045.722000**	**2589.902000**
EDPR	TF	**100.000000**		**100.000000**	**100.000000**
	TE	**100.000000**		**100.000000**	**100.000000**
	Costo	**179191.200000**		**209224.310000**	**184207.550000**
	mejor	8580		**8633**	
	mediana	9660		**9922**	
g06	peor	12900		**11392**	
-6961.813876	media	9756.494845		**9921.790000**	
	Desv. Est.	424.623233		**465.570000**	
HEDSPR	TF	98.000000		**100.000000**	
	TE	97.000000		**100.000000**	
	Costo	10058.242109		**9921.790000**	
	mejor	**127800**		92358	
	mediana	**180000**		185644	
g07	peor	**180000**		200500	
24.306209	media	**176707.200000**		171424.659794	
	Desv. Est.	**5626.416000**		28772.022106	
EDPR	TF	**100.000000**		100.000000	
	TE	**100.000000**		97.000000	
	Costo	**176707.200000**		176726.453396	
	mejor	**3480**		**3503**	
	mediana	**3900**		**3903**	
g08	peor	**4500**		**4604**	
-0.095825	media	**3950.400000**		**3942.280000**	
	Desv. Est.	**172.608000**		**165.839200**	
HEDSPR	TF	**100.000000**		**100.000000**	
	TE	**100.000000**		**100.000000**	
	Costo	**3950.400000**		**3942.280000**	

Estadísticas con respecto al número de evaluaciones de la función objetivo					
ALGORITMO Máximo de evaluaciones de f		EDPR 180,000	EDPR (Ajuste) 180,000	HEDSPR 180,000-216,000	HEDSPR (Ajuste) 180,000-216,000
	mejor	**31680**		**27143**	
	mediana	**36960**		**32126**	
g09	peor	**48960**		**40101**	
680.630057	media	**37305.600000**		**32178.090000**	
	Desv. Est.	**1936.272000**		**1604.939000**	
HEDSPR	TF	**100.000000**		**100.000000**	
	TE	**100.000000**		**100.000000**	
	Costo	**37305.600000**		**32178.090000**	
	mejor	**180000**		203284	**187399**
	mediana	**180000**		205372	**189209**
g10	peor	**180000**		206858	**190117**
7049.248021	media	**180000.000000**		205311.343434	**189126.040000**
	Desv. Est.	**0.000000**		629.538210	**493.874400**
EDPR	TF	**100.000000**		100.000000	**100.000000**
	TE	**100.000000**		99.000000	**100.000000**
	Costo	**180000.000000**		207385.195388	**189126.040000**
	mejor	**16620**		199344	
	mediana	**20220**		199895	
g11	peor	**43740**		200974	
0.749900	media	**21384.600000**		199942.272727	
	Desv. Est.	**2509.764000**		219.461629	
EDPR	TF	**100.000000**		79.000000	
	TE	**100.000000**		77.000000	
	Costo	**21384.600000**		259665.289256	
	mejor	**9780**		**8201**	
	mediana	**12060**		**11066**	
g12	peor	**15060**		**15024**	
-1.000000	media	**12103.200000**		**10985.550000**	
	Desv. Est.	**910.464000**		**940.817000**	
HEDSPR	TF	**100.000000**		**100.000000**	
	TE	**100.000000**		**100.000000**	
	Costo	**12103.200000**		**10985.550000**	
	mejor	35400		30483	
	mediana	127260		144516	
g13	peor	180000		202274	
0.053942	media	121146.153846		137147.557692	
	Desv. Est.	28789.349112		27697.411982	
HEDSPR	TF	100.000000		100.000000	
	TE	39.000000		52.000000	
	Costo	310631.163708		263745.303254	
	mejor	105840		91781	
	mediana	180000		168467	
	peor	180000		198999	
g14	media	175647.272727		161484.557377	
-47.764888	Desv. Est.	7134.545455		29387.327600	
	TF	100.000000		100.000000	
	TE	66.000000		61.000000	
	Costo	266132.231405		264728.782585	
	mejor	**18120**		**16434**	**18054**
	mediana	**21360**		**19755**	**20686**
g15	peor	**57720**		**71744**	**53936**
961.715022	media	**23061.000000**		**23490.640000**	**21682.280000**
	Desv. Est.	**3623.220000**		**6423.162400**	**2494.608800**
HEDSPR	TF	**100.000000**		**100.000000**	**100.000000**
	TE	**100.000000**		**100.000000**	**100.000000**
	Costo	**23061.000000**		**23490.640000**	**21682.280000**
	mejor	**27660**		**25923**	**12930**
	mediana	**32340**		**28700**	**18819**
g16	peor	**35100**		**32980**	**36963**
-1.905155	media	**32266.200000**		**28869.880000**	**19939.370000**
	Desv. Est.	**1159.152000**		**1185.637600**	**4090.867000**
HEDSPR	TF	**100.000000**		**100.000000**	**100.000000**
	TE	**100.000000**		**100.000000**	**100.000000**
	Costo	**32266.200000**		**28869.880000**	**19939.370000**
	mejor	180000		195842	142931
	mediana	180000		201892	212336
g17	peor	180000		203193	216042
8853.539675	media	180000.000000		201450.873016	209541.239437
	Desv. Est.	0.000000		1033.328294	4856.478476
EDPR	TF	100.000000		100.000000	100.000000
	TE	77.000000		63.000000	71.000000
	Costo	233766.233766		319763.290501	295128.506249
	mejor	142560		64899	**117480**
	mediana	180000		83611	**188466**
g18	peor	180000		207002	**189388**
-0.866025	media	178378.666667		111289.517241	**181541.400000**
	Desv. Est.	3026.488889		44459.093143	**11743.628000**
HEDSPR	TF	99.000000		100.000000	**100.000000**
	TE	90.000000		87.000000	**100.000000**
	Costo	198198.518519		127918.985335	**181541.400000**

Estadísticas con respecto al número de evaluaciones de la función objetivo					
ALGORITMO Máximo de evaluaciones de f		EDPR 180,000	EDPR (Ajuste) 180,000	HEDSPR 180,000-216,000	HEDSPR (Ajuste) 180,000-216,000
	mejor	126660		146218	144900
	mediana	180000		201381	187102
g19	peor	180000		201973	187467
32.655593	media	177425.454545		198639.293478	182144.540816
	Desv. Est.	4159.669421		4885.911626	7206.021658
EDPR	TF	100.000000		100.000000	100.000000
	TE	99.000000		92.000000	98.000000
	Costo	179217.630854		215912.275520	185861.776343
	mejor	36720		31605	
	mediana	43020		37720	
g21	peor	54900		97315	
193.724510	media	43227.857143		40211.102564	
	Desv. Est.	3758.418367		4964.177515	
HEDSPR	TF	69.000000		86.000000	
	TE	28.000000		39.000000	
	Costo	154385.204082		103105.391190	
	mejor	89640		101597	74252
	mediana	157140		155721	138258
g23	peor	180000		203191	239403
-400.055100	media	155669.090909		160472.891304	139161.880000
	Desv. Est.	20813.663912		31267.620983	32656.275200
HEDSPR	TF	88.000000		92.000000	84.000000
	TE	33.000000		46.000000	50.000000
	Costo	471724.517906		348854.111531	278323.760000
	mejor	**5100**		**5549**	**5404**
	mediana	**6000**		**6190**	**6129**
g24	peor	**6780**		**7094**	**6981**
-5.508013	media	**6036.600000**		**6173.040000**	**6157.670000**
	Desv. Est.	**242.064000**		**234.716000**	**246.023600**
EDPR	TF	**100.000000**		**100.000000**	**100.000000**
	TE	**100.000000**		**100.000000**	**100.000000**
	Costo	**6036.600000**		**6173.040000**	**6157.670000**

Tabla 6.8: Resultados obtenidos con los algoritmos EDPR y HEDSPR sin utilizar el ajuste empírico de parámetros y utilizandolo, considerando el número de evaluaciones de la función objetivo, f, requeridas para obtener la solución $(\vec{x}, f(\vec{x}))$. Donde TF es la tasa de factibilidad y TE es la tasa de éxito. Los resultados en negritas indican que en dicha función el algoritmo HEDSPR obtuvo un porcentaje de éxito de 100 %.

En el apéndice B se muestran las gráficas de convergencia, tanto de la función objetivo como de la función de violación a las restricciones, de las veintidós funciones de prueba, correspondientes a la solución mediana de cada experimento.

Capítulo 7

Conclusiones y trabajo futuro

El manejo de restricciones es una tarea díficil, en la que las técnicas de programación matemática tienen varias limitaciones, por ejemplo, cuando la función objetivo y/o las restricciones son no diferenciables o discontinuas o no se pueden expresar de manera algebraica, o cuando la región factible es disjunta. Los algoritmos evolutivos han podido resolver funciones que tienen estas características. Sin embargo, no cuentan con un mecanismo explícito para manejo de restricciones. El problema principal de las técnicas para el manejo de restricciones en los AEs es lograr el balance correcto entre tener prioridad en minimizar el valor de la función objetivo o tener prioridad en minimizar la violación a las restricciones.

En esta tesis, se estudiaron diversos algoritmos evolutivos que nos permiten resolver problemas de optimización con restricciones concluyendo, con base en los resultados reportados, que dos de los más competitivos son el algoritmo de jerarquización estocástica propuesto en [28] y el algoritmo Diversity-DE presentado en [1]. Es importante mencionar que se estudiaron diversas propuestas de algoritmos híbridos, para resolver problemas con restricciones. Sin embargo, de acuerdo a los resultados reportados por sus autores, los algoritmos previamente propuestos no lograron superar significativamente a los algoritmos evolutivos del estado del arte.

Al estudiar diversos algoritmos evolutivos, se observó una posible deficiencia en los criterios utilizados para el manejo de restricciones. Por ejemplo, si tenemos dos posibles soluciones a un problema de minimización, $\vec{x}^{(1)}$ y $\vec{x}^{(2)}$, es evidente que:

1. Si tanto la solución $\vec{x}^{(1)}$ como la solución $\vec{x}^{(2)}$ son factibles y además $f\left(\vec{x}^{(1)}\right) \leq f\left(\vec{x}^{(2)}\right)$ entonces la solución $\vec{x}^{(1)}$ es mejor que la solución $\vec{x}^{(2)}$.

2. Si la solución $\vec{x}^{(1)}$ tiene una violación a las restricciones menor a la solución $\vec{x}^{(2)}$ y además $f\left(\vec{x}^{(1)}\right) \leq f\left(\vec{x}^{(2)}\right)$ entonces la solución $\vec{x}^{(1)}$ es mejor que la solución $\vec{x}^{(2)}$.

En la mayoría de los trabajos, siempre toman en cuenta los casos antes descritos y, en caso de que éstos no se cumplan, idean un mecanismo para balancear la búsqueda.

Sin embargo, existe otro caso en el que también existe superioridad de una solución sobre otra y que no se ha considerado en los documentos leídos para la elaboración de esta tesis. El caso es el siguiente: si tanto la solución $\vec{x}^{(1)}$ como la $\vec{x}^{(2)}$ tienen el mismo valor de violación a las restricciones y además $f\left(\vec{x}^{(1)}\right) \leq f\left(\vec{x}^{(2)}\right)$, entonces la solución $\vec{x}^{(1)}$ es mejor que la solución $\vec{x}^{(2)}$.

Por lo anterior, en esta tesis se propone un operador de selección para el algoritmo de evolución diferencial que toma en cuenta los casos antes descritos y, en caso de no cumplir alguno de ellos, se decide probabilísticamente si comparar con respecto a la función objetivo o con respecto a las restricciones. El algoritmo resultante es llamado Evolución Diferencial para Problemas con Restricciones (EDPR).

Como se mostró en el capítulo 6, la consideración antes descrita mejora considerablemente el comportamiento de la búsqueda, ya que a pesar de utilizar únicamente este criterio como estrategia de búsqueda en espacios restringidos se obtuvieron resultados competitivos con respecto a dos técnicas del estado del arte en el área (jerarquización estocástica [28] y Diversity-DE [1]).

Finalmente, teniendo como objetivo mejorar los resultados obtenidos con el algoritmo EDPR se decidió hibridizarlo con una técnica de programación matemática. Primeramente, se pensó en utilizar el método complex [43] ya que este método está basado en el método de Nelder-Mead e incluye un mecanismo para el manejo de restricciones. Sin embargo, el mecanismo para el manejo de restricciones utilizado en el método complex tiene la desventaja de que sólo se puede aplicar en zonas factibles convexas. Por lo anterior, se decició utilizar el método de Nelder-Mead e incorporar una técnica para el manejo de restricciones que fuera más general.

Posteriormente, se intentó utilizar el método de Nelder-Mead como un buscador local dentro del algoritmo EDPR. Sin embargo, no se obtuvieron buenos resultados ya que se presentaron diversos problemas, entre los que destacan la construcción del simplex inicial y la pobre escalabilidad en problemas de alta dimensionalidad. En [38], se propone un híbrido del algoritmo de evolución diferencial y el método de Nelder-Mead, donde solucionan estos problemas y obtienen buenos resultados. Sin embargo, esta propuesta sólo considera problemas de optimización sin restricciones.

A partir de la idea presentada en [38], se propuso en esta tesis el uso de un nuevo operador basado en el método de Nelder-Mead, para el algoritmo EDPR el cual ya incluye un esquema de manejo de restricciones interno, el cual da prioridad a las soluciones factibles y, en caso de tener dos soluciones con el mismo valor de violación a las restricciones, se elige con base en el valor de la función objetivo. Este operador, a diferencia del propuesto en [38], se aplica con una frecuencia determinada durante la búsqueda y a partir de cierta generación, debido a que no se tiene el mismo impacto en todas las funciones e incluso en algunas afecta de manera negativa el comportamiento de la búsqueda.

De acuerdo a los resultados presentados en el capítulo 6 podemos decir que los dos algoritmos propuestos son competitivos, con respecto a los mejores algoritmos conocidos hasta el momento llegando incluso a superarlos al requerir un menor número de evaluaciones de la función objetivo. El algoritmo EDPR realiza un máximo de 180,000 evaluaciones y el algoritmo HEDSPR realiza un máximo de 216,000 evaluaciones y se comparan con respecto a los algoritmos de jerarquización estocástica y Diversity-DE que realizan 350,000 y 225,000 evaluaciones, respectivamente.

Comparando los dos algoritmos propuestos, podemos concluir que el algoritmo HEDSPR logra mejorar los resultados obtenidos con EDPR en doce de las veintidós funciones de prueba ya sea disminuyendo el número de evaluaciones de la función objetivo o mejorando las tasas de éxito y factibiliad; sin embargo, en ocho funciones se empeoran los resultados. Este problema puede disminuir ajustando los parámetros utilizados por el operador simplex.

Al analizar por un lado las funciones beneficiadas por el nuevo operador y por otro las funciones afectadas adversamente por el mismo, no se encontró una relación con el tipo de función (lineal, cuadrática, cúbica, no lineal, polinomial), ni con el número de variables, ni con el tamaño de la región factible. Tampoco se halló relación con la diversidad existente en la población, ya que algunas veces el operador obtiene buenos resultados al aplicarse desde el inicio de la búsqueda y con una alta frecuencia e incluso sustituye totalmente al operador de mutación utilizado por el algoritmo de evolución diferencial. Y, en otros casos, obtiene buenos resultados cuando se aplica ya casi al final de la búsqueda (cuando existe poca diversidad en la población). Por lo tanto, se piensa que su comportamiento está relacionado con la forma de la zona factible y que dicho problema puede solucionarse haciendo un análisis profundo de la forma en la se eligen los puntos que conformarán el *m-simplex*. Sin embargo, esto último ya no se comprobó en esta tesis.

Finalmente, podemos concluir que los dos algoritmos propuestos son competitivos entre sí. Sin embargo, es importante mencionar que EDPR es más simple que HEDSPR y que HEDSPR no logra superar a EDPR en todas las funciones de prueba.

Como parte del trabajo futuro, se pueden mejorar aún más los algoritmos propuestos en esta tesis haciendo un estudio más profundo de su funcionamiento a fin de entender más claramente las razones por las que a veces no funciona adecuadamente el operador simplex, y así poder proponer una alternativa para mejorar el comportamiento de dicho operador. También, se podría intentar acoplar el operador simplex a otras heurísticas tales como la optimización mediante el cúmulo de partículas o las estrategias evolutivas.

Bibliografía

[1] Jesús Velázquez-Reyes Efrén Mezura-Montes and Carlos A. Coello Coello. Promising infeasibility and multiple offspring incorporated to differential evolution for constrained optimization. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 225–232, New York, NY, USA, 2005. ACM.

[2] Rao Singiresu S. *Engineering Optimization: Theory and Practice*. John Wiley and Sons, third edition, 1996.

[3] Carlos A. Coello Coello. 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*, pages 1245–1287, January 2002.

[4] George B. Dantzig. *Linear Programming and Extensions*. Princeton Landmarks in Mathematics and Physics. 1998.

[5] Richard Ernest Bellman. *Dynamic Programming*. Dover Publications, Incorporated, 2003.

[6] H. W. Kuhn and A. Tucker. Nonlinear programming. In *Proceedings of the 2nd Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, 1951. University of California Press.

[7] D. Boukari and A. V. Fiacco. Survey of penalty, exact-penalty and multiplier methods from 1968 to 1993. *Optimization*, 32:301–334, 1995.

[8] R. J. Duffin, E. L. Peterson, and C. M. Zener. Geometric programming. *Systems, Man and Cybernetics, IEEE Transactions on*, 4:593–593, 1974.

[9] R. T. Marler and J. S. Arora. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26(6):369–395, 2004.

[10] Drew Fudenberg and Jean Tirole. *Game Theory*. The MIT Press, 1991.

[11] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983.

[12] John H. Holland. Outline for a Logical Theory of Adaptive Systems. *J. ACM*, 9(3):297–314, 1962.

[13] Wing-Kay Kan and Igor Aleksander. A probabilistic logic neuron network for associative learning. In *Neural computing architectures: the design of brain-like machines*, pages 156–171. MIT Press, Cambridge, MA, USA, 1989.

[14] Rainer Storn and Kenneth Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, Berkeley, CA, 1995.

[15] James Kennedy and Russell C. Eberhart. The particle swarm: social adaptation in information-processing systems. In *New ideas in optimization*, pages 379–388. McGraw-Hill Ltd., UK, Maidenhead, UK, England, 1999.

[16] Thomas P. Runarsson and Xin Yao. Search biases in constrained evolutionary optimization. *IEEE Transactions on Systems, Man and Cybernetics 35*, 2:233–243, May 2005.

[17] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7:308–313, 1965.

[18] Pizzo J. T. Jacoby S. L. S., Kowalik J. S. *Iterative Methods for Nonlinear Optimization Problems*. Prentice-Hall, 1972.

[19] G. R. Hext W. Spendley and F. R. Himsworth. Sequencial application of simplex designs in optimization and evolutionary operation. *Technometrics*, 4(4):441 – 461, November 1962.

[20] W. D. Cannon. *Tne Wisdom of the Body*. Norton and Company, New York, 1932.

[21] Alan Mathison Turing. Computing machinery and intelligence. *Mind*, 59:94–101, 1950.

[22] Jesús Velázquez Reyes. Propuesta de evolución diferencial para optimización de espacios restringidos. Master's thesis, CINVESTAV-IPN, México, D. F., Enero 2006.

[23] Antonin Ponsich, Catherine Azzaro-Pantel, Serge Domenech, and Lue Pibouleau. Some guidelines for genetic algorithm implementation in minlp batch plant design problems. In Patrick Siarry and Zbigniew Michalewicz, editors, *Advances in Metaheuristic Methods for Hard Optimization*, pages 293–315. Springer, Berlin, 2008. ISBN 978-3-540-72959-4.

[24] Zbigniew Michalewicz. 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*, pages 135–155. The MIT Press, Cambridge, Massachusetts, 1995.

[25] S. H. Y. Lai A. Homaifar and X. Qi. Constrained optimization via genetic algorithms. *Simulation*, 62(4):242–254, 1994.

[26] J. Joines and C. Houck. 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*, pages 579–584, Orlando, Florida, 1994. IEEE Press.

[27] A. E Smith D. W. Coit and D. M. Tate. Adaptive penalty methods for genetic optimization of constrained combinatorial problems. *INFORMS Journal on Computing 8*, 2:173–182, June 1996.

[28] Thomas P. Runarsson and Xin Yao. Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):284–294, September 2000.

[29] Bo Liu, Hannan Ma, and Xuejun Zhang. A Co-evolutionary Differential Evolution Algorithm for Constrained Optimization. In *ICNC '07: Proceedings of the Third International Conference on Natural Computation (ICNC 2007)*, pages 51–57, Washington, DC, USA, 2007. IEEE Computer Society.

[30] Rainer Storn and Kenneth Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *J. of Global Optimization*, 11(4):341–359, 1997.

[31] Z. Michalewicz and M. Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary computation*, 4(1):1–32, 1996.

[32] Z. Michalewicz. Evolutionary computation techniques for nonlinear programming problems. *International Transactions in Operational Research*, 1(2):223–240, 1994.

[33] J. Paredis. Genetic State-Space Search for Constraint Optimization Problems. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, volume 112, pages 967–997, 1993.

[34] J. Paredis. Co-evolutionary Constraint Satisfaction. In *Proceedings of the 3rd Conference on Parallel Problem Solving from Nature*, pages 46–55, New York, 1994. Springer Verlag.

[35] Kalyanmoy Deb. An Efficient Constraint Handling Method for Genetic Algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2/4):311–338, 2000.

[36] Rachid Chelouah and Patrick Siarry. Genetic and nelder–mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions. *European Journal of Operational Research*, 148(Issue 2):335–348, july 2003.

[37] Fang Wang and Yuhui Qiu. Empirical Study of Hybrid Particle Swarm Optimizers with the Simplex Method Operator. In *ISDA '05: Proceedings of the 5th International Conference on Intelligent Systems Design and Applications*, pages 308–313, Washington, DC, USA, 2005. IEEE Computer Society.

[38] Changtong Luo and Bo Yu. Low Dimensional Simplex Evolution–A Hybrid Heuristic for Global Optimization. In *SNPD '07: Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007)*, pages 470–474, Washington, DC, USA, 2007. IEEE Computer Society.

[39] Wen-Jun Zhang and Xiao-Feng Xie. Depso: hybrid particle swarm with differential evolution operator. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 3816–3821, Inst. of Microelectron., Tsinghua Univ., Beijing, China, October 2003. IEEE Computer Society.

[40] Yung-Chien Lin, Kao-Shing Hwang, and Feng-Sheng Wang. Hybrid Differential Evolution with Multiplier Updating Method for Nonlinear Constrained Optimization Problems. In *Proceedings of the Congress on Evolutionary Computation 2002 (CEC'2002)*, volume 1, pages 872–877, Piscataway, New Jersey, May 2002. IEEE Service Center.

[41] Sheng Liu, Xingyu Wang, and Xiaoming You. Cultured Differential Particle Swarm Optimization for Numerical Optimization Problems. In *ICNC '07: Proceedings of the Third International Conference on Natural Computation (ICNC 2007)*, pages 642–648, Washington, DC, USA, 2007. IEEE Computer Society.

[42] M. J. D. Powell. Algorithms for nonlinear constraints that use lagrangian functions. *Mathematical Programming*, 4:224–248, 1978.

[43] M. J. Box. A new method of constrained optimization and a comparison with other methods. *Computer Journal*, 8:42–52, 1965.

[44] G. E. P. Box and N. R. Draper. *Evolutionary operation. A statistical method for process improvement.* John Wiley and Sons, 1969.

[45] Efrén Mezura-Montes Maurice Clerc P. N. Suganthan Carlos A. Coello Coello K. Deb J. J. Liang, Thomas Philip Runarsson. Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization. Technical report, September 2006.

[46] Carlos. A. Coello Coello Efrén Mezura-Montes and Edy I. Tun-Morales. Simple feasibility rules and differential evolution for constrained optimization. In Luis Enrique Sucar Raul Monroy, Gustavo Arroyo-Figueroa and Juan Humberto Sossa Azuela, editors, *In Proceedings of the 3rd Mexican International Conference on Aritficial Intelligence (MICAI'2004)*, volume 2972 of *Lecture Notes in Computer Science*, pages 707–716. Springer, 2004.

Apéndice A

Funciones de prueba

g01

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

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

Los límites de las variables son: $0 \leq x_i \leq 1$ $(i = 1, ..., 9)$, $0 \leq x_i \leq 100$ $(i = 10, 11, 12)$ y $0 \leq x_{13} \leq 1$. 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 son activas.

g02

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

$$
\begin{aligned}
g_1(\vec{x}) &= 0.75 - \prod_{i=1}^{n} x_i \leq 0 \\
g_2(\vec{x}) &= \sum_{i=1}^{n} x_i - 7.5n \leq 0
\end{aligned}
$$

Donde $n = 20$ y $0 \leq x_i \leq 10$ $(i = 1, ..., n)$. El óptimo global es desconocido; la mejor solución reportada 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 $f(\vec{x}^*) = 0.80361910412559$. La restricción g_1 es casi activa ($g_1 = -10^{-8}$).

g03

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

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

Donde $n = 10$ y $0 \leq x_i \leq 1$ $(i = 1, ..., n)$. El óptimo global esta en $x_i^* = \frac{1}{\sqrt{n}}$ $(i = 1, ..., n)$ donde $f(\vec{x}^*) = 1$.

g04

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

$$\begin{aligned}
g_1(\vec{x}) &= 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0 \\
g_2(\vec{x}) &= -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leq 0 \\
g_3(\vec{x}) &= 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leq 0 \\
g_4(\vec{x}) &= -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leq 0 \\
g_5(\vec{x}) &= 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0 \\
g_6(\vec{x}) &= -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0
\end{aligned}$$

Donde $78 \leq x_1 \leq 102$, $33 \leq x_2 \leq 45$, $27 \leq x_i \leq 45$ $(i = 3, 4, 5)$. El óptimo global es $\vec{x}^* = (78, 33, 29.995256025682, 45, 36.775812905788)$ donde $f(\mathbf{x}^*) = -30665.539$. Las restricciones g_1 y g_6 son activas.

g05

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

$$\begin{aligned}
g_1(\vec{x}) &= -x_4 + x_3 - 0.55 \leq 0 \\
g_2(\vec{x}) &= -x_3 + x_4 - 0.55 \leq 0 \\
h_3(\vec{x}) &= 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0 \\
h_4(\vec{x}) &= 1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0 \\
h_5(\vec{x}) &= 1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0
\end{aligned}$$

Donde $0 \leq x_1 \leq 1200$, $0 \leq x_2 \leq 1200$, $-0.55 \leq x_3 \leq 0.55$, y $-0.55 \leq x_4 \leq 0.55$. La mejor solución conocida es $\vec{x}^* = (679.9453, 1026.067, 0.1188764, -0.3962336)$ donde $f(\vec{x}^*) = 5126.4981$.

g06

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

$$g_1(\vec{x}) = -(x_1-5)^2 - (x_2-5)^2 + 100 \leq 0$$
$$g_2(\vec{x}) = (x_1-6)^2 + (x_2-5)^2 - 82.81 \leq 0$$

Donde $13 \leq x_1 \leq 100$ y $0 \leq x_2 \leq 100$. El óptimo es $\vec{x}^* = (14.095, 0.84296)$ donde$f(\vec{x}^*) = -6961.81388$. Ambas restricciones son activas.

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:

$$g_1(\vec{x}) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0$$
$$g_2(\vec{x}) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0$$
$$g_3(\vec{x}) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0$$
$$g_4(\vec{x}) = 3(x_1-2)^2 + 4(x_2-3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0$$
$$g_5(\vec{x}) = 5x_1^2 + 8x_2 + (x_3-6)^2 - 2x_4 - 40 \leq 0$$
$$g_6(\vec{x}) = x_1^2 + 2(x_2-2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leq 0$$
$$g_7(\vec{x}) = 0.5(x_1-8)^2 + 2(x_2-4)^2 + 3x_5^2 - x_6 - 30 \leq 0$$
$$g_8(\vec{x}) = -3x_1 + 6x_2 + 12(x_9-8)^2 - 7x^{10} \leq 0$$

Donde $-10 \leq x_i \leq 10$ $(i = 1, ..., 10)$. El óptimo global es $\vec{x}^* = (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927)$ donde $f(\vec{x}^*) = 24.3062091$. Las restricciones g_1, g_2, g_3, g_4, g_5 y g_6 son activas.

g08

Maximizar $f(\vec{x}) = \frac{\sin 3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1+x_2)}$ sujeto a:

$$g_1(\vec{x}) = x_1^2 - x_2 + 1 \leq 0$$
$$g_2(\vec{x}) = 1 - x_1 + (x_2-4)^2 \leq 0$$

Donde $0 \leq x_1 \leq 10$ y $0 \leq x_2 \leq 10$. La solución óptima es $\vec{x}^* = (1.2279713, 4.2453733)$ donde $f(\vec{x}^*) = 0.095825$.

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:

$$g_1(\vec{x}) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0$$
$$g_2(\vec{x}) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0$$
$$g_3(\vec{x}) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 0$$
$$g_4(\vec{x}) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0$$

Donde $-10 \leq x_i \leq 10$ $(i = 1, ..., 7)$. El óptimo global es $\vec{x}^* = (2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227)$ donde $f(\vec{x}^*) = 680.6300573$. Dos restricciones son activas (g_1 y g_4).

g10
Minimizar $f(\vec{x}) = x_1 + x_2 + x_3$ sujeto a:

$$g_1(\vec{x}) = -1 + 0.0025(x_4 + x_6) \leq 0$$
$$g_2(\vec{x}) = -1 + 0.0025(x_5 + x_7 - x_4) \leq 0$$
$$g_3(\vec{x}) = -1 + 0.01(x_8 - x_5) \leq 0$$
$$g_4(\vec{x}) = -x_1x_6 + 833.33252x_4 + 100x_1 - 83333.333 \leq 0$$
$$g_5(\vec{x}) = -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \leq 0$$
$$g_6(\vec{x}) = -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \leq 0$$

Donde $100 \leq x_1 \leq 10000$, $1000 \leq x_i \leq 10000$ $(i = 2, 3)$, $10 \leq x_i \leq 1000$ $(i = 4, ..., 8)$. El óptimo global es $\vec{x}^* = (579.19, 1360.13, 5109.92, 182.0174, 295.5985, 217.9799, 286.40, 395.5979)$, donde $f(\vec{x}^*) = 7049.248$. Las restricciones g_1, g_2 y g_3 son activas.

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 $-1 \leq x_1 \leq 1$, $-1 \leq x_2 \leq 1$. La solución óptima es $\vec{x}^* = (\pm 1/\sqrt{2}, 1/2)$ donde $f(\mathbf{x}^*) = 0.7499$.

g12
Minimizar $f(\vec{x}^*) = -(100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2)/100$ sujeto a:

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

Donde $0 \leq x_i \leq 10$ $(i = 1, 2, 3)$ y $p, q, r = 1, 2, ..., 9$. La región factible consiste de 9^3 esferas disjuntas. Un punto (x_1, x_2, x_3) es factible si y sólo si existen p, q, r tal que la desigualdad se cumple. El óptimo es localizado en $\vec{x}^* = (5, 5, 5)$ donde $f(\vec{x}^*) - 1$.

g13
Minimizar $f(\vec{x}) = e^{x_1x_2x_3x_4x_5}$ sujeto a:

$$h_1(\vec{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0$$
$$h_2(\vec{x}) = x_2x_3 - 5x_4x_5 = 0$$
$$h_3(\vec{x}) = x_1^3 + x_2^3 + 1 = 0$$

Donde $-2.3 \leq x_i \leq 2.3$ $(i = 1, 2)$ y $-3.2 \leq x_i \leq 3.2$ $(i = 3, 4, 5)$. La solución óptima es $\vec{x}^* = (-1.717143, 1.595709, 1.827247, -0.7636413, -0.763645)$ donde $f(\vec{x}^*) = 0.0539498$.

g14
Minimizar $f(x) = \sum_{i=1}^{10} x_i \left(c_i + \ln \frac{x_i}{\sum_{j=1}^{10} x_j} \right)$ 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 son $0 \leq xi \leq 10$ $(i = 1, ..., 10)$, y $c1 = -6.089$, $c2 = -17.164$, $c3 = -34.054$, $c4 = -5.914$, $c5 = -24.721$, $c6 = -14.986$, $c7 = -24.1$, $c8 = -10.708$, $c9 = -26.662$, $c10 = -22.179$. La mejor solución conocida está en $\vec{x}^* =$ (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 $f(\vec{x}) = 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x$ sujeto a:

$h_1(\vec{x}) = x_1^2 + x_2^2 + x_3^2 - 25 = 0$
$h2(\vec{x}) = 8x_1 + 14x_2 + 7x_3 - 56 = 0$

Donde los límites son $0 \leq x_i \leq 10$ $(i = 1, 2, 3)$. La mejor solución conocida está en $\vec{x}^* = (3.51212812611795133, 0.216987510429556135, 3.55217854929179921)$ 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:

$g_1(\vec{x}) = \frac{0.28}{0.72}y_5 - y_4 \leq 0$
$g_2(\vec{x}) = x_3 - 1.5x_2 \leq 0$
$g_3(\vec{x}) = 3496\frac{y_2}{c_{12}} - 21 \leq 0$
$g_4(\vec{x}) = 110.6 + y_1 - \frac{62212}{c_{17}} \leq 0$
$g_5(\vec{x}) = 213.1 - y_1 \leq 0$
$g_6(\vec{x}) = y_1 - 405.23 \leq 0$
$g_7(\vec{x}) = 17.505 - y_2 \leq 0$
$g_8(\vec{x}) = y_2 - 1053.6667 \leq 0$
$g_9(\vec{x}) = 11.275 - y_3 \leq 0$
$g_{10}(\vec{x}) = y_3 - 35.03 \leq 0$
$g_{11}(\vec{x}) = 214.228 - y_4 \leq 0$
$g_{12}(\vec{x}) = y_4 - 665.585 \leq 0$
$g_{13}(\vec{x}) = 7.458 - y_5 \leq 0$
$g_{14}(\vec{x}) = y_5 - 584.463 \leq 0$
$g_{15}(\vec{x}) = 0.961 - y_6 \leq 0$
$g_{16}(\vec{x}) = y_6 - 265.916 \leq 0$
$g_{17}(\vec{x}) = 1.612 - y_7 \leq 0$
$g_{18}(\vec{x}) = y_7 - 7.046 \leq 0$
$g_{19}(\vec{x}) = 0.146 - y_8 \leq 0$

$g_{20}(\vec{x}) = y_8 - 0.222 \leq 0$
$g_{21}(\vec{x}) = 107.99 - y_9 \leq 0$
$g_{22}(\vec{x}) = y_9 - 273.366 \leq 0$
$g_{23}(\vec{x}) = 922.693 - y_{10} \leq 0$
$g_{24}(\vec{x}) = y_{10} - 1286.105 \leq 0$
$g_{25}(\vec{x}) = 926.832 - y_{11} \leq 0$
$g_{26}(\vec{x}) = y_{11} - 1444.046 \leq 0$
$g_{27}(\vec{x}) = 18.766 - y_{12} \leq 0$
$g_{28}(\vec{x}) = y_{12} - 537.141 \leq 0$
$g_{29}(\vec{x}) = 1072.163 - y_{13} \leq 0$
$g_{30}(\vec{x}) = y_{13} - 3247.039 \leq 0$
$g_{31}(\vec{x}) = 8961.448 - y_{14} \leq 0$
$g_{32}(\vec{x}) = y_{14} - 26844.086 \leq 0$
$g_{33}(\vec{x}) = 0.063 - y_{15} \leq 0$
$g_{34}(\vec{x}) = y_{15} - 0.386 \leq 0$
$g_{35}(\vec{x}) = 71084.33 - y_{16} \leq 0$
$g_{36}(\vec{x}) = -140000 + y_{16} \leq 0$
$g_{37}(\vec{x}) = 2802713 - y_{17} \leq 0$
$g_{38}(\vec{x}) = y_{17} - 12146108 \leq 0$

Donde:

$y_1 = x_2 + x_3 + 41.6$
$c_1 = 0.024x_4 - 4.62$
$y_2 = \frac{12.5}{c_1} + 12$
$c_2 = 0.0003535x_1^2 + 0.5311x_1 + 0.08705y_2x_1$
$c_3 = 0.052x_1 + 78 + 0.002377y_2x_1$
$y_3 = \frac{c_2}{c_3}$
$y_4 = 19y_3$
$c_4 = 0.04782(x_1 - y_3) + \frac{0.1956(x_1 - y_3)^2}{x_2} + 0.6376y_4 + 1.594y_3$
$c_5 = 100x_2$
$c_6 = x_1 - y_3 - y_4$
$c_7 = 0.950 - \frac{c_4}{c_5}$
$y_5 = c_6c_7$
$y_6 = x_1 - y_5 - y_4 - y_3$
$c_8 = (y_5 + y_4)0.995$
$y_7 = \frac{c_8}{y_1}$
$y_8 = \frac{c_8}{3798}$
$c_9 = y_7 - \frac{0.0663y_7}{y_8} - 0.3153$
$y_9 = \frac{96.82}{c_9} + 0.321y_1$
$y_{10} = 1.29y_5 + 1.258y_4 + 2.29y_3 + 1.71y_6$
$y_{11} = 1.71x_1 - 0.452y_4 + 0.580y_3$
$c_{10} = \frac{12.3}{752.3}$

$c_{11} = (1.75y_2)(0.995x_1)$
$c_{12} = 0.995y_{10} + 1998$
$y_{12} = c_{10}x_1 + \frac{c_{11}}{c_{12}}$
$y_{13} = c_{12} - 1.75y_2$
$y_{14} = 3623 + 64.4x_2 + 58.4x_3 + \frac{146312}{y_9+x_5}$
$c_{13} = 0.995y_{10} + 60.8x_2 + 48x_4 - 0.1121y_{14} - 5095$
$y_{15} = \frac{y_{13}}{c_{13}}$
$y_{16} = 148000 - 331000y_{15} + 40y_{13} - 61y_{15}y_{13}$
$c_{14} = 2324y_{10} - 28740000y_2$
$y_{17} = 14130000 - 1328y_{10} - 531y_{11} + \frac{c_{14}}{c_{12}}$
$c_{15} = \frac{y_{13}}{y_{15}} - \frac{y_{13}}{0.52}$
$c_{16} = 1.104 - 0.72y_{15}$
$c_{17} = y_9 + x_5$

Y donde los límites son $704.4148 \leq x_1 \leq 906.3855$, $68.6 \leq x_2 \leq 288.88$, $0 \leq x_3 \leq 134.75$, $193 \leq x_4 \leq 287.0966$ y $25 \leq x5 \leq 84.1988$. La mejor solución conocida está en $\vec{x}^* = (705.174537070090537, 68.5999999999999943, 102.899999999999991, 282.324931593660324, 37.5841164258054832)$ donde $f(\vec{x}^*) = -1.90515525853479$.

g17
Minimizar $f(\vec{x}) = |f(x_1) + f(x_2)|$ donde:

$$f_1(x_1) = \begin{cases} 30x_1 & 0 \leq x_1 < 300 \\ 31x_1 & 300 \leq x_1 < 400 \end{cases}$$

$$f_2(x_2) = \begin{cases} 28x_2 & 0 \leq x_2 < 100 \\ 29x_2 & 100 \leq x_2 < 200 \\ 30x_2 & 200 \leq x_2 < 1000 \end{cases}$$

Sujeto a:

$h_1(\vec{x}) = -x_1 + 300 - \frac{x_3x_4}{131.078}\cos(1.48477 - x_6) + \frac{0.90798x_3^2}{131.078}\cos(1.47588)$
$h_2(\vec{x}) = -x_2 - \frac{x_3x_4}{131.078}\cos(1.48477 + x_6) + \frac{0.90798x_4^2}{131.078}\cos(1.47588)$
$h_3(\vec{x}) = -x_5 - \frac{x_3x_4}{131.078}\sin(1.48477 + x_6) + \frac{0.90798x_4^2}{131.078}\sin(1.47588)$
$h_4(\vec{x}) = 200 - \frac{x_3x_4}{131.078}\sin(1.48477 - x_6) + \frac{0.90798x_3^2}{131.078}\sin(1.47588)$

Donde los límites son $0 \leq x_1 \leq 400$, $0 \leq x_2 \leq 1000$, $340 \leq x_3 \leq 420$, $340 \leq x_4 \leq 420$, $-1000 \leq x_5 \leq 1000$ y $0 \leq x_6 \leq 0.5236$. La mejor solución conocida está en $\vec{x}^* = (201.784467214523659, 99.9999999999999005, 383.071034852773266, 420, -10.9076584514292652, 0.0731482312084287128)$ donde $f(\vec{x}^*) = 8853.53967480648$.

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:

$g_1(\vec{x}) = x_3^2 + x_4^2 - 1 \leq 0$
$g_2(\vec{x}) = x_9^2 - 1 \leq 0$
$g_3(\vec{x}) = x_5^2 + x_6^2 - 1 \leq 0$
$g_4(\vec{x}) = x_1^2 + (x_2 - x_9)^2 - 1 \leq 0$
$g_5(\vec{x}) = (x_1 - x_5)^2 + (x_2 - x_6)^2 - 1 \leq 0$
$g_6(\vec{x}) = (x_1 - x_7)^2 + (x_2 - x_8)^2 - 1 \leq 0$
$g_7(\vec{x}) = (x_3 - x_5)^2 + (x_4 - x_6)^2 - 1 \leq 0$
$g_8(\vec{x}) = (x_3 - x_7)^2 + (x_4 - x_8)^2 - 1 \leq 0$
$g_9(\vec{x}) = x_7^2 + (x_8 - x_9)^2 - 1 \leq 0$
$g_{10}(\vec{x}) = x_2 x_3 - x_1 x_4 \leq 0$
$g_{11}(\vec{x}) = -x_3 x_9 \leq 0$
$g_{12}(\vec{x}) = x_5 x_9 \leq 0$
$g_{13}(\vec{x}) = x_6 x_7 - x_5 x_8 \leq 0$

Donde los límites son $-10 \leq xi \leq 10$ $(i = 1, ..., 8)$ y $0 \leq x9 \leq 20$. La mejor solución conocida es $\vec{x}^* = (-0.657776192427943163, -0.153418773482438542, 0.323413871675240938, -0.946257611651304398, -0.657776194376798906, -0.753213434632691414, 0.323413874123576972, -0.346462947962331735, 0.59979466285217542)$ donde $f(\vec{x}^*) = -0.866025403784439$.

g19

Minimizar $f(\vec{x}) = \sum_{j=1}^{5}\sum_{i=1}^{5} c_{ij} x_{(10+i)} x_{(10+j)} + 2\sum_{j=1}^{5} d_j x_{(10+j)}^3 - \sum_{i=1}^{10} b_i x_i$ sujeto a:

$$g_j(\vec{x}) = -2\sum_{i=1}^{5} c_{ij} x_{(10+i)} - 3 d_j x_{(10+j)}^2 - e_j + \sum_{i=1}^{10} a_{ij} x_i \leq 0 \qquad j = 1, ..., 5$$

Donde $\vec{b} = [-40, -2, -0.25, -4, -4, -1, -40, -60, 5, 1]$ y los datos restantes se muestran en la tabla A.1. Los límites son $0 \leq x_i \leq 10$ $(i = 1, ..., 15)$. La mejor solución conocida está en $\vec{x}^* = (1.66991341326291344e-17, 3.95378229282456509e-16, 3.94599045143233784, 1.06036597479721211e-16, 3.2831773458454161, 9.99999999999999822, 1.12829414671605333e-17, 1.2026194599794709e-17, 2.50706276000769697e-15, 2.24624122987970677e-15, 0.370764847417013987, 0.278456024942955571, 0.523838487672241171, 0.388620152510322781, 0.298156764974678579)$ donde $f(\vec{x}^*) = 32.6555929502463$.

g21

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

$g_1(\vec{x}) = -x_1 + 35x_2^{0.6} + 35x_3^{0.6} \leq 0$
$h_1(\vec{x}) = -300x_3 + 7500x_5 - 7500x_6 - 25x_4x_5 + 25x_4x_6 + x_3x_4 = 0$
$h_2(\vec{x}) = 100x_2 + 155.365x_4 + 2500x_7 - x_2x_4 - 25x_4x_7 - 15536.5 = 0$
$h_3(\vec{x}) = -x_5 + \ln(-x_4 + 900) = 0$
$h_4(\vec{x}) = -x_6 + \ln(x_4 + 300) = 0$

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 $g19$

$h_5(\vec{x}) = -x_7 + \ln(-2x_4 + 700) = 0$

Donde los límites son $0 \leq x_1 \leq 1000$, $0 \leq x_2, x_3 \leq 40$, $100 \leq x_4 \leq 300$, $6.3 \leq x_5 \leq 6.7$, $5.9 \leq x_6 \leq 6.4$ y $4.5 \leq x_7 \leq 6.25$. La mejor solución conocida está en $\vec{x}^* = (193.7245100700349 67, 5.56944131553368433e - 27, 17.3191887294084914, 100.047897801386839, 6.68445185362377892, 5.99168428444264833, 6.21451648886070451)$ donde $f(\vec{x}^*) = 193.724510070035$.

g23
Minimizar $f(\vec{x}) = -9x_5 - 15x_8 + 6x_1 + 16x_2 + 10(x_6 + x_7)$ sujeto a:

$g_1(\vec{x}) = x_9x_3 + 0.02x_6 - 0.025x_5 \leq 0$
$g_2(\vec{x}) = x_9x_4 + 0.02x_7 - 0.015x_8 \leq 0$
$h_1(\vec{x}) = x_1 + x_2 - x_3 - x_4 = 0$
$h_2(\vec{x}) = 0.03x_1 + 0.01x_2 - x_9(x_3 + x_4) = 0$
$h_3(\vec{x}) = x_3 + x_6 - x_5 = 0$
$h_4(\vec{x}) = x_4 + x_7 - x_8 = 0$

Donde los límites son $0 \leq x_1, x_2, x_6 \leq 300$, $0 \leq x_3, x_5, x_7 \leq 100$, $0 \leq x_4, x_8 \leq 200$ y $0.01 \leq x_9 \leq 0.03$. La mejor solución conocida está en $\vec{x}^* = (0.005100000000000259465, 99.9947000000000514, 9.01920162996045897e - 18, 99.9999000000000535, 0.000100000000027086086, 2.75700683389584542e - 14, 99.9999999999999574, 2000.0100000100000100008)$ donde $f(\vec{x}^*) = -400.055099999999584$.

g24
Minimizar $f(\vec{x}) = -x_1 - x_2$ sujeto a:

$g_1(\vec{x}) = -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 2 \leq 0$
$g_2(\vec{x}) = -4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x_2 - 36 \leq 0$

Donde los límites son $0 \leq x_1 \leq 3$ y $0 \leq x_2 \leq 4$. El mínimo global factible está en $\vec{x}^* = (2.32952019747762, 3.17849307411774)$ donde $f(\vec{x}^*) = -5.50801327159536$. Este problema tiene una región factible que consiste de dos sub-regiones desconectadas.

Apéndice B

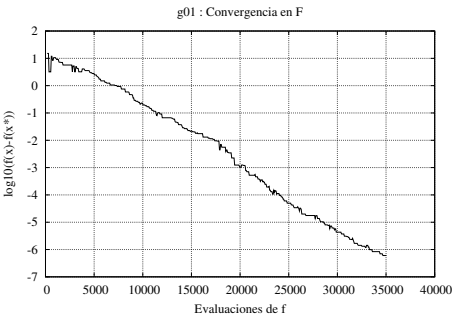
Gráficas de convergencia

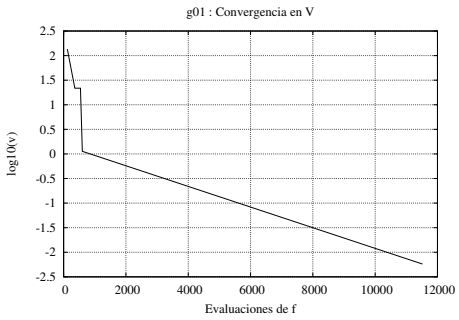
A continuación se muestran las gráficas de convergencia correspondientes a los experimentos realizados sin utilizar el ajuste de parámetros empírico. Recordemos que únicamente se presentan las gráficas correspondientes a la solución mediana.

En el lado izquierdo se muestran las gráficas de convergencia de la función objetivo, f. En el lado derecho se muestran las gráficas de convergencia de la función de violación a las restricciones, v. Cuando $f(x) - f(x*) \leq 0$ o $v \leq 0$ no se gráfica.

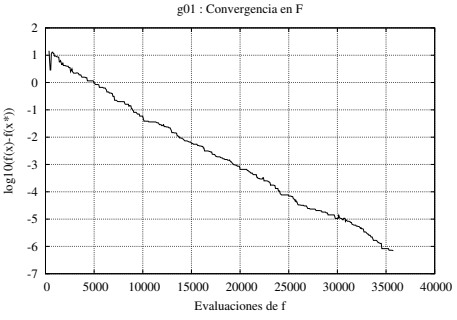
Gráficas de convergencia: Función $g01$

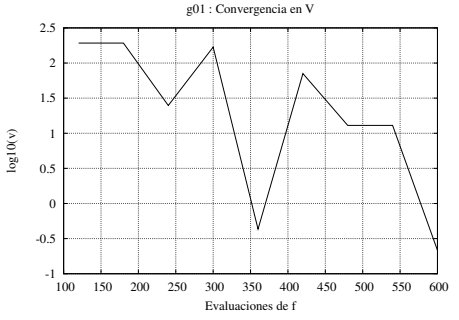
EDPR (MEF = 180,000)





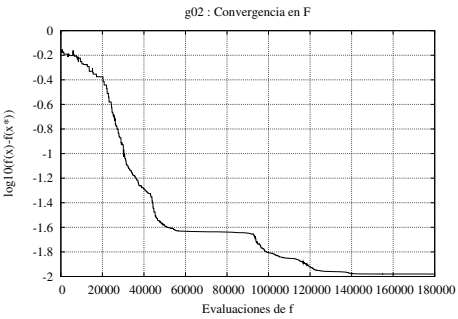
HEDSPR (MEF = 216,000)

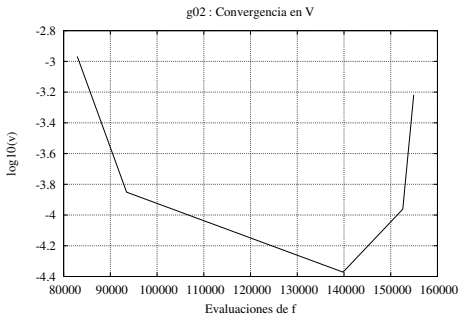




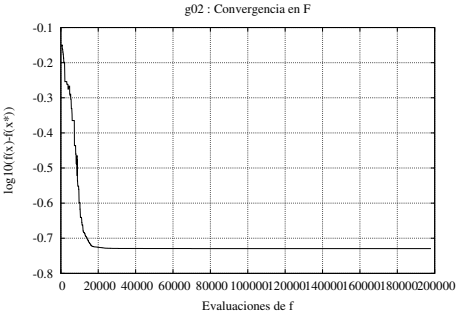
Gráficas de convergencia: Función $g02$

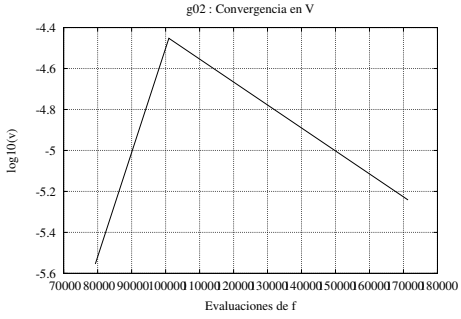
EDPR (MEF = 180,000)





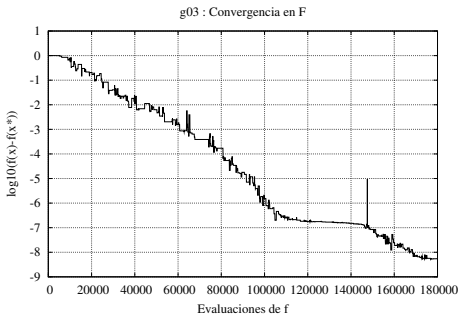
HEDSPR (MEF = 216,000)

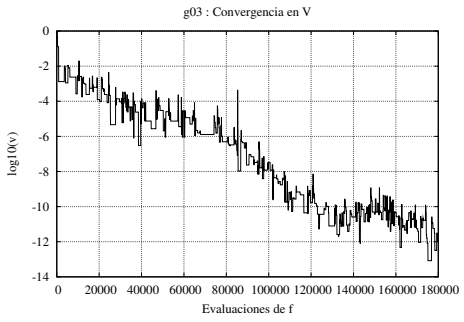




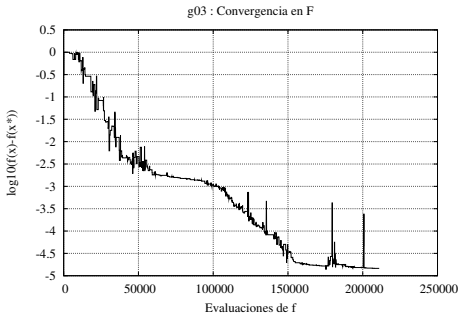
Gráficas de convergencia: Función $g03$

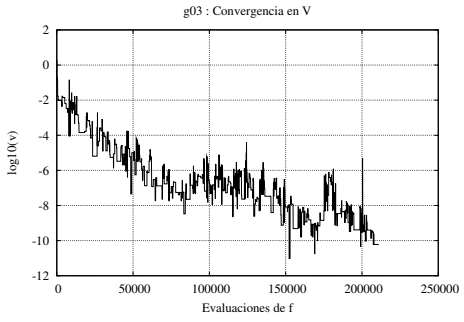
EDPR (MEF = 180,000)





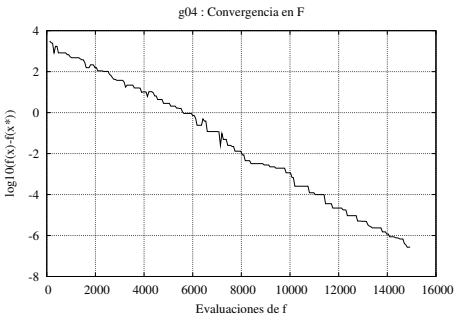
HEDSPR (MEF = 216,000)



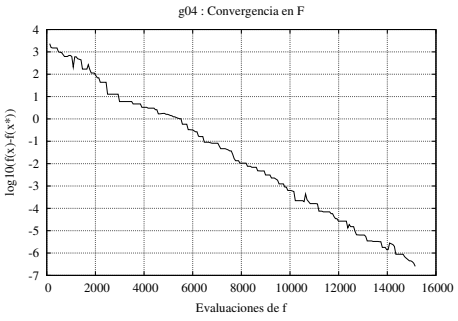


Gráficas de convergencia: Función $g04$

EDPR (MEF = 180,000)

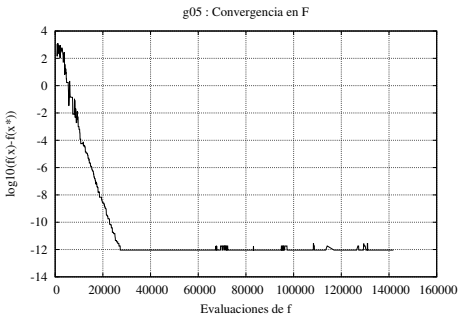


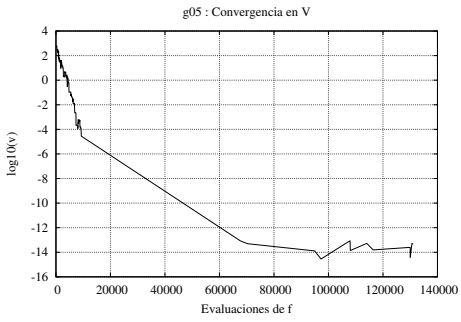
HEDSPR (MEF = 216,000)



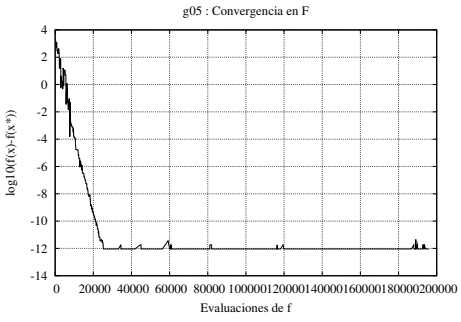
Gráficas de convergencia: Función $g05$

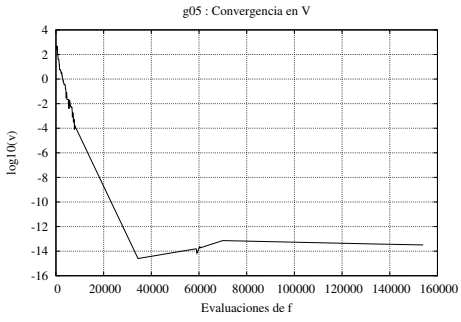
EDPR (MEF = 180,000)





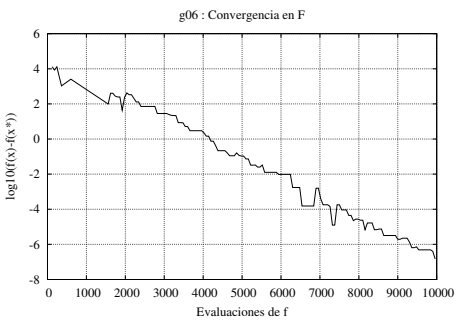
HEDSPR (MEF = 216,000)

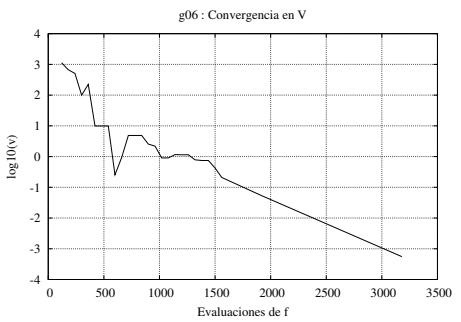




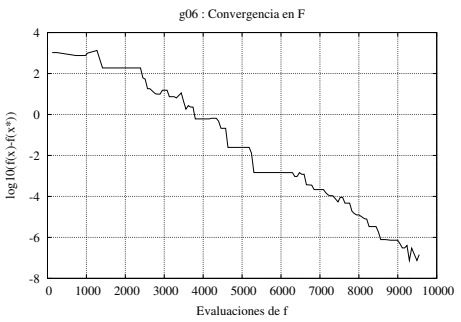
Gráficas de convergencia: Función $g06$

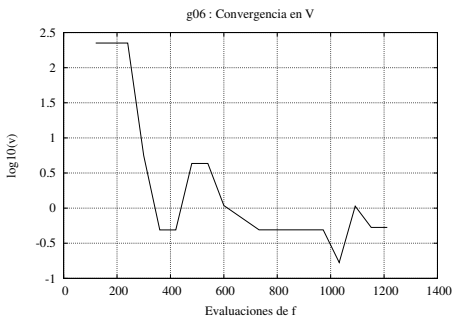
EDPR (MEF = 180,000)





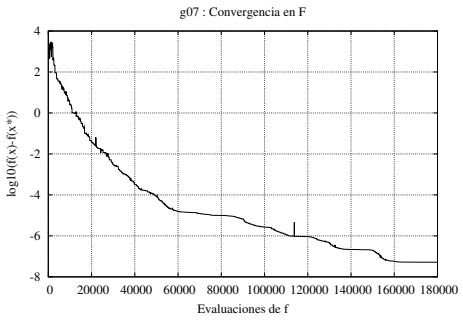
HEDSPR (MEF = 216,000)

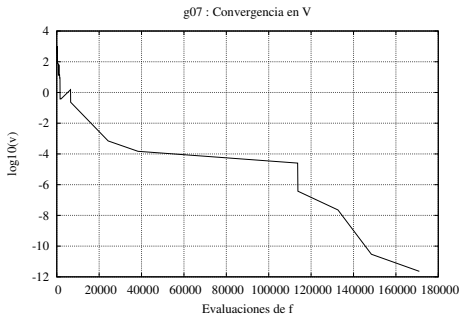




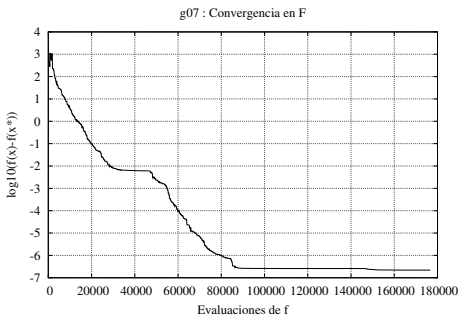
Gráficas de convergencia: Función $g07$

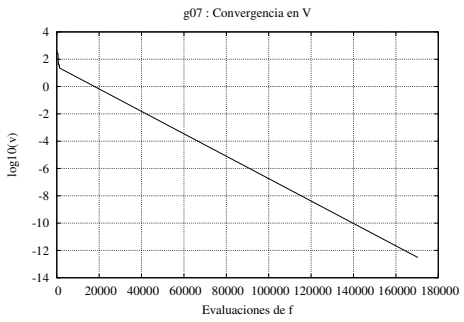
EDPR (MEF = 180,000)





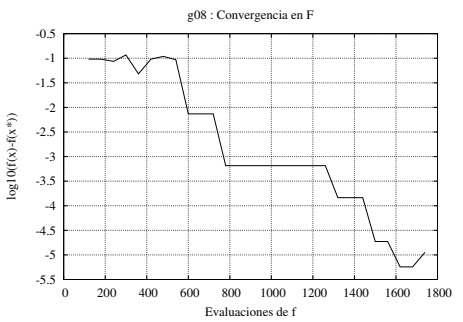
HEDSPR (MEF = 216,000)

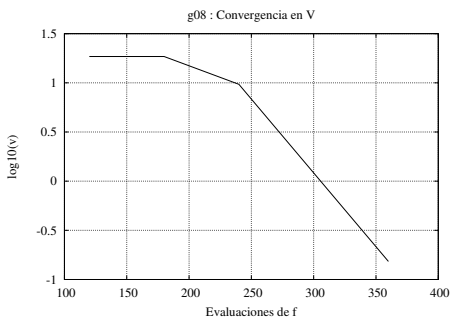




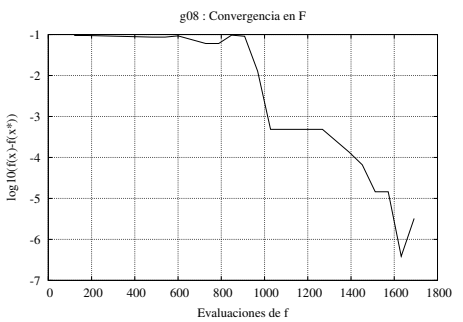
Gráficas de convergencia: Función $g08$

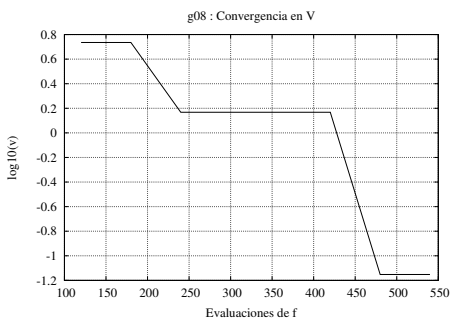
EDPR (MEF = 180,000)





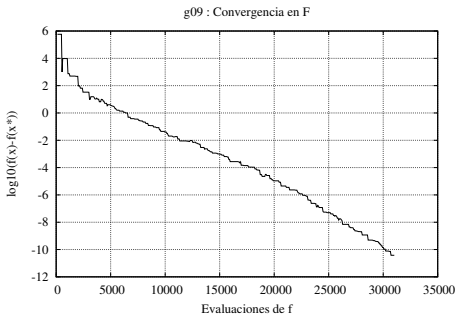
HEDSPR (MEF = 216,000)

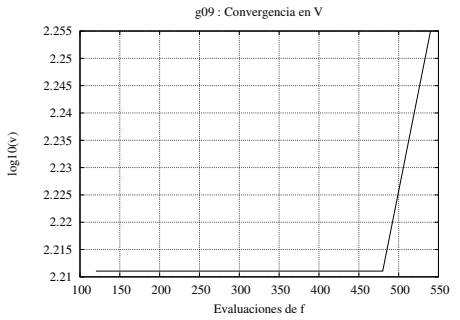




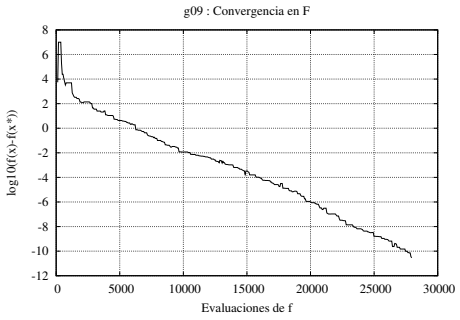
Gráficas de convergencia: Función $g09$

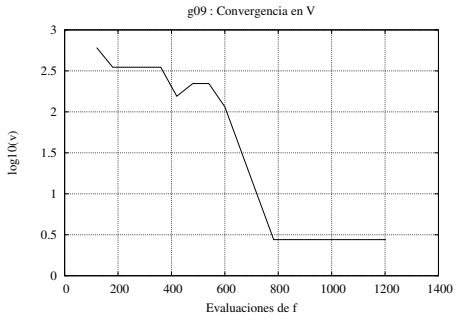
EDPR (MEF = 180,000)





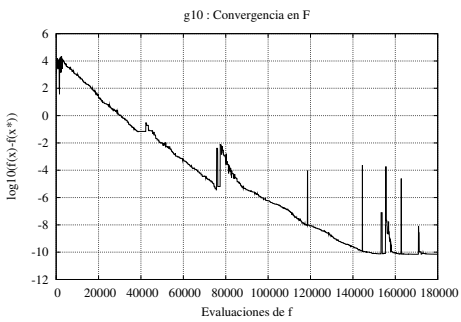
HEDSPR (MEF = 216,000)

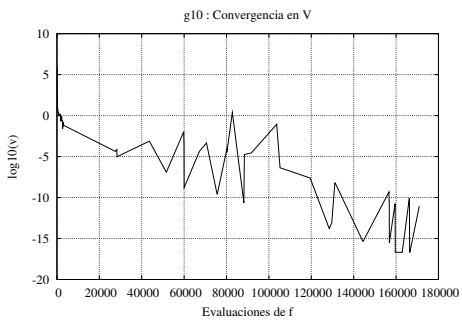




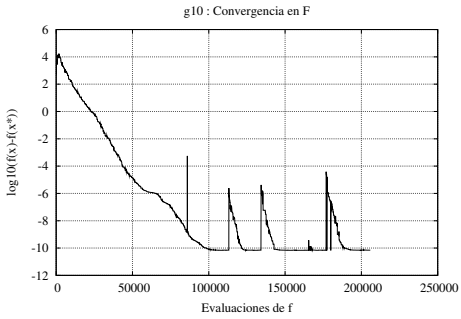
Gráficas de convergencia: Función $g10$

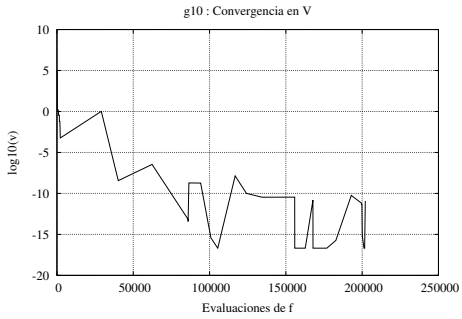
EDPR (MEF = 180,000)





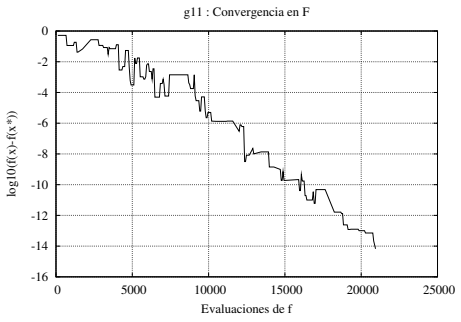
HEDSPR (MEF = 216,000)

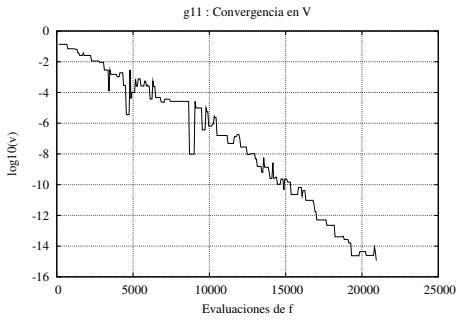




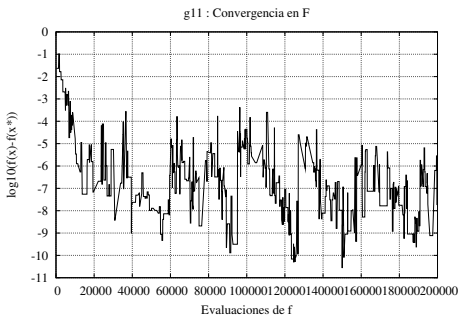
Gráficas de convergencia: Función $g11$

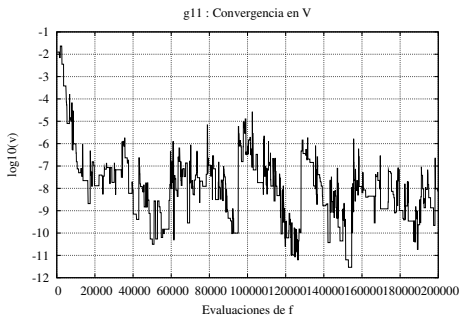
EDPR (MEF = 180,000)





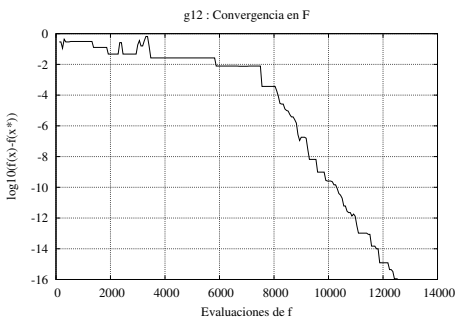
HEDSPR (MEF = 216,000)

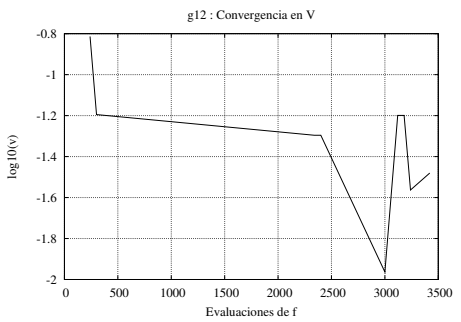




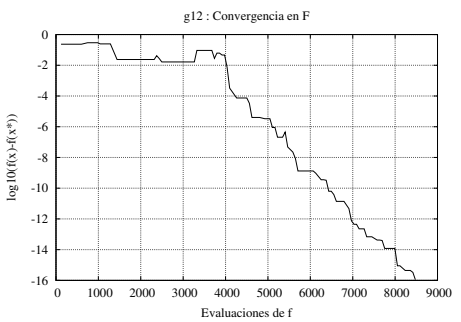
Gráficas de convergencia: Función $g12$

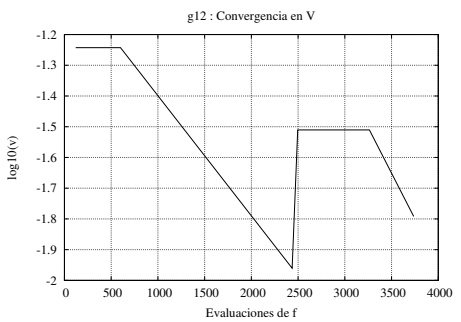
EDPR (MEF = 180,000)





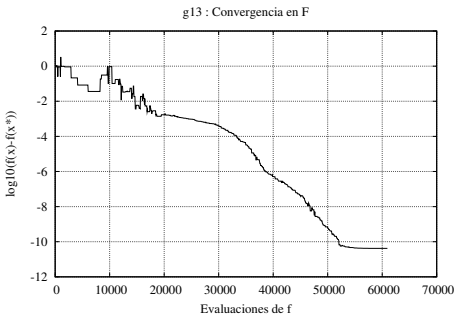
HEDSPR (MEF = 216,000)

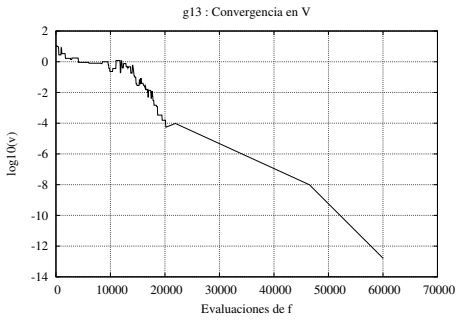




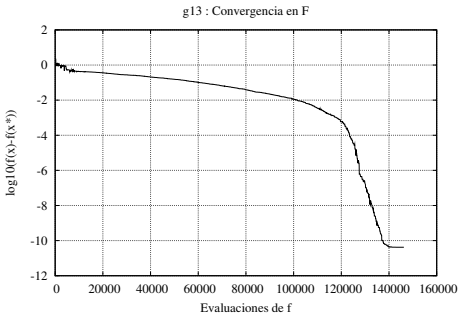
Gráficas de convergencia: Función $g13$

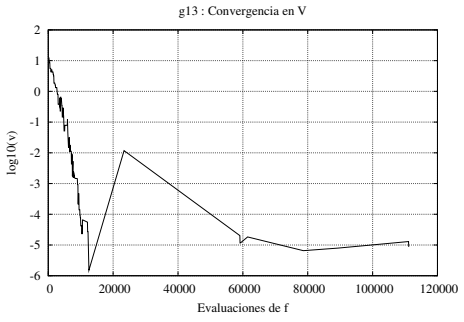
EDPR (MEF = 180,000)





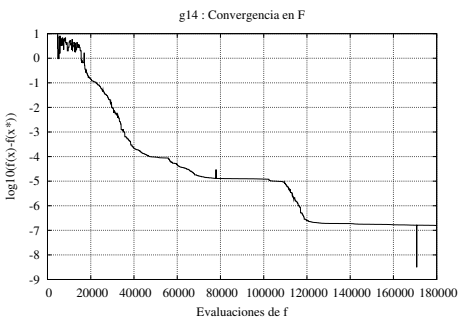
HEDSPR (MEF = 216,000)

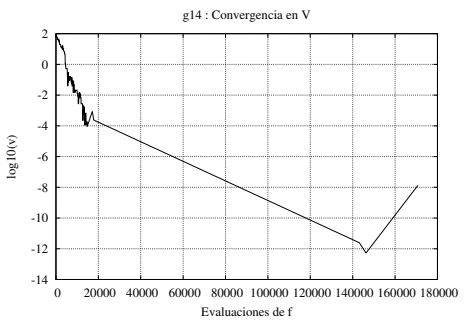




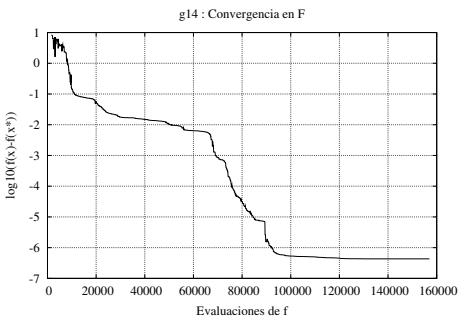
Gráficas de convergencia: Función $g14$

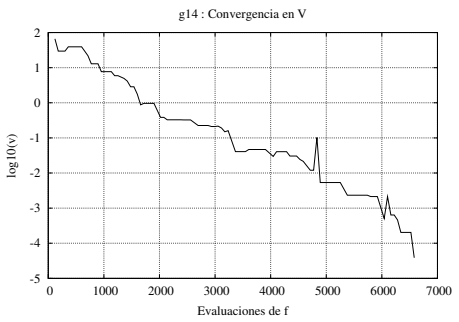
EDPR (MEF = 180,000)





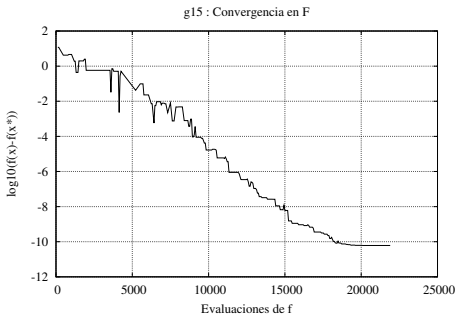
HEDSPR (MEF = 216,000)

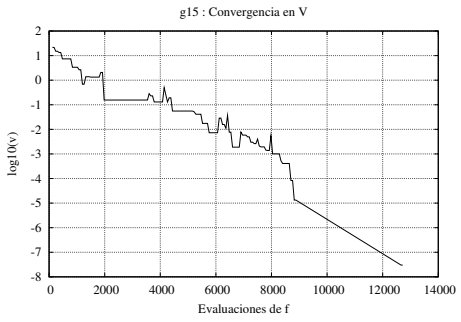




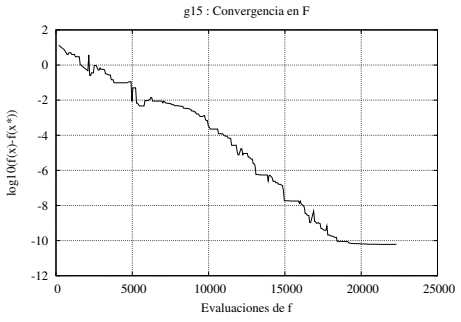
Gráficas de convergencia: Función $g15$

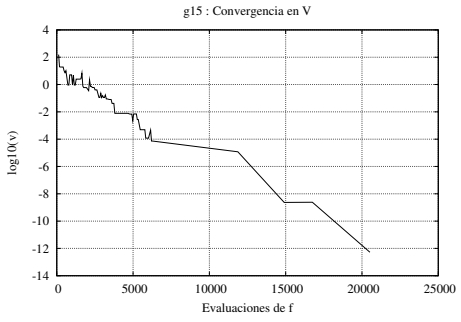
EDPR (MEF = 180,000)





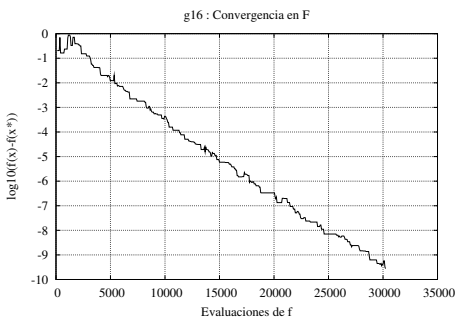
HEDSPR (MEF = 216,000)

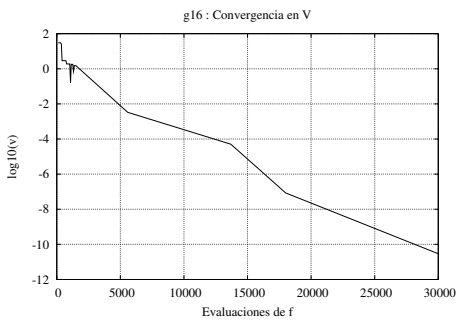




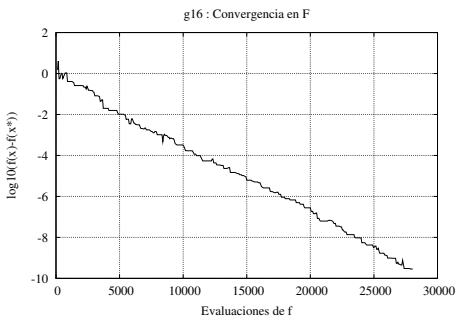
Gráficas de convergencia: Función $g16$

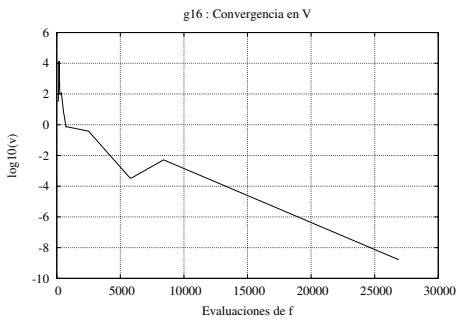
EDPR (MEF = 180,000)





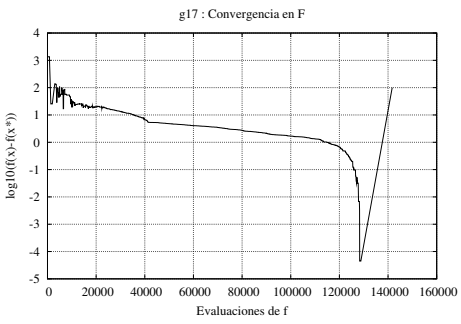
HEDSPR (MEF = 216,000)

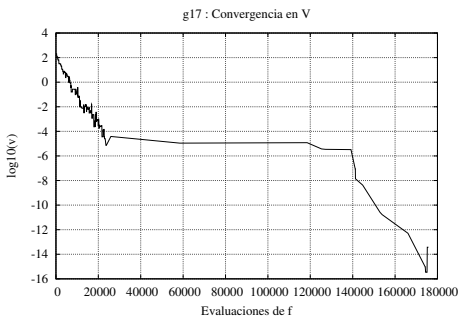




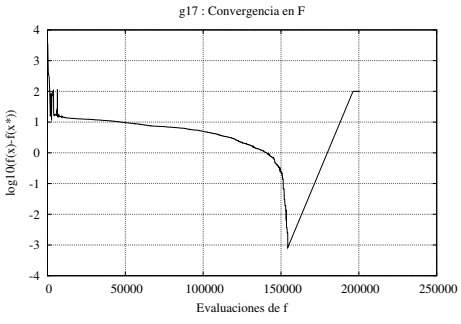
Gráficas de convergencia: Función $g17$

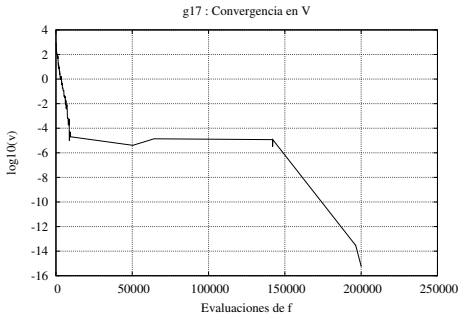
EDPR (MEF = 180,000)





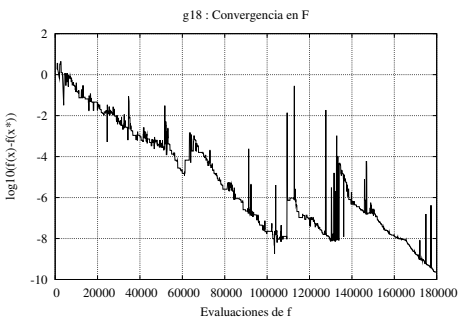
HEDSPR (MEF = 216,000)

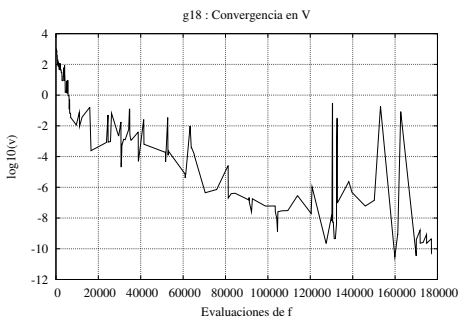




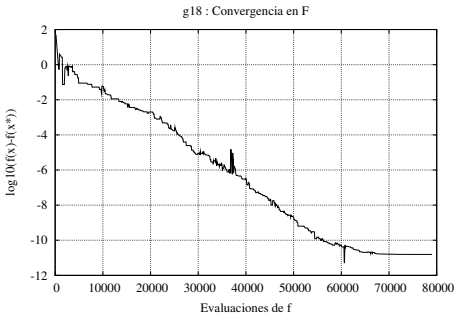
Gráficas de convergencia: Función $g18$

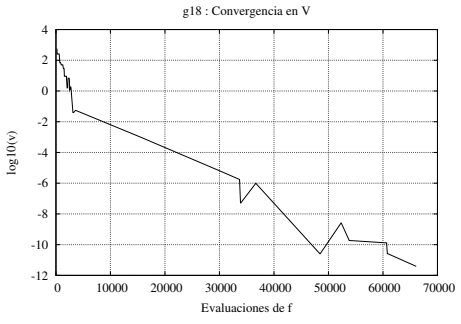
EDPR (MEF = 180,000)





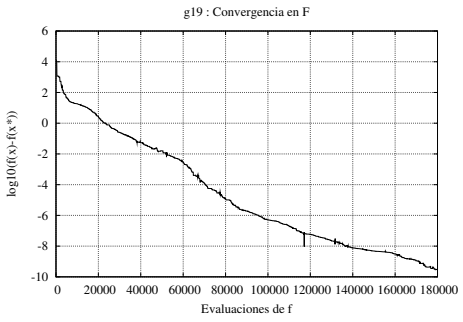
HEDSPR (MEF = 216,000)

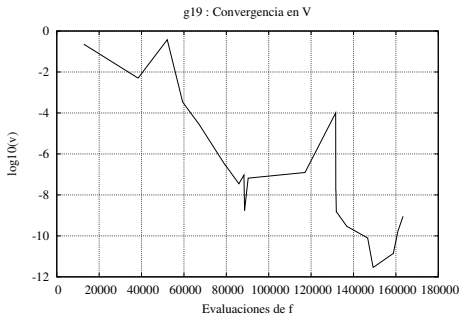




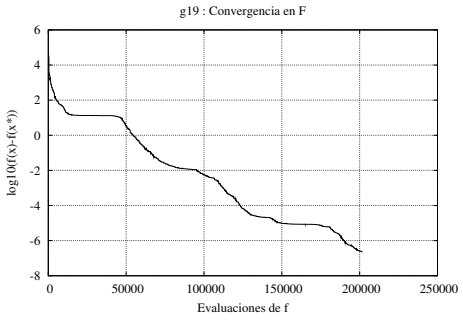
Gráficas de convergencia: Función $g19$

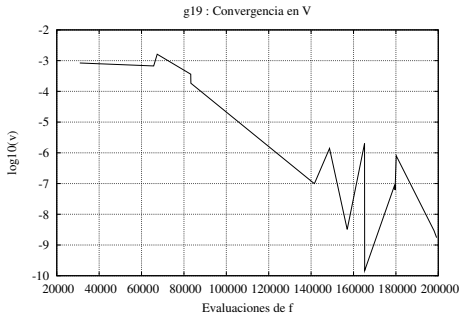
EDPR (MEF = 180,000)





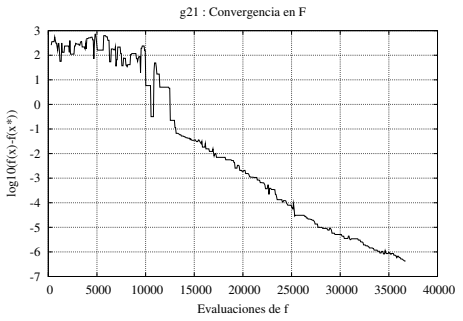
HEDSPR (MEF = 216,000)

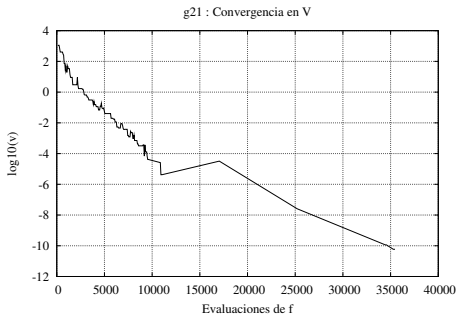




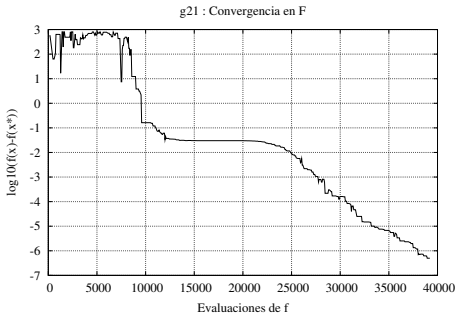
Gráficas de convergencia: Función $g21$

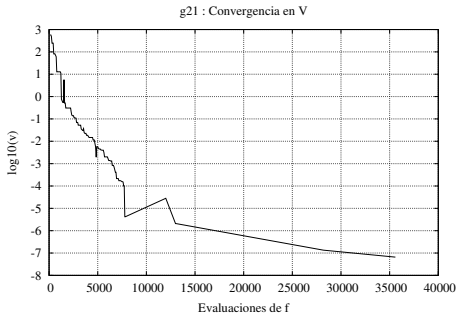
EDPR (MEF = 180,000)





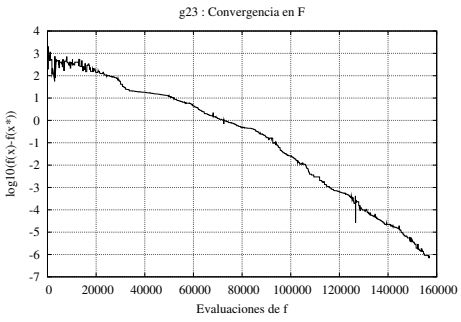
HEDSPR (MEF = 216,000)

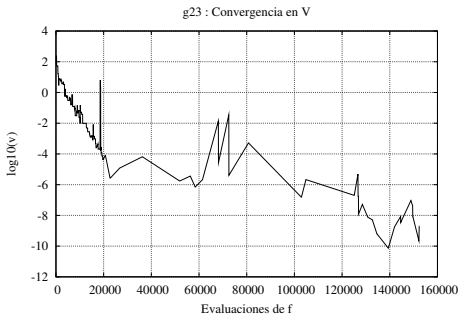




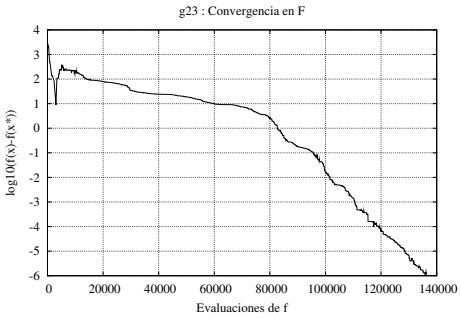
Gráficas de convergencia: Función $g23$

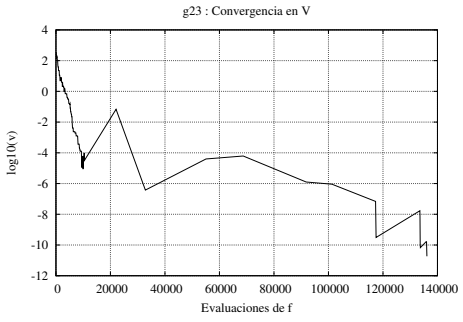
EDPR (MEF = 180,000)





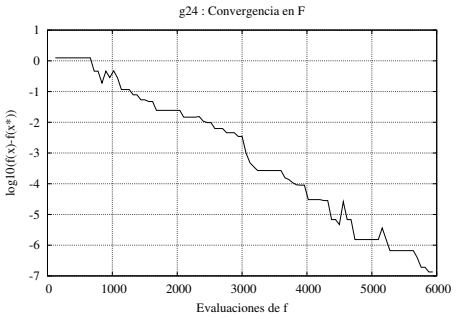
HEDSPR (MEF = 216,000)

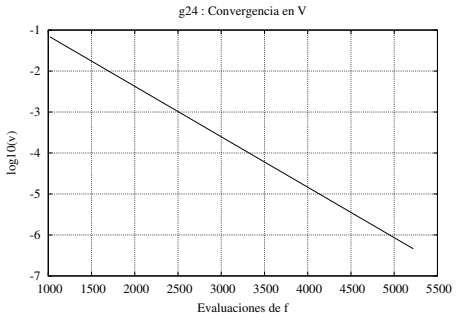




Gráficas de convergencia: Función $g24$

EDPR (MEF = 180,000)





HEDSPR (MEF = 216,000)