UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO

FACULTAD DE INGENIERÍA ELÉCTRICA
DIVISIÓN DE ESTUDIOS DE POSGRADO

"OPTIMIZACIÓN ESTOCÁSTICA DE FUNCIONES EN ESPACIOS ACOTADOS POR RESTRICCIONES LINEALES"

TESIS

QUE PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS EN INGENIERÍA ELÉCTRICA

PRESENTA
ING. ERICK GALAAD DE LA VEGA CARDIEL

DIRECTOR DE TESIS
DR. FÉLIX CALDERÓN SOLORIO

MORELIA, MICHOACÁN FEBRERO DE 2011

OPTIMIZACIÓN ESTOCÁSTICA DE FUNCIONES EN ESPACIOS ACOTADOS POR RESTRICCIONES LINEALES

TESIS

Que para obtener el grado de

MAESTRO EN CIENCIAS EN INGENIERÍA ELÉCTRICA

presenta

Erick Galaad De la Vega Cardiel

Dr. Félix Calderón Solorio

Director de Tesis

Universidad Michoacana de San Nicolás de Hidalgo

División de Estudios de Posgrado de la Facultad de Ingeniería Eléctrica

Febrero 2012

A mi familia que siempre me ha brindado su apoyo incondicional.

A mi amada esposa, Paola, por su comprensión y apoyo en los momentos más complicados.

A mi asesor el Dr. Félix Calderón por su dedicación, su continuo apoyo y ayuda inestimable. Siempre encontró el tiempo necesario para orientarme en el camino de la investigación. Su espíritu crítico y constructivo han influenciado mucho las orientaciones seguidas en esta tesis.

Resumen

La optimización global basada en algoritmos evolutivos puede ser utilizada para resolver muchos de los problemas de optimización propuestos en el área de ingeniería. Estos algoritmos han presentado resultados prometedores para resolver problemas de optimización donde la función objetivo puede ser no-lineal, no-diferenciable, o multimodal. En particular, se utiliza el algoritmo evolutivo Evolución Diferencial (DE) debido a su simplicidad, confiabilidad y eficiencia sobre espacios continuos. Este algoritmo es suficientemente potente para converger de manera confiable en el óptimo global.

Sin embargo, la mayoría de los problemas de optimización en el mundo real involucran encontrar una solución óptima sujeta a una o más restricciones. Las restricciones típicamente hacen que la optimización sea más complicada, porque crean regiones prohibidas en el espacio de búsqueda de la función objetivo y el verdadero reto es crear soluciones que siempre cumplan con las restricciones dadas.
Para resolver este problema, aumentar la precisión de la solución y la velocidad de convergencia, se plantea la posibilidad de la creación de un algoritmo que utiliza la generación de números aleatorios como principal característica y como complemento a la metaheurística utilizada por DE. Esta generación de números se realiza mediante el algoritmo propuesto (NGFRS), el cual utiliza la envoltura convexa generada por las restricciones para modelar la forma geométrica de la región de factibilidad, asegurando que todo número generado se encuentre dentro de la región de factibilidad.

En esta tesis se propone un algoritmo que toma como base los principios básicos del funcionamiento de DE, mejorando su desempeño con la aportación del algoritmo NGFRS, logrando un algoritmo de optimización global sujeto a restricciones (DE-NGFRS). El algoritmo propuesto utiliza pocos parámetros de ajuste y supera notablemente en tiempo y precisión a DE utilizando las técnicas generales más conocidas para manejar los parámetros que se encuentren fuera de los límites de la región de factibilidad, como los son las funciones de penalización y las funciones de reparación. En particular, DE-NGFRS fué probado con algunas funciones de referencia y con el problema de Despacho Económico de Energía Eléctrica, demostrando ser un algoritmo robusto y preciso.

Abstract

Global optimization based on evolutionary algorithms can be used for solving many engineering optimization problems. These algorithms have yielded promising results in solving nonlinear, non-differentiable, and multi-modal optimization problems. Differential Evolution (DE) particularly is used due to his simplicity, reliability and efficiency over continuous spaces. This algorithm is powerful enough to reliably converge to the true optimum.

The majority of real-world problems involve finding a solution that not only is optimal, but also satisfies one or more constraints. Typically these constraints make optimization harder, because they can create forbidden regions on the search space, and the challenge is to create solutions that always meet the given constraints. To solve this problem, increase the accuracy of the solution and the convergence speed, raises the possibility of the creation of an algorithm that uses the generation of random numbers as its main feature and as a complement to the DE methaheuristic. This random number generation is performed through the proposed algorithm (NGFRS), using the constraint-generated convex hull to model the geometric shape of the feasible region, ensuring that any number is generated within the feasible region.

This thesis proposes an algorithm that takes DE's basic principles of operation, improving his performance with the contribution of the NGFRS algorithm, achieving a global optimizer algorithm that can handle constraints (DE-NGFRS). The proposed algorithm uses just a few adjust parameters and is far superior in time and acuraccy to DE using the more general techniques for handling out-bounds parameters, i.e, penalty and repair functions. DE-NGFRS was tested with some benchmark functions and the real-world problem Economic Dispatch of electrical energy, proving to be a robust and accurate algorithm.

Contenido

Lista de Figuras

Lista de Tablas

Lista de Algoritmos

Lista de Símbolos

$\mathbb{R}$	Conjunto de números reales
$P(x)$	Probabilidad de x
μ	Media
σ	Varianza
Σ	Matriz de covarianza
$\mathcal{E}$	Conjunto de igualdades
$\mathcal{I}$	Conjunto de desigualdades
x^*	Óptimo de una función
$\mathbb{E}$	Función de error para mínimos cuadrados
Ω	Región de factibilidad
C_r	Factor de cruza para DE
F	Factor escalamiento para DE
γ	Factor de desición en DE-NGFRS
q	Conjunto de vértices que definen la envoltura convexa
p	Conjunto de puntos generados
$E[x]$	Valor esperado de x
R	Matriz de rotación
λ_i	Valores característicos
Λ	Matriz diagonal de valores característicos
L	Matriz de la raíz cuadrada de los valores diagonales de Λ
P_c	Probabilidad de cruza
N_{pop}	Número de individuos en una población
D	Dimensiones
g	Generaciones
$P^{(g)}$	Población en la generación g
$x_i^{(g)}$	Individuo objetivo
$v_i^{(g)}$	Individuo mutante
$u_i^{(g)}$	Individuo de prueba
$\nabla f(x)$	Gradiente de $f(x)$
$\nabla^2 f(x)$	Hessiano de $f(x)$
$\mu_{f(x)}$	Media de la función de error $f(x)$
λ	Factor de penalización
ψ	Conjunto de restricciones

Capítulo 1

Introducción

Durante los últimos años, el campo de la optimización ha estado en proceso de desarrollo y evolución [Nocedal y Wright, 1999]. Las técnicas de optimización y sus conceptos no están limitados a alguna disciplina en particular y dado el beneficio que proveen, están tomando un rol importante en la solución de problemas de ingeniería, economía, biología, estadística y algunas otras áreas.

La optimización de un problema es vista como un proceso de decisión que involucra encontrar y escoger las mejores evaluaciones sobre un conjunto n de posibles valores. Estos posibles valores se denominan variables de decisión $(w_1, w_2, \ldots, w_n)$, donde la medida de desempeño de cada una (por ejemplo, la ganancia o la pérdida) se mide a través de una función matemática f, llamada función objetivo [Hillier y Lieberman, 2006].

Cuando se aplican las técnicas de optimización a un problema, existen dos posibles variantes, la minimización y la maximización. Para un problema de minimización, el mejor resultado encontrado será aquel con el valor más pequeño dentro del conjunto de valores; por otro lado, si se trata de un problema de maximización el mejor resultado será el mayor valor posible. Por motivos de generalización, solo se tratarán problemas de minimización, dado que minimizar una función f es equivalente a maximizar una función $-f$.

Los problemas de optimización pueden ser clasificados en categorías de acuerdo a las distintas propiedades de la función objetivo, las restricciones y las variables de decisión.

La función objetivo puede ser lineal o no-lineal y las restricciones (si hay alguna presente) también son clasificadas como lineales o no-lineales.

Las variables de decisión pueden ser continuas, discretas o una combinación de ambas. Si un problema tiene una función objetivo lineal y restricciones lineales se considera que es un problema de optimización lineal, sin embargo si en un problema encontramos no-linealidad en la función objetivo, las restricciones o en ambas, se clasifica como un problema de optimización no-lineal [Chong y Zak, 2001]. La investigación presentada en este trabajo involucra encontrar la solución de problemas de optimización lineales y no-lineales.

1.1. Planteamiento del problema

En general un problema de optimización está dado por la Ecuación (1.1).

$$min \;\; f(x) \;\; \text{s.a} \begin{cases} c_i(x) \leq 0 & \text{para } i \in \mathcal{I} \\ c_i(x) = 0 & \text{para } i \in \mathcal{E} \end{cases} \tag{1.1}$$

donde f es la función objetivo, x es un vector con j variables independientes $\forall j \in [1, \ldots, D]$ tal que $x = [x_1, x_2, \ldots, x_D]^T \in \mathbb{R}^n$, $\mathcal{E}$ es el conjunto de restricciones de igualdad, y $\mathcal{I}$ es el conjunto de restricciones de desigualdad [Nocedal y Wright, 1999]. Con esta información podemos definir en la Ecuación (1.2) a un conjunto factible Ω como el conjunto de vectores x que satisfacen las restricciones; cabe destacar que las propiedades geométricas que caracterizan a Ω serán de gran utilidad a lo largo de este trabajo.

$$\Omega = \{x | c_i(x) = 0, \;\; i \in \mathcal{E} \wedge c_j(x) \geq 0, \;\; j \in \mathcal{I}\} \tag{1.2}$$

Dado el conjunto Ω, podemos reescribir (1.1) de manera más compacta en la Ecuación (1.3), y decimos que el vector $x \in \Omega$ que resulte con la menor evaluación de su función objetivo se denominará óptimo.

$$\min_{x \in \Omega} f(x) \tag{1.3}$$

La Figura 1.1, muestra una función que contiene dos óptimos, los cuales se definen de la siguiente manera:

Un punto x^* es un *óptimo local* del problema si $x^* \in \Omega$ y existe un entorno $\mathcal{N}$ de x tal que $f(x^*) \leq f(x)\ \forall x \in \mathcal{N} \cap \Omega$.

Un punto x^* es un *óptimo global* del problema si $x^* \in \Omega$ y $f(x^*) \leq f(x)\ \forall x \in \Omega$.

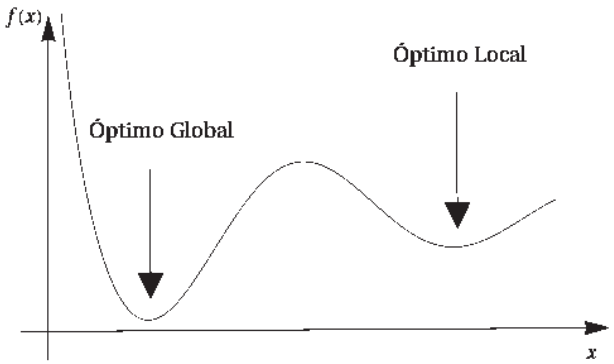


Figura 1.1: Diferencia entre un óptimo local y un óptimo global

En general, en el problema dado por la Ecuación (1.3) los óptimos globales son difíciles de encontrar [Nocedal y Wright, 1999, pág. 6] y aunado a esto, las restricciones del problema ocasionan que la optimización se convierta en una tarea más complicada, porque es necesario realizar la distinción entre una solución factible y una solución no-factible. Muy pocos métodos son capaces de validar de manera confiable los resultados, y los métodos más conocidos carecen de precisión ó son completamente dependientes del problema [Michalewicz, 1995].

Dependiendo de la naturaleza de las restricciones (linealidad y no-linealidad), Ω puede adquirir distintas propiedades (por ejemplo convexidad) y con el debido análisis, estas propiedades servirán para proponer un método que siempre genere soluciones dentro de Ω

y que permita determinar si una solución es válida o no.

Entonces, el problema principal es resolver la Ecuación (1.3) generando soluciones que se encuentren siempre dentro de Ω, aplicar un proceso de búsqueda de la solución y validar las soluciones encontradas. Finalmente, aplicando un algoritmo de optimización, se desea mejorar la eficiencia de la búsqueda y la calidad de la solución encontrada.

1.2. Antecedentes

Desde comienzos de 1940, se ha realizado un gran esfuerzo en el desarrollo de algoritmos para resolver distintas clases de problemas de optimización, analizando sus propiedades y desarrollando buenas implementaciones de software [Lee y El-Sharkawi, 2008]. La efectividad de estos algoritmos, por ejemplo, su habilidad para resolver problemas de optimización sujetos a restricciones, varía considerablemente y depende de factores tales como las formas particulares de la función objetivo, las restricciones del problema, cuantas variables y restricciones existen y alguna otra estructura especial como la dependencia entre las variables.

Aún cuando la función objetivo y las restricciones son suaves (por ejemplo, función objetivo diferenciable y restricciones lineales) el problema general de optimización es sorprendentemente difícil de resolver [Boyd y Vandenberghe, 2004, pág. 3], donde los métodos más utilizados son los basados en gradiente [Nocedal y Wright, 1999]. Sin embargo para resolver un problema como el planteado en la Sección 1.1, donde existe dependencia entre las variables y la función objetivo puede ser lineal o no-lineal, los métodos basados en gradiente no son suficientes, porque en general involucran alguna desventaja, tal como tiempo de cómputo largo, gasto de memoria innecesario o la posibilidad de no encontrar la solución [Boyd y Vandenberghe, 2004].

Debido a su fácil entendimiento e implementación, los métodos más conocidos para resolver este tipo de problemas son los Algoritmos Evolutivos (*Evolutionary Algorithms* **EA**), estos algoritmos calculan aleatoriamente un conjunto de soluciones iniciales a los

que se les denomina población inicial. La manera más común de generar una población aleatoria inicial en una región factible A_f, es calculando un número aleatorio con distribución uniforme por medio de la ecuación (1.4). La población generada estará uniformemente distribuida dentro de A_f, por lo tanto cuando se minimiza la función $f(x)$ con restricciones del tipo de $x_j^{min} \leq x_j \leq x_j^{max}$ es muy fácil calcular la solución dentro de un espacio con estas características utilizando EAs.

$$x_j = x_j^{min} + \alpha(x_j^{max} - x_j^{min}) \quad \forall j \in [1, D] \tag{1.4}$$

donde D es el número de dimensiones, $\alpha \sim U(0, 1)$, x_j^{min} es el límite inferior y x_j^{max} el límite superior de la variable x_j.

Una vez generada la población inicial, el proceso evolutivo generará soluciones de prueba para buscar x^*, sin embargo, las soluciones generadas mediante este proceso no garantizan encontrarse dentro de Ω, permitiendo que soluciones no-factibles y con una menor evaluación de su función objetivo sean consideradas como óptimas. Por lo tanto, se han desarrollado distintas técnicas para manejar las restricciones de un problema, entre las cuales destacan, las técnicas de penalización [Montes y Coello, 1998] y las técnicas de reparación [Torn y Zilinskas, 1989, Storn y Price, 2005].

Es común, en EAs, utilizar funciones de penalización para favorecer que la solución permanezca dentro de Ω. En las funciones de penalización, por cada restricción lineal que es violada se paga un costo λ. Sin embargo, el costo asociado sesga la solución perdiendo precisión y aumentando el tiempo de búsqueda.

Por otro lado en las técnicas de reparación, los parámetros violados son reinicializados en valores aleatorios dentro del rango permitido. Sin embargo, cuando la solución está muy cercana a los límites de Ω (por ejemplo en un vértice o una arista de Ω), la reinicialización abrupta de los parámetros provocaría una lenta convergencia del algoritmo [Storn y Price, 2005].

1.3. Motivación

Consideremos el siguiente problema: dado un conjunto factible Ω_1 representado mediante cuatro desigualdades en la Ecuación (1.5). Se desea generar un conjunto de puntos mediante la Ecuación (1.4).

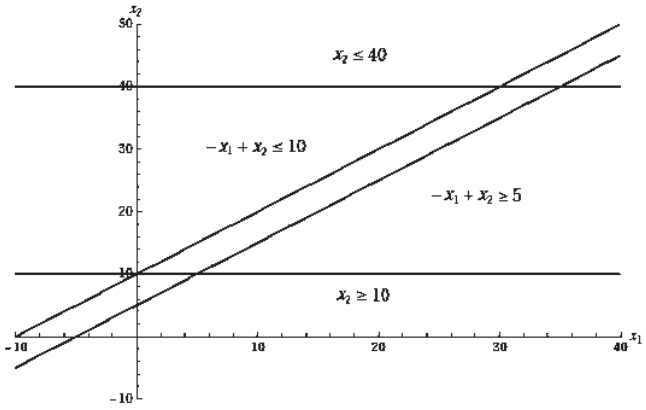
$$\Omega_1 = \begin{cases} x_2 & \geq & 10 \\ x_2 & \leq & 40 \\ -x_1 + x_2 & \leq & 10 \\ -x_1 + x_2 & \geq & 5 \end{cases} \tag{1.5}$$

En la Figura 1.2(a) podemos observar la representación gráfica del conjunto de desigualdades contenido en la Ecuación (1.5). Cada una de las desigualdades representa una semi-espacio, de manera que Ω_1 es la intersección de este conjunto de semi-espacios.

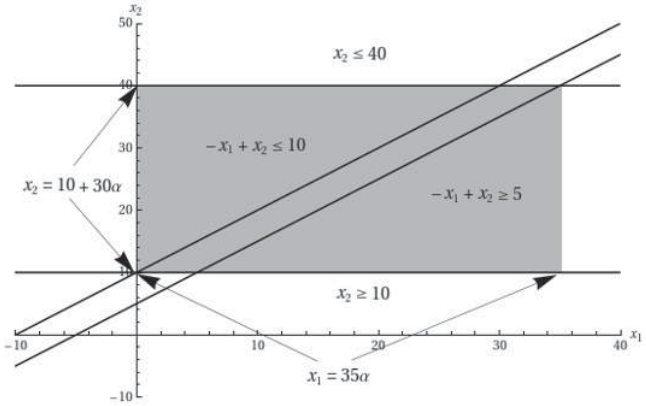
Dado que queremos utilizar la Ecuación (1.4) para generar un conjunto de puntos, es necesario acotar el espacio mediante límites inferiores y límites superiores. Para x_1 decimos que: $0 \leq x_1 \leq 35$ y para x_2 decimos que: $10 \leq x_2 \leq 40$. En la Figura 1.2(b) el espacio acotado por los límites de x_1 y x_2 se muestra como un "rectángulo" sombreado. Si se sustituyen los límites de x_1 y x_2 en la Ecuación (1.4) podemos deducir que $x_1 = 35\alpha$ y $x_2 = 10 + 30\alpha$. Si $\alpha \sim U(0, 1)$, entonces la solución será generada dentro del espacio acotado por los límites inferiores y superiores de x_1 y x_2.

Sin embargo, como se puede observar en la Figura 1.2(c), Ω_1 es una delgada banda inclinada que está acotada por límites inferiores y superiores. Por lo tanto cuando se generan soluciones como en la Figura 1.3(a) solamente algunas soluciones son generadas dentro de Ω_1 y la gran mayoría se encuentra en una región no-factible.

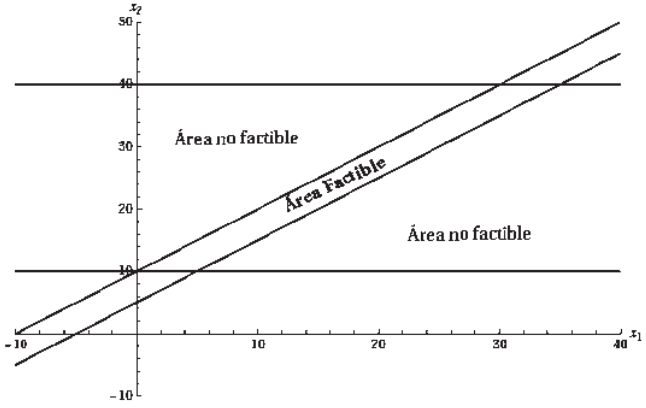
Más formalmente podemos decir que la probabilidad de generación de un número



(a)

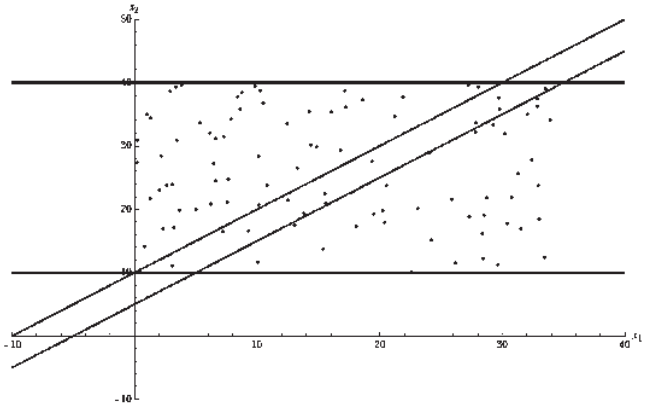


(b)

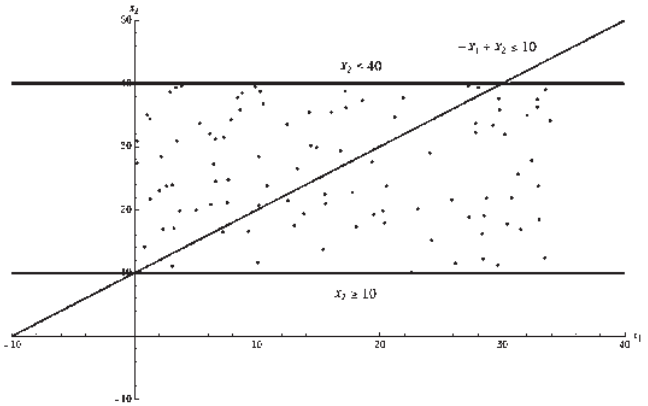


(c)

Figura 1.2: Restricciones. a) Interpretación gráfica de la Ecuación (1.5), b) Zona acotada entre $[x_1^{min}, x_1^{max}]$ y $[x_2^{min}, x_2^{max}]$, c) Identificación de factibilidad



(a)



(b)

Figura 1.3: Generacion de números. a) Probabilidad muy pequeña de generar soluciones dentro de la región de factibilidad, b) Probabilidad nula

dentro de Ω_1 está dada por la Ecuación (1.6).

$$P(A_f) = \frac{A_f}{A_T} \tag{1.6}$$

donde el A_f es el área factible y A_T es el área total.

En nuestro ejemplo $A_f = 150$ y $A_T = 1050$, de tal manera que $P(A_f) = \frac{150}{1050} = 0.14$ la cual es una probabilidad muy baja.

Si decidiéramos eliminar una restricción del conjunto Ω_1 como en la Figura 1.3(b), la probabilidad de generar un número dentro de Ω_1 sería $P(A_f) = \frac{A_f}{A_T} = 0$, de tal manera

que ningún número sería generado dentro de ella, y por lo tanto será necesario proponer otro método de generación de puntos.

Un ejemplo de un problema real con restricciones, es el Despacho Económico (*Economic Dispatch*-**ED**) [Stevenson, 1982], el cual consiste en suministrar energía eléctrica a una carga utilizando un conjunto de unidades de generación. Los generadores eléctricos tienen restricciones que limitan la energía que proporciona cada unidad y una restricción lineal que tiene que ver con conservación de energía. En la Figura 1.4 se puede observar que para un problema de ED en tres dimensiones, los límites de generación de potencia forman un cubo y la restricción de conservación de la energía es un plano, de manera que Ω está definido por la intersección del plano y el cubo.

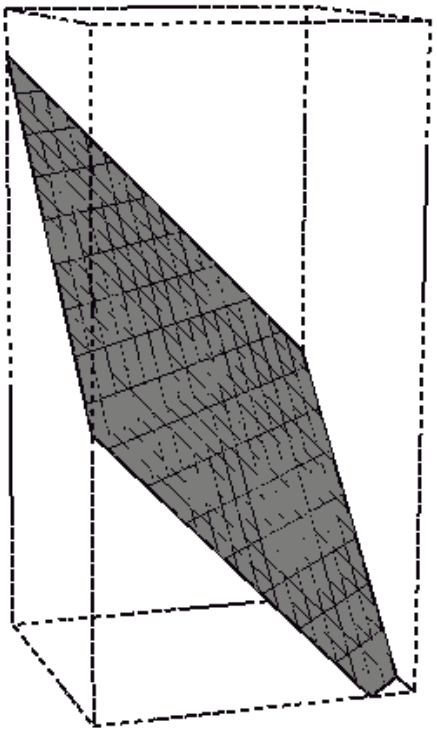


Figura 1.4: Límites de generación (cubo) y restricción lineal de conservación de la energía (plano)

En la década pasada, muchos investigadores se esforzaron para mejorar las técnicas de optimización y resolver el problema de ED [Chowdhury y Rahman, 1990], por ejemplo, la Programación Genética (*Genetic Programming*-**PG**) aplicada a resolver el problema de ED con múltiples funciones de costo de combustible. Los Algoritmos Genéticos (*Genetic Algorithms*-**GA**) se han aplicado al problema de ED con distintos tipos de funciones de costo. Wong en [Wong y Fung, 1993] presentó un GA híbrido y la técnica de Recocido Simulado (*Simulated Annealing*-**SA**) para tratar de resolver el problema. En [Santos y Marianib, 2007] se propone una solución del problema utilizando Evolución Diferencial (*Differential Evolution*-**DE**) con funciones de penalización para un problema de DE con restricciones y zonas prohibidas de generación. [Calderón et al., 2008] presentan una solución basada en GA con un conjunto de vectores que delimitan el área de factibilidad, sin embargo, debido a la naturaleza del problema de ED, el cálculo de este conjunto puede resultar muy costoso en tiempo de cómputo.

[Lara et al., 2009] proponen un algoritmo para calcular la envoltura convexa correspondiente al problema de ED donde existe un conjunto de desigualdades que delimitan el hiper–cubo de generación y una restricción lineal correspondiente a un hiper–plano. La intersección del hiper–cubo y el hiper–plano es donde viven las soluciones factibles del problema. Con este conjunto de puntos resulta mas sencillo para el algoritmo propuesto por [Calderón et al., 2008] calcular una población inicial que viva dentro de la región de factibilidad haciendo una combinación lineal de los puntos.

Dado que se requiere mejorar el tiempo de búsqueda y la calidad de la solución, se utilizará la técnica desarrollada por [Lara et al., 2009] para calcular el conjunto de vectores que delimiten la envoltura convexa de Ω.

1.4. Objetivos de la Tesis

1.4.1. Objetivo general

El objetivo general de la presente tesis, es la aportación de un nuevo esquema de generación de números aleatorios como complemento a la heurística que maneja DE, y que permita encontrar la solución óptima en el espacio de una función sujeta a restricciones,

de manera precisa, veloz y eficiente, así como demostrar su utilidad en la resolución de problemas reales.

1.4.2. Objetivos particulares

- Proponer un método para la generación de números aleatorios mediante una distribución gaussiana, que simule la forma de la región de factibilidad.
- Evitar la convergencia prematura a óptimos locales, incorporando el método de generación de números aleatorios como complemento a la estrategia de búsqueda.
- Implementar un algoritmo eficiente y robusto que garantice la convergencia en el óptimo global de la función sujeta a restricciones.
- Dada la envoltura convexa obtenida a partir de las restricciones de un problema, proponer un método distinto a los ya conocidos para manejar las restricciones y determinar si las soluciones generadas se encuentran dentro de Ω.
- Mejorar la precisión de la solución encontrada en comparación con la reportada por otros autores utilizando EAs.

1.5. Descripción de capítulos

El Capítulo 2 da una introducción a los números aleatorios, su creación y su relevancia en la toma de decisiones que utilizan los métodos de optimización estocásticos. Se mencionan dos de las más importantes distribuciones de probabilidad conocidas, la distribución normal y la distribución uniforme, las cuales son ampliamente utilizadas por los algoritmos de búsqueda estocástico para la toma de sus decisiones.

En las secciones finales del Capítulo 2 se demuestra que es posible generar un número aleatorio dentro de una región de factibilidad tal como lo propone Calderon en [Calderón et al., 2008]. Sin embargo se observa que es muy poco probable generar números cerca de los vértices de la región de factibilidad, por lo tanto se propone generar números aleatorios utilizando una distribución de probabilidad gaussiana que capture la forma de la

región de factibilidad mediante un elipsoide. Estos números son generados con una media μ y una covarianza σ distinta a la tradicional $N(0,1)$.
En la parte final del capítulo se propone un algoritmo para asegurar que cualquier número generado se encuentre dentro de la región de factibilidad.

El Capítulo 3 da una introducción al los tipos de problemas de optimización que existen, señalando las diferencias entre cada uno de ellos. Posteriormente hace referencia a las técnicas de optimización recomendadas para resolver los distintos tipos de problemas, entre ellas se mencionan los algoritmos deterministas y los algoritmos estocásticos. Se presentan los algoritmos evolutivos como una herramienta destacada para resolver problemas no-lineales de optimización global; particularmente se hace mención a dos algoritmos evolutivos: GA's y DE. Se describen las partes que conforman a cada uno de estos algoritmos, así como los pasos a seguir para resolver un problema con estos algoritmos. Finalmente se realiza una comparación entre ellos, definiendo las ventajas y desventajas que posee cada uno de ellos y se plantean las distintas maneras de manejar las restricciones de un problema.

En el capitulo 4 se desarrolla el algoritmo propuesto en este trabajo llamado DE-NGFRS, mencionando las partes integrales de su funcionamiento, las cuales marcan la diferencia entre los algoritmos clásicos que se utilizan para resolver problemas no-lineales de optimización global. Adicionalmente se presentan los resultados y comparaciones entre los métodos más comunes para el manejo de restricciones y DE-NGFRS. Esta comparación es realizada a través de la aplicación de los métodos a funciones de referencia, y al problema de Despacho Económico de energía eléctrica, (en el cual se analizan tres redes eléctricas distintas con múltiples restricciones). Finalmente se realiza un análisis de tiempo de búsqueda y precisión de la solución encontrada.

Las conclusiones y sugerencias para trabajo futuro se presentan en el Capítulo 5.

Capítulo 2

Generación de números aleatorios

2.1. Introducción

Como se vio en el capítulo anterior, nos interesa la manera en la que son generados distintos números aleatorios en una región factible. Generalmente, los números son calculados siguiendo una distribución de probabilidad, donde cada uno tiene una probabilidad específica de pertenecer a un rango de valores determinado. Dependiendo de la distribución de probabilidad escogida, la forma en que los números son generados cambia considerablemente, por ejemplo, no es lo mismo generar números con una distribución uniforme o generarlos con una distribución normal, la elección de que distribución utilizar dependerá de los objetivos que se deseen satisfacer.

Cuando los números aleatorios están involucrados en algún proceso de optimización, es necesario elegir una distribución de probabilidad adecuada que permita simplificar la búsqueda de la solución, y como se verá más adelante, la elección dependerá de la información que se tenga del problema y del espacio de búsqueda.

Por lo tanto, en este capítulo se introducen algunos conceptos referentes a los números aleatorios. Se describen sus características más importantes, su uso en aplicaciones prácticas, diferencias entre las distribuciones de probabilidad más comunes y se indican algunas técnicas de generación.

2.2. Números aleatorios

Los números aleatorios son secuencias de números reales generados como datos físicos obtenidos de un experimento aleatorio ó como la salida de un programa determínistico de computadora [Papoulis y Pillai, 1999]. Habitualmente, tienen una distribución de probabilidad asociada y debido a su gran funcionalidad, los números aleatorios han sido utilizados en diversas aplicaciones, tales como:

- Simulación de sistemas (colas de espera, loterías, juegos).
- Criptografía y Seguridad Informática
- Integración numérica usando Monte Carlo.
- Optimización de problemas

En la mayoría de estas aplicaciones es necesario generar una gran cantidad de números aleatorios. Cuando se genera una secuencia de números aleatorios con alguna distribución probabilidad mediante el uso de computadora se le conoce como *generador de números aleatorios* [Kennedy y Gentle, 1980].

Generalmente, los generadores de números aleatorios utilizan algoritmos deterministicos que parten de un valor inicial llamado semilla. Estos programas deberían llamarse más propiamente generadores de números pseudo-aleatorios, ya que generan una secuencia de números que eventualmente puede volver a repetirse y solamente se aproxima a una secuencia aleatoria [Papoulis y Pillai, 1999].

Existen diferentes métodos de generación de números aleatorios que permiten obtener una secuencia de números aleatorios para una distribución dada. Sin embargo, la mayoría de estos métodos se basan en la generación de observaciones independientes de una distribución uniforme en el rango $[0, 1]$. Por lo tanto comenzaremos por analizar las propiedades de una distribución uniforme.

2.2.1. Generadores de números aleatorios con una distribución uniforme

Si X es una variable aleatoria que representa las posibles salidas de un experimento, y todas las posibles salidas tienen la misma probabilidad de ocurrir, entonces decimos que X está distribuida uniformemente [Grinstead y Snell, 1997]. La función de distribución de probabilidad uniforme está dada por la Ecuación (2.1).

$$P(X = j) = \frac{1}{n} \quad \forall j \in S \tag{2.1}$$

donde $n = |S|$ y S es el espacio muestral.

Muchos de los lenguajes de programación son capaces de generar secuencias de números aleatorios[1] con una distribución uniforme. Por ejemplo, podemos observar en la Figura 2.1 el histograma de la generación de n números aleatorios generados por la función $RandomInteger[]$, de Mathematica. Cada número entero en el rango $[1, 10]$ fue generado aproximadamente la misma cantidad de veces, por lo tanto la probabilidad de generar un número entero aleatorio en este rango es la misma para todos.

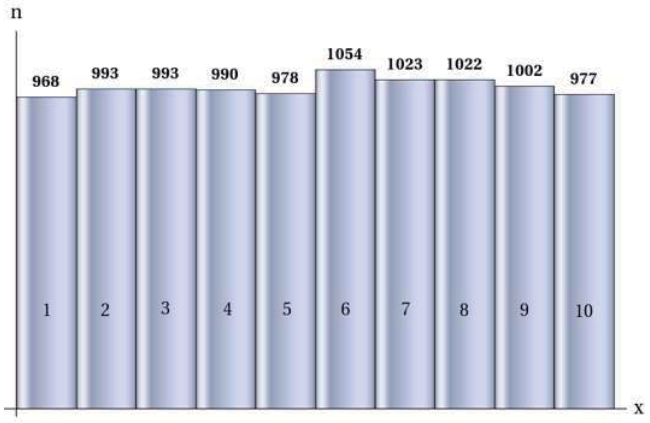


Figura 2.1: Números enteros aleatorios generados en el intervalo [1,10] con una distribución uniforme cuando $n = 10,000$

[1] Si son generados mediante un programa deterministico de computadora, son llamados números pseudo-aleatorios

En optimización, quizás el mayor uso de los números uniformemente distribuidos es la toma de decisiones y la generación de soluciones en un espacio de búsqueda.
Muchos algoritmos estocásticos utilizan la información de un número aleatorio uniformemente distribuido para determinar la probabilidad de tomar una decisión u otra. Por ejemplo, si se tienen dos alternativas (a y b) para dar el siguiente paso en un algoritmo, la variable aleatoria $X \sim U(0,1)$ puede ser utilizada para tomar la decisión. Cuando queremos la misma probabilidad para ambas alternativas, es decir, $P(a) = P(b) = 0.5$, generamos X y si el valor que toma es menor a 0.5 se escoge la alternativa a, de otro modo, se escoge la alternativa b. Sin embargo, si no deseáramos la misma probabilidad para ambos, en este caso $P(a) = 0.75$ y $P(b) = 0.25$, basta con generar X y verificar que si es menor a 0.75 se escoge la alternativa a, y si no, se escoge b.

Por otro lado, la generación de números aleatorios es ampliamente utilizada por los algoritmos de búsqueda estocástica, y la manera más común de generar números es cuando cubren uniformemente el espacio de búsqueda. En la Figura 2.2 podemos observar la manera en la que está distribuido un conjunto de soluciones generadas con una distribución uniforme.

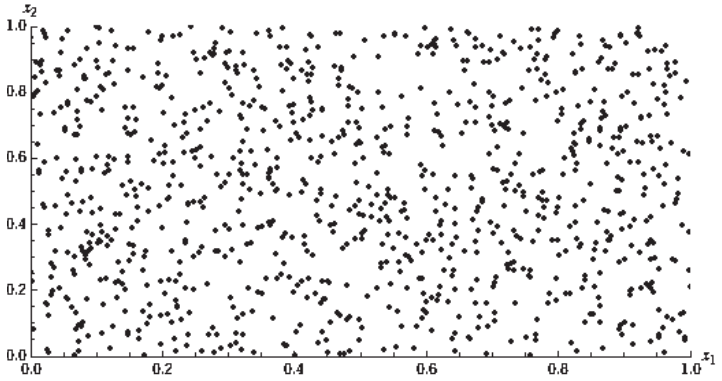


Figura 2.2: Conjunto de puntos generados mediante una distribución uniforme tal que $x_1 \sim U(0,1)$ y $x_2 \sim U(0,1)$

2.2.2. Generación de números aleatorios con una distribución normal

Existen situaciones donde se requiere generar números aleatorios que no se encuentren uniformemente distribuidos, por lo cual otra distribución de probabilidad es necesaria, y la más popular y conocida es la distribución gaussiana, comúnmente llamada *distribución normal*, porque la mayoría de las distribuciones de uso frecuente tienden a aproximarse a la distribución normal bajo ciertas condiciones [Lejarza y Lejarza, 2007].

La distribución normal es una de las más importantes distribuciones en el estudio de la probabilidad y la estadística. La importancia de esta distribución radica en que permite modelar numerosos fenómenos naturales, sociales y psicológicos [Devore, 2008].

Una variable aleatoria continua X tiene una distribución normal con parámetros μ y Σ si la función de densidad de probabilidad de X está dada por la Ecuación (2.2).

$$f(X;\mu,\Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} e^{-(X-\mu)^T\Sigma^{-1}(X-\mu)/2} \tag{2.2}$$

donde μ es la media muestral dada por la Ecuación (2.3) y Σ es la matriz de covarianza dada por la Ecuación (2.4).

$$\mu = \frac{x_1 + x_2 + \cdots + x_n}{n} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2.3}$$

$$\Sigma = \begin{bmatrix} \sigma_{1,1}^2 & \cdots & \cdots & \sigma_{1,D}^2 \\ \vdots & \sigma_{2,2}^2 & & \vdots \\ \vdots & & \ddots & \vdots \\ \sigma_{D,1}^2 & \cdots & \cdots & \sigma_{D,D}^2 \end{bmatrix} \tag{2.4}$$

donde σ^2 es la varianza y la desviación estándar es $\sigma = \sqrt{\sigma^2}$. En el caso unidimensional, para la Ecuación (2.4), $\Sigma = \sigma^2$. Por ejemplo, en la Figura 2.3 se presentan gráficas de $f(X;\mu,\sigma)$, con media $\mu = 0$ y distintos valores de σ . Cada curva de densidad es simétrica con respecto a μ, de modo que el centro de la gaussiana es tanto la media de la distribución,

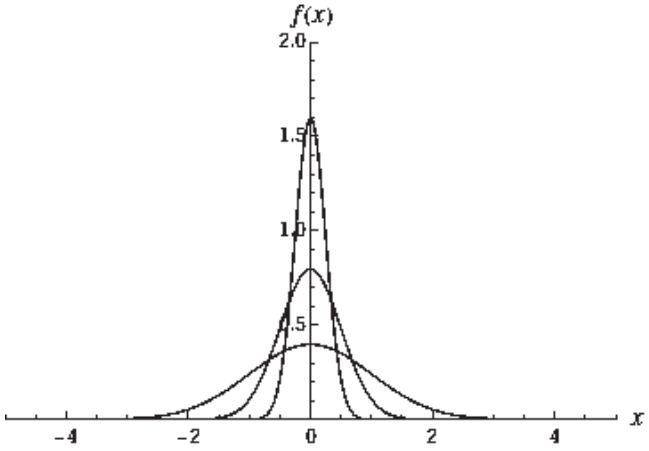


Figura 2.3: Distribución Normal con $\mu = 0$ y $\sigma = 1, 0.5$, y 0.25

como la mediana. El valor de σ es la distancia desde μ hasta los puntos de inflexión de la curva [Devore, 2008].

Cuando los valores de los parámetros de una distribución normal son $\mu = 0$ y $\sigma = 1$, la distribución es conocida como distribución normal estándar, y una variable aleatoria que tiene una distribución normal estándar se denotará por z.

Para simulaciones en una computadora, en ocasiones es útil generar números con una distribución normal estándar $z \sim N(0, 1)$ y existen distintos métodos para generarlos, donde algunos de los más conocidos son: el método de Box-Müller, el método de las coordenadas polares y el método de la densidad acumulada [Papoulis y Pillai, 1999]. Generalmente, en la mayoría de los lenguajes de programación existen funciones basadas en estos métodos que los generan con el simple llamado a una función.

Aunque la distribución normal estándar es comúnmente la más utilizada, en ocasiones es necesario generar números aleatorios con otros valores para μ y Σ. Por ejemplo, en la Figura 2.4, se generaron $n = 1000$ puntos, tal que $\mu = [0.5, 0.5]^T$ y $\Sigma = \begin{bmatrix} 0.25 & 0 \\ 0 & 0.25 \end{bmatrix}$. A diferencia de la Figura 2.2, los puntos generados se encuentran con mayor probabilidad en el centro de su media.

Dado que las distribuciones uniforme y normal son ampliamente utilizadas como herramientas de decisión y como generadores de soluciones en los algoritmos estocásticos,

discutiremos brevemente el funcionamiento de estos algoritmos y en capítulos posteriores se realizará un análisis detallado.

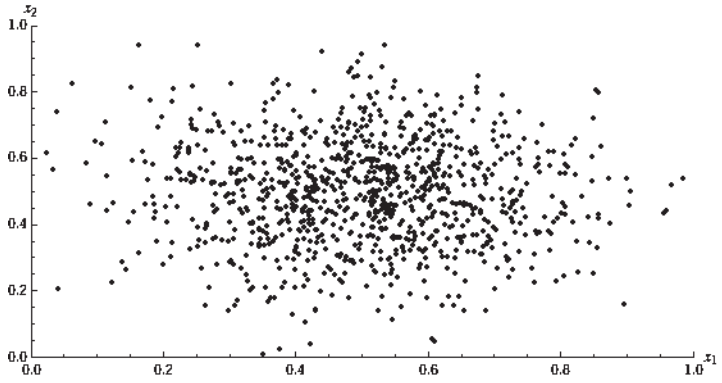


Figura 2.4: Conjunto de números aleatorios generados mediante una distribución normal en el rango $[0, 1]$

2.3. Algoritmos estocásticos

Representan una amplia clase de estrategias de optimización que utilizan heurísticas desarrolladas para la búsqueda de valores óptimos. Comúnmente están inspirados en procesos físicos, evolución natural y eventos estocásticos. Algunas de las características que estos algoritmos incluyen es que son completamente independientes del problema, su implementación es sencilla y generalmente son de fáciles de comprender [Kleywegt y Saphiro, 2000].

Quizás el algoritmo de optimización más simple en la categoría estocástica es la búsqueda aleatoria, debido a que puede ser efectiva en muchos problemas y a que su simplicidad relativa es una característica atractiva a los que la desarrollan.

2.3.1. Búsqueda aleatoria

Este método tiene algunas ventajas sobre la mayoría de las búsquedas. Las ventajas incluyen su sencillo entendimiento, una eficiencia computacional razonable y la facilidad de implementación, pues solo necesita obtener un rango de búsqueda [Chong y Zak, 2001]. El principio del funcionamiento de la búsqueda aleatoria es sencillo. Dada una función f

en el dominio acotado por $x_i \in (x_i^{min}, x_i^{max})\ \forall i \in [1, \ldots, D]$, se genera un punto x con la Ecuación (1.4) y se evalúa $f(x)$. En la siguiente iteración un nuevo punto es generado y evaluado. Después de k iteraciones del algoritmo, el valor x con $f(x)$ más pequeño es la solución.

Considere el siguiente ejemplo. Se desea minimizar la función $f(x_1, x_2) = x_2 + x_1 + 2x_1^2 + 2x_1x_2 + x_2^2$. Dados los dominios para $x_1 \in [-2, 2]$ y $x_2 \in [1, 3]$ podemos aplicar una búsqueda aleatoria sencilla. Cabe señalar que el mínimo de la función dentro del espacio de búsqueda acotado por el dominio de x_1 y x_2 es $x^* = [-0.75, 1]$ con $f(x^*) = 0.875$.

En la Tabla 2.1 se encuentran los resultados de $k = 1,000$ iteraciones realizadas por el algoritmo, donde se puede observar que el mejor valor se encuentra próximo al óptimo. Sin embargo, a pesar de su sencillez, este método carece de convergencia rápida debido a que es una búsqueda no guiada y dentro del espacio de búsqueda es muy complicado generar un número real con el valor exacto de x^*.

Dado que los límites $[x_i^{min}, x_i^{max}] \subset \mathbb{R}$, la probabilidad de generar x^* en este intervalo es $P(x^*) = \frac{1}{\infty} = 0$, debido a que cualquier intervalo que pertenece al conjunto $\mathbb{R}$ contiene una cantidad infinita de números.

Tabla 2.1: Minimización mediante búsqueda aleatoria

k	x_1	x_2	$f(x_1, x_2)$
0	0	0	0
1	-1.43332	2.04502	3.040305
2	-1.43332	2.04502	3.040305
3	-1.43332	2.04502	3.040305
4	-1.43332	2.04502	3.040305
12	-1.51498	1.32113	2.13889
17	-1.26989	1.29692	1.64039
28	-1.03509	1.25319	1.33709
52	-0.65149	1.13962	1.15084
82	-0.59307	1.08951	1.09464
276	-0.83062	1.00196	0.89064
575	-0.70825	1.00477	0.88606
999	-0.70825	1.00477	0.88606

Un aspecto a considerar es la limitación del espacio de búsqueda acotado por el dominio de x_i^{min} y x_i^{max}, porque no permite que ninguna solución sea generada y/o

aceptada fuera de este dominio. Dentro del área de optimización estos límites o acotamientos son conocidos como *restricciones*, las cuales pueden ser tan simples como $x_1 \geq -2$ o tan complejas como un conjunto de ecuaciones no-lineales.

2.4. Restricciones

Dada una función $f(x) : \mathbb{R}^D \to \mathbb{R}$, con D grados de libertad, una restricción es una limitación del grado de libertad que se tiene al momento de proveer una solución, y dependiendo de la naturaleza del problema o de las restricciones, la optimización puede volverse sencilla o compleja.

En general, para la optimización las restricciones son condiciones que una solución a un problema debe satisfacer [Leffingwell, 2011]. En la literatura se tienen generalizados dos grupos de restricciones: las restricciones de igualdad y las restricciones de desigualdad ($\mathcal{E}$ y $\mathcal{I}$ respectivamente), mencionadas en la Ecuación (1.1). De acuerdo con la definición dada por la Ecuación (1.2), Ω está compuesta por estos dos conjuntos de restricciones. Sin embargo, la geometría del conjunto factible está definida la linealidad o no-linealidad de las restricciones.

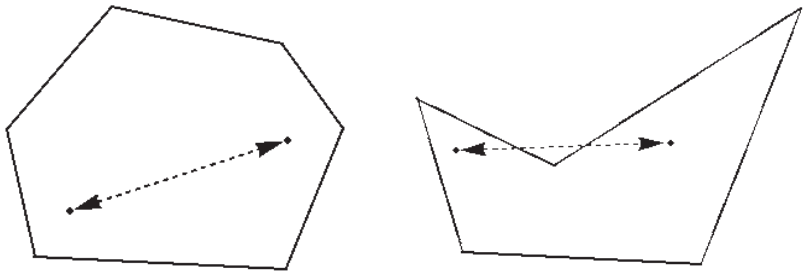
Cuando $f(x)$ está solamente sujeta a restricciones lineales, el conjunto factible es convexo. Por lo tanto a continuación se define el término convexidad y se menciona que si Ω es definido mediante la intersección de restricciones lineales, Ω tiene la propiedad de ser convexo.

2.4.1. Convexidad en las restricciones lineales

Un conjunto convexo es una colección de puntos tales que, para cada par de puntos de la colección, el segmento rectilíneo completo que une estos dos puntos también está en la colección [Hillier y Lieberman, 2006]. Por ejemplo en la Figura 2.5(a) el segmento rectilíneo que une a los dos puntos, se encuentra dentro del conjunto, y en la Figura 2.5(b), existe un pequeño segmento que no es parte del conjunto.

El conjunto de soluciones de una restricción lineal de tipo desigualdad, constituye una parte del espacio $\mathbb{R}^D$ y la selección de la parte del espacio está definida por el signo de la desigualdad. En consecuencia, una desigualdad es convexa porque cualquier par de puntos que se encuentre en la parte del espacio seleccionada, todo el segmento de línea que los une forma parte del espacio [Rockafellar y Wets, 1998].

Por lo tanto, cuando se tiene un conjunto de desigualdades, el conjunto de soluciones corresponde a la intersección de las partes de los espacios definidos por los signos de cada desigualdad. El conjunto resultante es convexo porque de acuerdo con la definición mostrada arriba, todas las desigualdades son convexas y la intersección de un espacio convexo con otro resulta es un espacio convexo [Solodovnikov, 1980, pág. 53].



(a) **(b)**

Figura 2.5: a) Conjunto Convexo, b) Conjunto no-convexo

De igual manera, si se tiene alguna restricción lineal del tipo igualdad, cualquier segmento de línea que una a dos puntos del conjunto, forma parte del conjunto, por lo tanto la restricción es convexa [Hillier y Lieberman, 2006]. Cuando se tiene un conjunto de restricciones de este tipo, la solución o el conjunto de soluciones está dado por la intersección de las igualdades y el resultado también es un conjunto convexo.

2.4.2. Geometría en las restricciones lineales

Geométricamente las restricciones lineales representan líneas, planos o hiper-planos en $\mathbb{R}^2$, $\mathbb{R}^3$ ó $\mathbb{R}^n$, respectivamente [de Berg et al., 2010].

En términos generales, cuando Ω está definido por la intersección de restricciones lineales, el conjunto convexo resultante se le conoce como poliedro convexo [Solodovnikov, 1980]. Por ejemplo, en la Figura 2.6, se encuentran algunos ejemplos de la intersección de restricciones lineales. El espacio que cumple con la desigualdad está indicado por la parte sombreada. En la Figura 2.6(a) se muestra la intersección de seis restricciones que comparten una región en común. La intersección también puede ser un segmento de línea, o un punto como en la Figura 2.6(b), o puede ser vacía como en la Figura 2.6(c).

Este enfoque geométrico puede ser de gran ayuda debido a que el manejo de restricciones realizado en este trabajo utiliza la envoltura convexa de Ω.

Dado un conjunto de puntos en el plano, su envoltura convexa está definida por el polígono convexo de área mínima que cubre todos los puntos (esto es, todos los puntos están dentro del polígono). Esta definición también se puede generalizar para puntos en el espacio o para puntos en alguna dimensión mayor, cambiando el polígono por un poliedro [Guillen, 2007].

(a) **(b)** **(c)**

Figura 2.6: Ejemplos de la intersección de restricciones lineales.

2.5. Generación de números aleatorios dentro de la región de factibilidad

Como se menciono en la Sección 1.3, la motivación principal de este trabajo es generar puntos que siempre se encuentren dentro de Ω. Por razones de simplicidad, en las siguientes secciones consideraremos que el término números aleatorios puede adquirir el significado de puntos aleatorios. Dado un conjunto de puntos $q = \{q_0, q_1, \ldots, q_{N-1}\}$ como se puede observar en la Figura 2.7(a), nos interesa generar un punto p dentro su envoltura convexa. El caso más simple que podemos analizar es cuando se tienen dos puntos. Tomamos los puntos q_0 y q_1 y decimos que Ω es una línea trazada desde el punto q_0 al punto q_1 como se observa en la Figura 2.7(b). Utilizando la ecuación paramétrica de la línea, podemos generar un punto dentro de Ω mediante la Ecuación (2.5).

$$p = q_0 + \alpha(q_1 - q_0) \tag{2.5}$$

donde $\alpha \sim U(0, 1)$. Si $\alpha = 0$ entonces $p_0 = q_0$, por otro lado si $\alpha = 1$ entonces $p_0 = q_1$.

El mismo principio utilizado en la Ecuación (2.5) puede ampliarse para N puntos. Consideremos un caso más general donde se desea generar un punto p dentro del conjunto de puntos q cuando $N = 5$. Aplicando la Ecuación (2.5) y desarrollando se tiene que el punto p_0 puede ser generado mediante la Ecuación (2.6).

$$\begin{aligned} p_0 &= q_1 + \alpha_0(q_0 - q_1) \\ &= \alpha_0 q_0 + (1 - \alpha_0)q_1 \\ &= a_0 q_0 + a_1 q_1 \end{aligned} \tag{2.6}$$

Este punto p_0 servirá como referencia para posteriormente generar p_1, como se demuestra en la Figura 2.7(c) y en la Ecuación (2.7).

$$\begin{aligned} p_1 &= q_2 + \alpha_1(p_0 - q_2) \\ &= \alpha_0\alpha_1 q_0 + \alpha_1(1 - \alpha_0)q_1 + (1 - \alpha_1)q_2 \\ &= a_0 q_0 + a_1 q_1 + a_2 q_2 \end{aligned} \tag{2.7}$$

de la cual podemos obtener la Ecuación (2.8), y observar el resultado en la Figura 2.7(d).

$$\begin{aligned} p_2 &= q_3 + \alpha_2(p_1 - q_3) \\ &= \alpha_0\alpha_1\alpha_2 q_0 + \alpha_1\alpha_2(1-\alpha_0)q_1 + \alpha_2(1-\alpha_1)q_2 + (1-\alpha_2)q_3 \\ &= a_0 q_0 + a_1 q_1 + a_2 q_2 + a_3 q_3 \end{aligned} \tag{2.8}$$

y finalmente en la Ecuación (2.9) podemos observar que para generar el punto p_3 se utilizan todos los puntos q, de tal manera que $p = p_3$, como en la Figura 2.7(e).

$$\begin{aligned} p_3 &= q_4 + \alpha_3(p_1 - q_4) \\ &= \alpha_0\alpha_1\alpha_2\alpha_3 q_0 + \alpha_1\alpha_2\alpha_3(1-\alpha_0)q_1 + \alpha_2\alpha_3(1-\alpha_1)q_2 + \alpha_3(1-\alpha_2)q_3 + (1-\alpha_3)q_4 \\ &= a_0 q_0 + a_1 q_1 + a_2 q_2 + a_3 q_3 + a_4 q_4 \end{aligned} \tag{2.9}$$

De acuerdo al proceso descrito mediante las Ecuaciones (2.6), (2.7), (2.8) y (2.9) podemos observar que por cada q_i un nuevo a_i es calculado y se deduce que se puede generar un punto p utilizando N puntos como los vértices de la envoltura convexa de Ω mediante la ecuación (2.10).

$$p = \sum_{i=0}^{N-1} a_i q_i \tag{2.10}$$

Sin embargo, durante el proceso, hay que notar que los valores de a_i están formados por distintos valores aleatorios α_i, los cuales contribuyen a determinar la ubicación del punto p. En la Tabla 2.2 podemos observar los valores que toman los a_i en un ejemplo con cinco vértices. Nótese que para cada renglón de la tabla, la suma de las variables a_i siempre es igual a uno. Para demostrar esta propiedad, consideremos el caso cuando $N = 5$; la sumatoria de los a_i se desarrolla en la Ecuación (2.11).

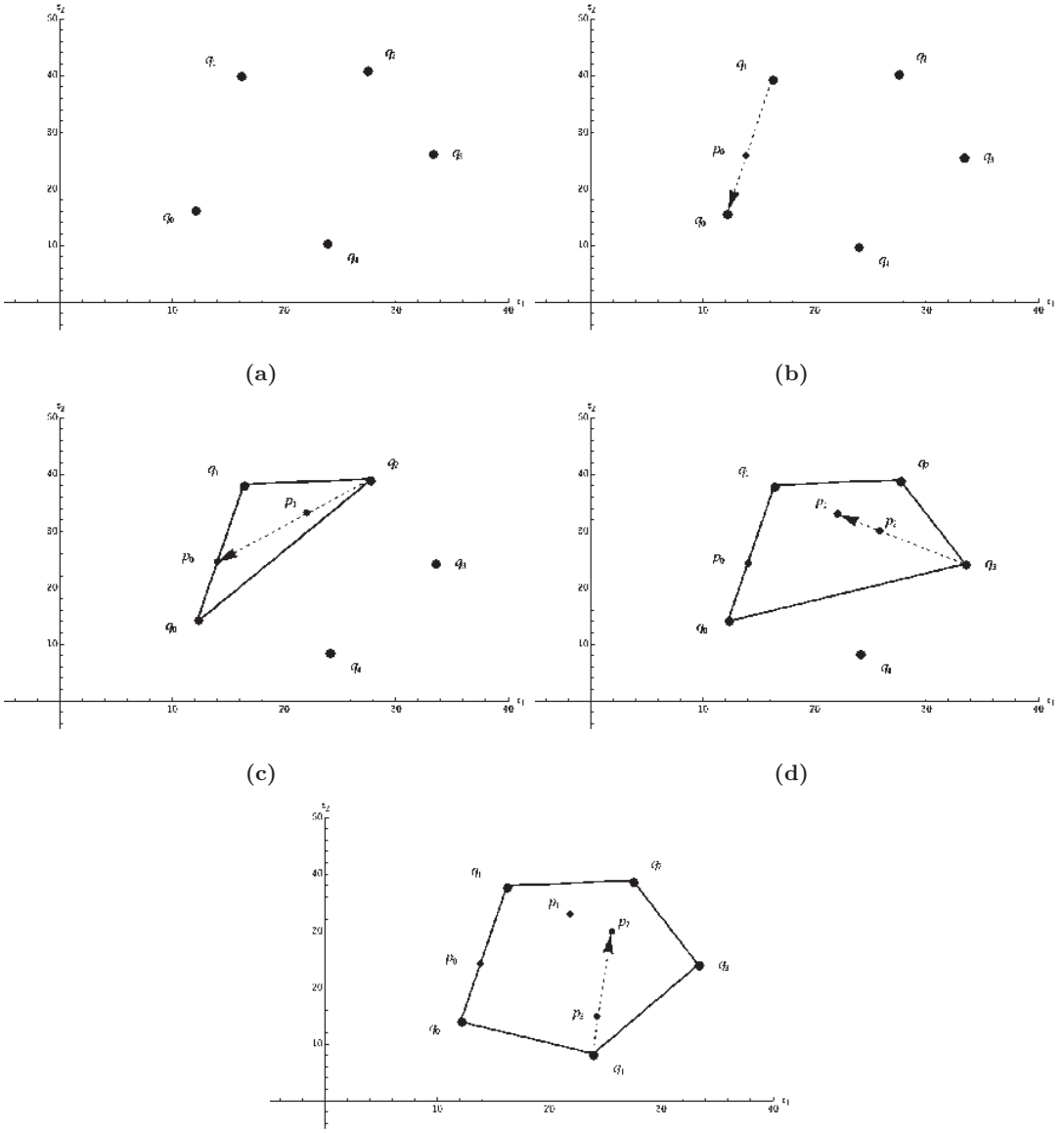


Figura 2.7: Generación de puntos dentro de un conjunto de puntos. a) envoltura convexa de un conjunto de puntos b) usando (2.6), c) usando (2.7), d) usando (2.8) y e) usando (2.9)

$$\begin{aligned}
\sum_{i=0}^{N-1} a_i &= \alpha_0\alpha_1\alpha_2 + \alpha_1\alpha_2(1-\alpha_0) + \alpha_2(1-\alpha_1) + (1-\alpha_2) \\
&= \alpha_0\alpha_1\alpha_2 + \alpha_1\alpha_2 - \alpha_0\alpha_1\alpha_2 + \alpha_2 - \alpha_1\alpha_2 - \alpha_2 + 1 \\
&= \alpha_0\alpha_1\alpha_2 + \alpha_1\alpha_2 + \alpha_2 - \alpha_0\alpha_1\alpha_2 - \alpha_1\alpha_2 - \alpha_2 + 1 \\
&= \cancel{\prod_{i=0}^{N-1} \alpha_i + \prod_{i=1}^{N-1} \alpha_i + \prod_{i=2}^{N-1} \alpha_i} - \cancel{(\prod_{i=0}^{N-1} \alpha_i + \prod_{i=1}^{N-1} \alpha_i + \prod_{i=2}^{N-1} \alpha_i)} + 1 \\
&= 1
\end{aligned} \tag{2.11}$$

La sumatoria mostrada en la Ecuación (2.11) siempre se cumple sin importar el número de vértices utilizados en la generación del punto p. Permitiendo que cualquier combinación de los valores a_i genere un punto dentro de la envoltura convexa de Ω.

Tabla 2.2: Sumatoria de las a_i para cada N.

N	a_0	a_1	a_2	a_3	a_4	Suma
0	1	0	0	0	0	1
1	α_0	$1-\alpha_0$	0	0	0	1
2	$\alpha_0\alpha_1$	$\alpha_1(1-\alpha_0)$	$1-\alpha_1$	0	0	1
3	$\alpha_0\alpha_1\alpha_2$	$\alpha_1\alpha_2(1-\alpha_0)$	$\alpha_2(1-\alpha_1)$	$1-\alpha_2$	0	1
4	$\alpha_0\alpha_1\alpha_2\alpha_3$	$\alpha_1\alpha_2\alpha_3(1-\alpha_0)$	$\alpha_2\alpha_3(1-\alpha_1)$	$\alpha_3(1-\alpha_2)$	$1-\alpha_3$	1

De manera que para asegurar que el punto p se encuentre dentro de Ω es necesario cumplir con la Ecuación (2.11), donde a_i está compuesto de distintos pesos aleatorios α_i que indican la contribución de cada vértice y N es la cantidad de puntos o vértices de la envoltura convexa de Ω.

2.6. Generación de números aleatorios con la forma de la región de factibilidad

Como se vio en la sección anterior, es posible generar un punto p dentro de Ω mediante la Ecuación (2.10) si y solo si se cumple la Ecuación (2.11). Sin embargo, utilizando este enfoque, la probabilidad de generar un punto muy cercano a los vértices o en un vértice

de la envoltura convexa que define a Ω, es muy pequeña. Esto se debe a que los valores de a_i en la Ecuación (2.11), están compuestos de valores aleatorios α_i, los cuales de acuerdo al Teorema del Límite Central [Fischer, 2010], tienden a aproximarse a una distribución gaussiana. En la Figura 2.8 se puede observar que es más probable generar puntos en la media del conjunto q, y menos probable generarlos cerca de las aristas de q.

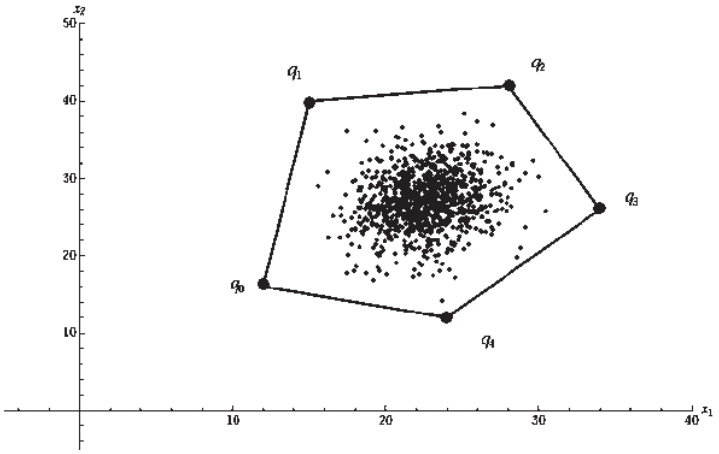


Figura 2.8: Puntos generados utilizando las Ecuaciones (2.10) y (2.11).

Nuestra propuesta consiste básicamente en generar puntos con una distribución normal que cubra Ω. Así la media μ estará en el centro de la región y la matriz de covarianza Σ tendrá la forma de un elipsoide alrededor de Ω, tal como se muestra en la Figura 2.9. Para hacer el cálculo de los puntos con la forma de la región de factibilidad, asumiremos que tenemos el conjunto de vértices q, los cuales corresponden a la envoltura convexa de Ω. La media y la covarianza muestral de esta distribución la calcularemos a partir de q, utilizando las Ecuaciones (2.12) y (2.13) respectivamente.

$$\mu = \frac{1}{N}\sum_{i=0}^{N-1} q_i \tag{2.12}$$

$$\Sigma = \frac{1}{N}\sum_{i=0}^{N-1} (q_i - \mu)(q_i - \mu)^T \tag{2.13}$$

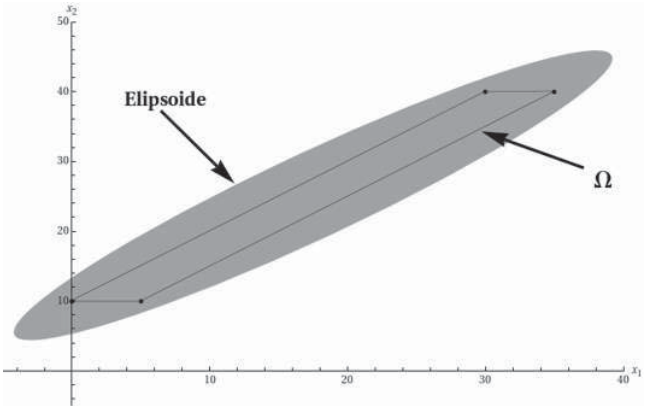


Figura 2.9: Simulando la forma de la región de factibilidad mediante un elipse

Como se vio en la Sección 2.2.2, en cualquier plataforma resulta simple hacer la generación de números aleatorios con media cero y varianza unitaria. Sin embargo el problema radica en la generación de números aleatorios con una media y una covarianza distintas.

Consideremos un conjunto de N números $z = [z_1, z_2, z_3, \ldots, z_k, \ldots, z_N]^T$ tal que $z_k = [z_{k,1}, z_{k,2}, z_{k,3}, \ldots, z_{k,j}, \ldots, z_{k,D}]^T$, donde $z_{k,j} \sim N(0,1)$ y decimos que $z_{k,j}$ es el j-ésimo componente del k-ésimo número. Debido a que para z_k la media es cero y la varianza unitaria, entonces tenemos que el valor esperado de z_k es calculado con la Ecuación (2.14) y el valor esperado de $z_k z_k^T$ es calculado por la Ecuación (2.15)

$$E[z] = \frac{1}{N}\sum_{k=1}^{N} z_k = 0 \tag{2.14}$$

$$E[zz^T] = \frac{1}{N}\sum_{k=1}^{N} z_k z_k^T = I \tag{2.15}$$

donde I es la matriz identidad.

Como queremos generar puntos p_k con una media distinta a cero y una varianza distinta a uno, entonces dado un número z_k vamos a modificar su media y su covarianza. Comenzaremos modificando la covarianza por medio de una matriz A, de tal manera que $p_k = Az_k$, y por la propiedad de la covarianza podemos calcular su valor esperado mediante

la Ecuación (2.16).

$$\begin{aligned} E[pp^T] &= \frac{1}{N}\sum_{k=1}^{N} p_k p_k^T \\ &= \frac{1}{N}\sum_{k=1}^{N} Az_k(Az_k)^T \\ &= \frac{1}{N}\sum_{k=1}^{N} Az_k z_k^T A^T \\ &= A\Big[\frac{1}{N}\sum_{k=1}^{N} z_k z_k^T\Big]A^T \qquad I \\ &= AA^T = \Sigma \end{aligned} \tag{2.16}$$

Debido a las propiedades de la matriz de covarianza Σ (positiva semidefinida, simétrica y cuadrada) y que $AA^T = \Sigma$, entonces podemos aplicar el cálculo de valores y vectores propios tal que $\Sigma = R\Lambda R^T$, donde R es una matriz de rotación y $\Lambda = diag[\lambda_0, \lambda_1, \cdots, \lambda_{D-1}]$ (*Ver apéndice A*). De manera similar, Λ puede ser sustituido por LL^T y por lo tanto $L = diag[\sqrt{\lambda_0}, \sqrt{\lambda_1}, \cdots, \sqrt{\lambda_{D-1}}]$; entonces, el valor de A lo podemos sustituir con el producto de las matrices R y L.
Mediante el cálculo de valores y vectores propios y las sustituciones realizadas, podemos obtener la Ecuación (2.18).

$$\begin{aligned} AA^T &= \Sigma \\ &= R\Lambda R^T \\ \Sigma &= RLL^T R^T \end{aligned} \tag{2.17}$$

de manera que $A = RL$ y por lo tanto, ahora $p_k = RLz_k$.

Una vez que se ha modificado la covarianza de z_k, ahora solo basta modificar su media, añadiendo un valor b al número p_k tal que $p_k = RLz_k + b$. Entonces, por la propiedad de la media podemos calcular el valor esperado con la Ecuación (2.18).

$$\begin{aligned} E[p] &= \frac{1}{N}\sum_{k=1}^{N} p_k \\ &= \frac{1}{N}\sum_{k=1}^{N}(Az_k + b) \\ &= b + A\frac{1}{N}\sum_{k=1}^{N} z_k \quad \text{(el término } \sum z_k \text{ tiende a } 0\text{)} \\ &= b \end{aligned} \tag{2.18}$$

El desplazamiento de la media de z_k puede realizarse añadiendo el valor b a la media μ_{z_k}. tal y como se menciona en la Ecuación (2.18), de manera que estamos modificando el valor de la media del conjunto por uno diferente. Por ejemplo, consideremos un conjunto de N datos con $\mu_{z_k} = [27, 34]^T$ y $N = 500$. Para lograr la modificación de la media añadiremos el vector $[20, 50]^T$ a cada uno de los números z_k de tal manera que el valor de μ_{z_k} será alterado, pero la covarianza del conjunto se mantendrá igual. En la Figura 2.10 podemos observar como el conjunto de datos cambia su media de $\mu_{z_k} = [27, 34]^T$ a $\mu_{z_k} = [47, 84]^T$.

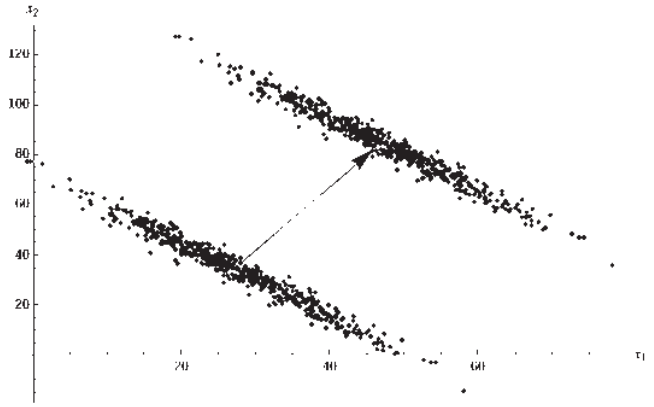


Figura 2.10: Cambio de la media de un conjunto de datos a una media diferente

Mediante las Ecuaciones (2.16) y (2.18) podemos obtener una fórmula para calcular un número con distribución normal $p_k \sim N(\mu, \Sigma)$ a partir de un número aleatorio z_k si

aplicamos la Ecuación (2.19).

$$p_k = RLz_k + \mu \tag{2.19}$$

La ecuación (2.19) garantiza tener un número aleatorio dentro del elipsoide de la distribución gaussiana, sin embargo, como se puede observar en la Figura 2.9, la superficie del elipsoide es mayor a la superficie de Ω, de manera que no podemos garantizar que el punto p se encuentre dentro del área de interés.

2.7. Validación de los números aleatorios generados

De acuerdo con la Sección 2.6, podemos generar un punto p con una media y una covarianza dada, sin embargo, existe cierta probabilidad de que p sea generado fuera de Ω, por lo tanto, es necesario determinar si p es un punto factible o no-factible.

Para validar si p se encuentra dentro de Ω, se desarrollan las Ecuaciones (2.10) y (2.11) para obtener las Ecuaciones (2.20) y (2.21). Podemos representar ambas ecuaciones mediante un producto matricial, el cual representa un sistema de ecuaciones de la forma $Qa = P$ y está dado por la Ecuación (2.22).

$$a_0q_0 + a_1q_1 + a_2q_2 + \cdots + a_{N-1} = p \tag{2.20}$$

$$a_0 + a_1 + a_2 + \cdots + a_{N-1} = 1 \tag{2.21}$$

$$\begin{bmatrix} q_0 & q_1 & q_2 & \cdots & q_{N-1} \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{N-1} \end{bmatrix} = \begin{bmatrix} p \\ 1 \end{bmatrix}$$

$$Qa = P \tag{2.22}$$

Dado un conjunto de vértices q y un punto p se desea encontrar los valores a_i con los cuales el punto p fue creado. En otras palabras, se desea encontrar la solución al sistema representado en la Ecuación (2.22).

El problema surge de la siguiente manera. La matriz Q es una matriz de $m \times n$ donde n es el número de columnas y m es el número de filas, por lo cual se espera que la solución de $Qa = P$ sea inconsistente [Strang, 1988, pág. 154].

Probablemente no existe una alternativa de que a se ajuste perfectamente a los datos de P, y el problema es escoger a tal que se minimice un error $\mathbb{E} = ||Qa - P||$ [Strang, 1988]. La mejor manera de ajustar a a los datos de P es mediante el método de mínimos cuadrados (*Ver apéndice B*), de manera que la solución de (2.22) por mínimos cuadrados puede calcularse con (2.23).

$$\begin{aligned} Qa &= P \\ a &= [Q^T Q]^{-1} Q^T P \end{aligned} \tag{2.23}$$

La Ecuación (2.23) resuelve los valores a_i con los cuales el punto p fue generado y de acuerdo con la Ecuación (2.11) el punto p estará dentro de la envoltura convexa de Ω si y solo si, los valores de a cumplen con $0 \leq a_i \leq 1$ y si $\sum_{i=1}^{N} = 1$. A este procedimiento se le conoce como cálculo de las coordenadas baricéntricas [Lawson, 1986].

El Algoritmo 1 resume todos los pasos descritos en las Secciones 2.6 y 2.7 para generar números aleatorios dentro de Ω, este algoritmo realiza el cálculo de puntos utilizando la envoltura convexa de Ω y capturando su forma geométrica en el espacio. Su principal característica es asegurar que el punto generado siempre se encuentre dentro de Ω, lo cual facilita la minimización de una función porque permite resolver el problema como si fuese libre de restricciones.

En resumen, para un conjunto factible Ω delimitado por el conjunto de vértices

$q = \{[0, 10], [30, 40], [35, 40], [5, 10]\}$, se realiza una comparación en la Figura 2.11, en la cual se pueden observar las tres distintas maneras de generar números aleatorios dentro de Ω que se vieron en este capítulo.

Algoritmo 1 NGFRS (Number Generation Feasible Region Shape)

Entrada: $q = \{q_0, q_1, \cdots, q_{N-1}\}$

Salida: Un conjunto de números aleatorios p

–Calcular la media μ y la matriz de covarianza Σ

mediante la Ecuación (2.12) y (2.13)

–Calcular la descomposición en vectores y valores propios $\Sigma = R\Lambda R^T$

–Calcular $L = diag[\sqrt{\lambda_0}, \sqrt{\lambda_1}, \cdots, \sqrt{\lambda_{N-1}}]$ a partir de Λ

–**Para** $k = 0, 1, 2, \cdots$

–Generar $z_{k,j} \sim N(0, 1)\ \forall j \in [1, \ldots, D]$

–Calcular $p_k = RLz_k + \mu$

–Calcular los valores a resolviendo la Ecuación (2.23)

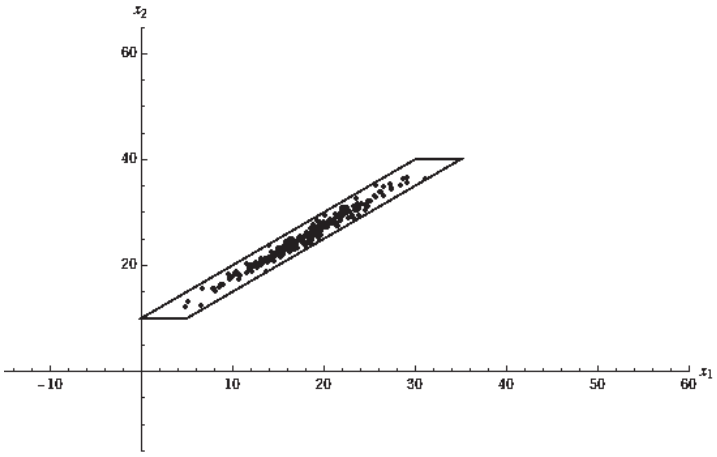
–**Si** $a_j < 0$ ó $a_j > 1 \qquad j = 0, 1, 2, \cdots$

Rechazar el número y generar uno nuevo.

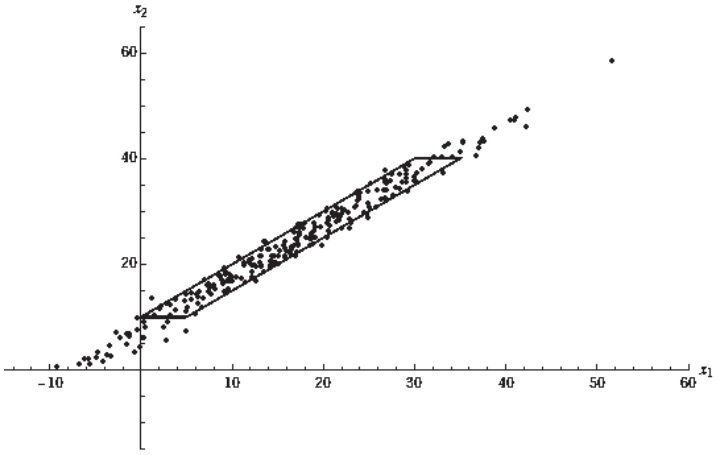
–**Si** $\sum_{j=0}^{D-1} a_j \neq 1$

Rechazar el número y generar uno nuevo.

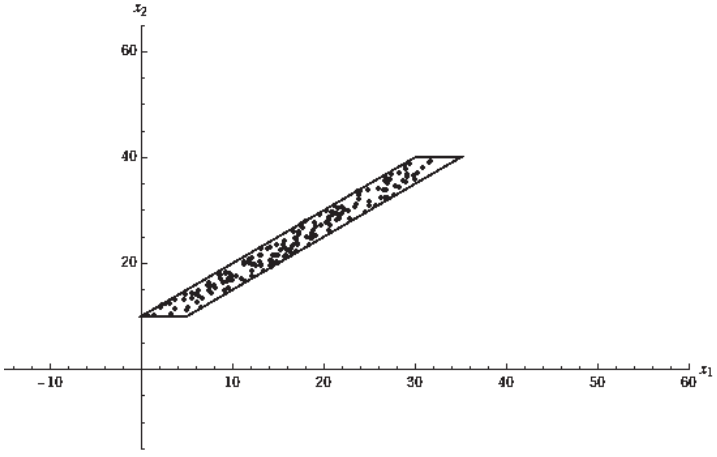
La Figura 2.11(a) muestra números aleatorios generados mediante la Ecuación (2.10). Debido a que la suma de los pesos que influyeron para la creación de estos números sigue una distribución normal, la mayoría de los puntos se encuentran en el centro de Ω. En la Figura 2.11(b) se observa lo que sucede cuando un conjunto de puntos es generado por la Ecuación (2.19). Aquí se puede apreciar que los puntos generados se encuentran dentro de un elipsoide que simula la forma geométrica de Ω, sin embargo existen puntos que se encuentran fuera de Ω. Finalmente, en la Figura 2.11(c) fue aplicado el Algoritmo 1, este algoritmo descarta los puntos que fueron generados fuera de Ω, dejando solamente puntos factibles.



(a)



(b)



(c)

Figura 2.11: Generación de números aleatorios. a) usando (2.10), b) usando (2.19), c)usando el Algoritmo 1

2.8. Conclusiones del capítulo

El primer enfoque de este capítulo es dar a conocer la importancia que tiene la generación de números aleatorios sobre la toma de decisiones en aquellos algoritmos que utilizan la incertidumbre como herramienta para resolver problemas. Así mismo se toma consideración de la diferencia que existe entre generar un punto con una distribución u otra. Este análisis ayuda a establecer un nuevo método de generación de números aleatorios llamado NGFRS el cual ayudará a resolver el problema planteado en el Capítulo 1, debido a que servirá como complemento a la heurística de búsqueda utilizada por el algoritmo de optimización a utilizar en el capítulo siguiente.

El método que se propone en la Sección 2.7, captura la forma de la región de factibilidad y considera el análisis de explorar las propiedades geométricas que puede tener una región factible, dando preferencia a las zonas con mayor espacio de búsqueda.

Una de las partes más importantes de este capítulo es la validación de que un punto generado se encuentre dentro de Ω, independientemente del método utilizado para generarlo es posible identificar si es factible o no-factible. El funcionamiento de este método está basado en la creación de un sistema de ecuaciones que contiene el punto generado y los vértices de la envoltura convexa de Ω. La solución que se desea encontrar a través de este sistema de ecuaciones es determinar los pesos con los cuales el punto fue generado.
Sin embargo, debido a que el sistema de ecuaciones no contiene la misma cantidad de incógnitas y de ecuaciones, se espera que no se encuentre una solución exacta, por lo tanto su solución está basada en una aproximación mediante mínimos cuadrados.

Como punto final de las conclusiones de este capítulo se señala que aunque la cantidad de puntos a validar puede llegar ser muy grande y que para cada nuevo número generado es necesario resolver un sistema de ecuaciones diferente, las técnicas de programación empleadas permiten una validación rápida.

Capítulo 3

Evolución Diferencial

3.1. Introducción

En el capítulo anterior, se estudiaron distintos modelos que permiten la generación de números aleatorios dentro de la región de factibilidad. Principalmente, se desarrolló un algoritmo que utiliza la envoltura convexa de Ω para realizar la distinción entre un punto factible y no factible. Esta aportación puede ser incorporada a cualquier algoritmo de optimización estocástico para validar las soluciones encontradas. Por lo tanto, solamente falta elegir un algoritmo de optimización estocástico adecuado que permita resolver el problema planteado en la Sección 1.1, el cual se muestra en la Ecuación (3.1).

$$min \;\; f(x) \;\; \text{s.a} \begin{cases} c_i(x) \leq 0 \quad para\; i \in \mathcal{E} \\ c_i(x) = 0 \quad para\; i \in \mathcal{I} \end{cases} \tag{3.1}$$

Del campo de la optimización, existen dos enfoques posibles para resolver problemas: algoritmos deterministas y algoritmos estocásticos.

A continuación vamos a discutir un poco acerca de las características de los métodos deterministicos y de los métodos estocásticos, así como su funcionalidad para encontrar el óptimo global en problemas sujetos a restricciones.

3.2. Algoritmos de optimización

3.2.1. Algoritmos deterministas

Los algoritmos deterministas comienzan con una estimación inicial de la solución y generan una secuencia de estimaciones más precisas (iteración) hasta que alcanzan un óptimo. La estrategia usada para moverse de una iteración a la otra distingue un algoritmo de otro. La mayoría de las estrategias empleadas por estos algoritmos utilizan información de los valores de la función objetivo, de las restricciones o de la primera y segunda derivada de la función objetivo [Nocedal y Wright, 1999, pág. 8].

La ventaja de estos algoritmos es que son extremadamente efectivos dentro de su alcance, por lo tanto se dice que la optimización realizada por la mayoría de los algoritmos deterministicos depende completamente de su punto inicial [Bagdadhi, 2007]. Para problemas sin restricciones varios métodos tales como los de Newton, Quasi-Newton y Dirección Conjugada son utilizados debido a su rápida convergencia y precisión [Chong y Zak, 2001].

Existen algunos algoritmos deterministas que fueron diseñados para encontrar el óptimo global. Los más exitosos son los métodos de rama y frontera (*Branch and Bound*-**B&B**) [Lawler y Wood, 1966]. Estos métodos funcionan dividiendo sistemáticamente Ω en sucesivas pequeñas regiones. Para cada una de estas regiones las soluciones locales son encontradas y se asume que el óptimo global es el valor más pequeño entre todas las soluciones locales. Otros métodos incluyen el método de bisección multidimensional (*Multidimensional bisection*-**MB**)[Zhang y Baritompa, 1993] y el método de optimización global Lipschitz desarrollado por János Pinter [Pinter, 1996].

Desafortunadamente, los algoritmos deterministas no son capaces de resolver cualquier problema. Existen algunas situaciones donde la solución que proveen no es la mejor o no son capaces de resolver el problema en absoluto. Por lo tanto, es necesario tomar en cuenta algunas consideraciones respecto a los algoritmos deterministas.

Son sensibles al punto inicial que se les proporcione. Si el punto inicial no está lo suficientemente cerca del óptimo global, solo se asegurará una convergencia al óptimo local más cercano. Porque el algoritmo intenta entonces moverse, a partir de este punto, en una dirección a través de la región factible, de tal forma que el valor de la función objetivo mejore.

Es necesario calcular el gradiente($\nabla f(x)$) y el hessiano($\nabla^2 f(x)$). Requieren la existencia de derivadas continuas en la función objetivo y para muchos problemas de optimización el calculo de las primeras derivadas puede tomar más tiempo de cómputo que la evaluación de la función objetivo, debido a que $\nabla f(x)$ es un vector de tamaño N y calcular las segundas derivadas puede tomar mucho más tiempo y requerir una gran cantidad de memoria, debido a que $\nabla^2 f(x)$ es una matriz de $N \times N$ y calcularlo requiere tiempo $O(N^2)$.

A pesar de estas consideraciones, los algoritmos deterministas han demostrado ser una herramienta confiable en la optimización.

3.2.2. Algoritmos estocásticos

Los algoritmos estocásticos (en ocasiones llamados algoritmos de heurística) representan una amplia clase de estrategias computacionales que mediante búsquedas inteligentes tratan de encontrar el óptimo global [Spall, 2004].

Regularmente están inspirados en procesos físicos, evolución natural y eventos estocásticos. Algunas de las características de estos algoritmos a diferencia de sus contrapartes deterministas incluyen: ser independientes del problema, ser relativamente sencillos de implementar y solamente requerir realizar evaluaciones de la función objetivo para su funcionamiento [Bagdadhi, 2007].

La diferenciabilidad, linealidad o continuidad de la función objetivo son irrelevantes, porque la búsqueda está guiada por mecanismos que no rebasan en estos conceptos. Estos algoritmos han probado empíricamente ser una estrategia de búsqueda del óptimo global efectiva y práctica, porque incrementan el rango de resolución de problemas. Los algoritmos estocásticos más populares tienen su inspiración en los principios de la evolución

biológica, y a partir de estos fundamentos surgen los EAs como algoritmos de búsqueda y optimización, que inspirados en diversos mecanismos biológicos, intentan reproducir las características de robustez y simplicidad existentes en la naturaleza para evolucionar hacia mejores soluciones [Goldberg, 1989].

La Tabla 3.1 muestra las ventajas y desventajas que existen entre los algoritmos determinísticos y los algoritmos estocásticos.

Tabla 3.1: Tabla comparativa entre optimización deterministica y optimización estocástica

	Deterministicos	Estocásticos
Ventajas	• Rápida convergencia. • Óptimo local garantizado. • Buena precisión de la solución.	• Independencia del problema. • Implementación sencilla. • Solo se necesitan evaluaciones continuas de la función objetivo para su funcionamiento.
Desventajas	• Dependientes del problema. • Se requiere un punto inicial para iniciar la búsqueda. • El cálculo del gradiente y del hessiano puede tomar mucho tiempo de cómputo y mucha memoria.	• No existe garantía de convergencia. • Es necesario ajustar parámetros que guíen la búsqueda. • La múltiple evaluación de la función objetivo puede llevar a estos algoritmos a una lenta convergencia.

Algoritmos evolutivos

Las reglas de la evolución son notablemente simples. Las especies evolucionan por medio de la selección natural. Esto es, la conservación de las diferencias y variaciones individualmente favorables y la destrucción de las que son perjudiciales. Los individuos que tienen ventaja sobre otros, tendrán más probabilidades de sobrevivir y procrear su especie.

Los hijos heredan características de sus padres, por lo que cada nueva generación tiene individuos semejantes a los mejores individuos de la generación anterior. Además del proceso de selección natural, en la naturaleza ocasionalmente se producen mutaciones al azar. Algunas de estas mutaciones conducirán a nuevos y mejores individuos. De esta manera, se introducen alteraciones aleatorias y luego son propagadas a generaciones futuras. Por el contrario, las características negativas introducidas por la mutación tenderán a desaparecer pues conducen a individuos pobremente adaptados. Los individuos irán evolucionando gradualmente hasta que el periodo de generaciones finalice, dejando solamente a los individuos mas adaptados al entorno [Sivanandam y Deepa, 2008].

Este proceso evolutivo es aplicado a los EAs, y generalmente se distinguen cuatro etapas del proceso: creación y evaluación de individuos, selección de los mejores individuos, evolución mediante cruza ó mutación y un período de generaciones. Cada una de estas etapas es descrita a continuación.

1.- Se genera un conjunto de posibles soluciones para el problema que se intenta resolver. Cada una de estas soluciones representa un *individuo* y todo el conjunto representa una población. A cada individuo de la población se le asigna un valor de adaptabilidad de acuerdo a que tan buena sea como solución del problema en cuestión.

2.- El valor de adaptabilidad servirá para que un proceso de selección, que favorece a los individuos con mejor desempeño, tome pares de individuos de la población para ser sometidos a un operador de cruzamiento.

3.- El operador de cruzamiento implementa alguna forma de intercambio entre las cadenas que conforman los individuos seleccionados (*padres*), para generar nuevas soluciones (*hijos*) las cuales formarán una nueva población. Por otro lado el operador de mutación selecciona aleatoriamente algún individuo de la población para alterar la cadena que lo conforma y proporcionar diversidad genética en la población.

4.- Los procedimientos de creación y evaluación de individuos, selección y evolución mediante cruzamiento y mutación, se repiten de forma iterativa hasta que algún criterio de parada se cumpla. A cada iteración se le denomina comúnmente generación. Entonces, se dice que un nuevo conjunto de soluciones se crea en cada generación por medio de la selección de los mejores individuos de la generación actual y el cruzamiento de los mismos.

Este proceso análogo, llevado a cabo en entornos naturales, proporciona a los EAs las características necesarias para un comportamiento robusto que le permite ser utilizado para un amplio rango de problemas de optimización de funciones, optimización combinatoria, sistemas de aprendizajes de lógica difusa, aprendizaje de redes neuronales, entre otros [E. Zitzler, 2000].

Los EAs más conocidos incluyen a los GAs [Goldberg, 1989], Optimización mediante Enjambre de Partículas (*Particle Swarm Optimization*-**PSO**) [Trelea, 2003], Búsqueda Aleatoria Controlada(*Random Controled Search*-**RCS**) [Ali y Storey, 1994], Colonia de Hormigas (*Ant Colony*) [Dorigo y Caro, 1999] y DE [Storn y Price, 1995]. Para los fines de este trabajo nos centraremos particularmente en dos algoritmos, GA por su amplia utilización en la resolución de problemas y DE porque comparte diversas características con los GAs y porque ha probado empíricamente proveer mejores soluciones que GA. Por lo tanto, se dará una breve descripción del funcionamiento de ambos algoritmos y se enumerarán algunas diferencias entre ellos.

3.3. Algoritmos genéticos

Son un método de optimización y búsqueda basado en los principios genéticos y selección natural. Un GA permite que la población compuesta de distintos individuos evolucione bajo unas reglas de selección específica para maximizar la aptitud de los individuos, por ejemplo minimizar la función objetivo [Haupt y Haupt, 2004].

El método fue desarrollado por John H. Holland (1975) sobre el curso de los 60's

y los 70's, y finalmente popularizado por uno de sus estudiantes, David Goldberg, quien fuera el que resolvió un problema difícil para su trabajo de tesis que involucraba el control de una línea de transmisión de gas.

El núcleo de los GAs

Los pasos básicos que forman un GA se enlistan a continuación. Estos pasos son suficientemente generales para gobernar muchas (quizás todas) implementaciones modernas de GAs, incluyendo las que existen en software comercial. Por su puesto, el desempeño de los GAs típicamente depende en su mayoría de los detalles de su implementación, justo como con cualquier algoritmo de optimización evolutiva.

1. **Inicialización**

   Aleatoriamente se genera una población inicial con N_{pop} individuos y se evalua la aptitud de cada individuo.

2. **Selección de padres**

   Los padres son seleccionados de acuerdo a su aptitud, aquellos que en la evaluación tengan una mejor aptitud serán seleccionados con más frecuencia.

3. **Cruza**

   Para cada par de padres identificados en el paso dos, se realiza una cruza de los padres en un punto aleatoriamente escogido (quizás con una distribución uniforme) con probabilidad P_c. Si no existió la cruza (con probabilidad $1 - P_c$), entonces formar dos hijos que sean las copias exactas (clones) de ambos padres.

4. **Mutación y reemplazo**

   Si no se tiene cuidado, los GAs pueden converger de manera prematura en una región. Para evitar este problema, la rutina es forzada a explorar otras áreas, introduciendo cambios aleatorios a algunos de los parámetros.

   Mientras se mantiene a los mejores individuos de la generación (N_e), el reemplazo de la población se realizará en los restantes ($N_{pop} - N_e$) con la heurística del paso dos.

5. **Aptitud y Prueba de fin**

   Calcular los valores de aptitud para cada nueva población de N_{pop} individuos, y terminar el algoritmo si el criterio de paro se cumple. De no cumplirse este criterio se vuelve al paso dos.

Una vez terminado de afinar la codificación, la función objetivo, los parámetros de ajuste y un criterio de parada, un GA puede implementarse mediante el Algoritmo 2.

Algoritmo 2 Algoritmo simple de GA

Entrada: $f(x)$

Salida: x^*

–Generar la población inicial $P^{(0)}$ de forma aleatoria y hacer $g = 0$

Repetir

–Evaluar la aptitud de cada individuo en $P^{(g)}$

–Seleccionar padres de $P^{(g)}$ de acuerdo a su aptitud.

–Efectuar cruzamiento y mutación para producir la población $P^{(g+1)}$

–$g \leftarrow g + 1$

–Hasta alcanzar el criterio de parada

El Algoritmo 2 puede utilizarse para resolver problemas de optimización sin restricciones. Si se desea resolver problemas sujetos a restricciones será necesario aplicar alguna técnica que se capaz de manejarlas (como la validación propuesta en la Sección 2.7), debido a que cuando existen restricciones en el problema, los operadores evolutivos de los GAs son susceptibles a generar individuos fuera de Ω, causando una inconsistencia en el algoritmo y provocando una convergencia en una solución errónea.

3.4. Evolución diferencial

En 1995, Price y Storn propusieron un nuevo algoritmo evolutivo para lograr una optimización global, el cual fue llamado Evolución Diferencial (*Differential Evolution*-**DE**). El hecho de ser un método de fácil implementación y que tiene pocos parámetros para su

afinación, hicieron que el algoritmo se hiciera rápidamente popular [Storn y Price, 2005].

A diferencia de los EA's convencionales que dependen de una distribución de probabilidad predefinida para el proceso de mutación, DE utiliza las diferencias de pares de individuos escogidos aleatoriamente para su proceso de mutación. En consecuencia, las diferencias obtenidas pasan información topográfica a la función objetivo para mejorar el proceso de optimización y por lo tanto son capaces de proveer una capacidad más eficiente de optimización global [Lee y El-Sharkawi, 2008, pág. 171]. En esta sección se da una descripción de DE y se presenta como alternativa para la solución de problemas de optimización.

3.4.1. Inicialización

Como en casi todos los EAs, DE es un optimizador basado en la creación de una población, que ataca el problema del punto inicial esparciendo múltiples y aleatorios puntos iniciales en el espacio de búsqueda, como se puede observar en la Figura 3.1, mientras que las fronteras de los parámetros definen el dominio por el cual los individuos en esta población inicial son creados.

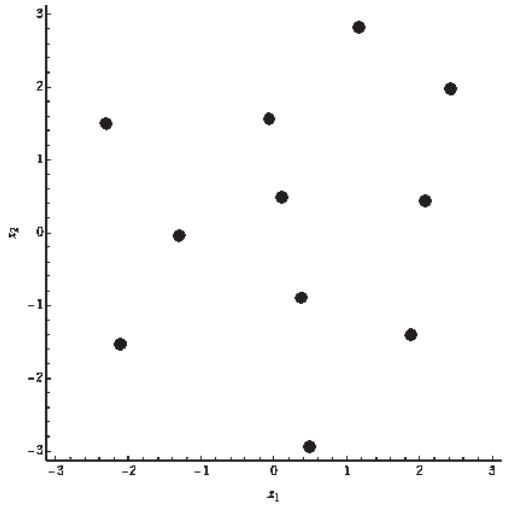


Figura 3.1: Inicialización del algoritmo DE.

La población actual $P^{(g)}$, se compone de los individuos $x_i^{(g)}$ que serán declarados como población inicial mediante la Ecuación (3.2); el índice $g = 0, 1, \ldots, g_{max}$ indica la generación a la cual un individuo pertenece. A cada individuo se le asigna un índice de la población i, el cual va desde 0 a N_{pop-1} y los parámetros dentro de los individuos se indízan con $j \in [0, \ldots, D-1]$.

$$\begin{gathered} P^{(g)} = \{x_i^{(g)}\}, \quad i = 0, 1, \ldots, N_{pop-1} \quad g = 0, 1, \ldots, g_{max} \\ x_i^{(g)} = [x_{i,0}, x_{i,1}, x_{i,2}, \ldots, x_{i,D-1}]^T \end{gathered} \tag{3.2}$$

Habitualmente, el tamaño de la población N_{pop} se determina por la dimensión del problema. Un valor popular para N_{pop} es $N_{pop} = 10 \times D$ [Storn y Price, 2005]. Sin embargo valores más pequeños que $10 \times D$ pueden ser utilizados cuando el problema tenga una alta dimensionalidad.

Por conveniencia, en DE, los parámetros de un sistema son usualmente representados como individuos con una codificación real. Incluso si una variable es discreta o continua, debe de ser inicializada con un valor real, debido a que DE trata a todas las variables como valores reales independientemente de su tipo [Storn y Price, 2005].

Distribución de la población inicial

Cuando se requiere resolver un problema de optimización sin restricciones, en DE la población inicial puede ser creada con cualquier método de generación de individuos, por ejemplo con la Ecuación (1.4) y su distribución puede ser cualquier distribución conocida (distribución uniforme, distribución gaussiana, distribución binomial, etc.), la decisión de usar uno u otro enfoque depende de que tanto se conoce de la ubicación del óptimo global.

Por otro lado si se requiere resolver un problema como el planteado en la Ecuación (3.1), es indispensable asegurarnos que la población sea generada dentro de Ω; de acuerdo a la Figura 2.11 y al análisis realizado en la Sección 2.7, la opción a utilizar será el método

NGRFS (propuesto en el Algoritmo 1 del Capítulo 2) debido a que es adaptable a la forma geométrica del espacio y siempre generará individuos dentro de ésta.

3.4.2. Mutación diferencial

Una vez inicializado, DE realiza la mutación y recombina la población para producir la población de N_{pop} individuos de prueba. En particular la mutación diferencial añade una diferencia escalada entre dos vectores a un tercer vector [Lee y El-Sharkawi, 2008]. La Ecuación (3.3) muestra como se utilizan tres vectores aleatoriamente escogidos para crear el vector mutante $v_i^{(g)}$.

$$v_i^{(g)} = x_{r_0}^{(g)} + F(x_{r_1}^{(g)} - x_{r_2}^{(g)}) \tag{3.3}$$

donde el factor de escalamiento $F \in (0, 1)$ es un número real positivo, que controla la tasa de convergencia de la población. Más adelante analizaremos las propiedades del factor de escalamiento.

La Figura 3.2 ilustra como construir el individuo mutante $v_i^{(g)}$ en un espacio de dos dimensiones. Escalar las diferencias entre dos individuos asegura que los individuos mutantes no se dupliquen. Hay que tomar en cuenta el efecto que el parámetro de escalamiento provee a la mutación, porque si el escalamiento de las diferencias es pequeño, se podría llegar a converger en un óptimo local, por otro lado, si el escalamiento es muy grande, la convergencia del algoritmo podría ser muy lenta.

Índices en la mutación diferencial

Como se observa en la Ecuación (3.3), existen cuatro índices en DE que definen la mutación. El índice objetivo i, especifica el vector con el cual $v_i^{(g)}$ será cruzado para crear el individuo de prueba $u_i^{(g)}$ y realizar el proceso de selección entre $x_i^{(g)}$ y $u_i^{(g)}$. Los tres índices restantes r_0, r_1 y r_2 determinan cuales individuos de la población se combinan para crear $v_i^{(g)}$.

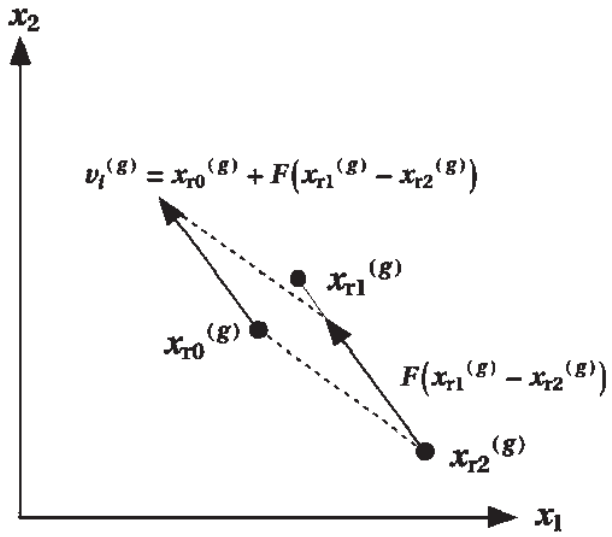


Figura 3.2: La diferencia escalada entre dos individuos añadida a un tercer individuos

Típicamente, el índice objetivo (i), r_0 y los índices que participan en la diferencia, son escogidos aleatoriamente cada vez que $u_i^{(g)}$ es generado. Por lo tanto, si los índices son escogidos aleatoriamente, existe la posibilidad de que algunos individuos sean escogidos más de una ocasión en una generación, mientras otros son omitidos. Duplicar un índice puede afectar el desempeño de búsqueda de una solución, mientras que omitir un índice puede privar a un individuo de la oportunidad de servir como un individuo base, por lo tanto es necesario asegurar que siempre se cumpla la condición: $i \neq r_0 \neq r_1 \neq r_2$.

Para asegurar que cada individuo sirve como individuo objetivo ($x_i^{(g)}$) solo una vez por generación, se utiliza el método de selección aleatoria de desplazamiento (*Random Offset Selection*). Este método asigna estocásticamente cada índice i a un único r_0; se calcula r_0 como la suma módulo N_{pop} del índice objetivo i y aleatoriamente se genera el desplazamiento r_g. El operador de módulo (mod) en la Ecuación (3.4) calcula el residuo de la división entera entre $(i + r_g)$ y N_{pop}.

$$r_0 = mod(i + r_g, N_{pop}) \tag{3.4}$$

donde r_g se calcula con la Ecuación (3.5)

$$r_g = \lfloor \alpha * N_{pop} \rfloor \tag{3.5}$$

donde $\alpha \sim U(0,1)$.

El desplazamiento aleatorio r_g es escogido al principio de cada generación, donde cada uno de los posibles valores que pueda adquirir r_g define un mapeo uno a uno entre los i y r_0. Por ejemplo, en la Figura 3.3 se observa un desplazamiento aleatorio cuando $r_g = 2$.

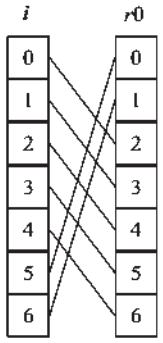


Figura 3.3: Mapeo uno a uno entre individuos objetivo e individuos base

Factor de escalamiento

Cuando se compara con las estrategias evolutivas, DE cambia la dinámica para adaptar el tamaño de paso. En el artículo original donde fue presentado el algoritmo DE [Storn y Price, 1995], el valor sugerido para el parámetro de escalamiento F se encontraba en el rango $[0, 2]$. Sin embargo pruebas empíricas han demostrado que para la mayoría de los problemas el valor óptimo de F se encuentra en el rango [0.4,1][Storn y Price, 1997].

Otro enfoque sugerido ha sido convertir F en una variable aleatoria, de tal forma que el escalamiento de las mutaciones siempre sea realizado con distintos valores de F. Más específicamente transformar el factor F en una variable aleatoria, amplia efectivamente el

espectro de las diferencias entre individuos más allá de las posibilidades permitidas por la combinación de individuos [Storn y Price, 2005].

Sin embargo, convertir el factor F en una variable aleatoria requiere seleccionar una función de densidad de probabilidad, conocer su grado de utilización y se pierde el control sobre la manera en que converge la población [Lee y El-Sharkawi, 2008]. Por lo tanto, para este trabajo se propone utilizar F como un valor fijo de ajuste manual.

3.4.3. Cruza

Para complementar la estrategia de búsqueda de la mutación diferencial, DE también emplea la cruza uniforme. Se define como un proceso en el cual se genera un número aleatorio y mediante una probabilidad se determina la fuente a ser heredada para cada parámetro j del individuo $u_i^{(g)}$. La cruza es uniforme en el sentido de que cada parámetro, a pesar de su posición en $u_{i,j}^{(g)}$, tiene la misma probabilidad C_r, de heredar su valor de $x_i^{(g)}$ ó de $v_i^{(g)}$.

La probabilidad de cruza $C_r \in (0, 1)$, es un valor definido por el usuario que controla la fracción de los valores de los parámetros que serán copiados del mutante o del vector objetivo. $C_r = 0.5$ es la probabilidad más utilizada, porque ambos donadores tienen la misma probabilidad de heredar sus valores. La manera de generar un individuo $u_i^{(g)}$ mediante una cruza está dada por la Ecuación (3.6).

$$u_i^{(g)} = u_{i,j}^{(g)} = \begin{cases} v_{i,j}^{(g)} \text{ si } (\alpha \leq C_r \text{ ó } j = j_r) \\ x_{i,j}^{(g)} \text{ si no} \end{cases} \tag{3.6}$$

donde $\alpha \sim U(0, 1)$ y j_r es un índice entero seleccionado aleatoriamente tal que $j_r \in [0, D-1]$.

Para determinar cual es el vector que contribuye con su valor del parámetro j, la cruza uniforme compara C_r a la salida de un generador de números aleatorios con distribución uniforme, α. Si α es menor o igual a C_r el valor del parámetro es heredado del individuo mutante $v_i^{(g)}$, de otra manera el parámetro es copiado del individuo objetivo $x_i^{(g)}$. En adición, si el índice aleatorio j_r es igual a j, entonces j es tomado de $v_i^{(g)}$ para asegurar que $u_i^{(g)}$ no duplique a $x_i^{(g)}$.

La Figura 3.4 muestra los posibles individuos de prueba que pueden resultar de escoger uniformemente un individuo mutante $v_i^{(g)}$, con el individuo objetivo, $x_i^{(g)}$

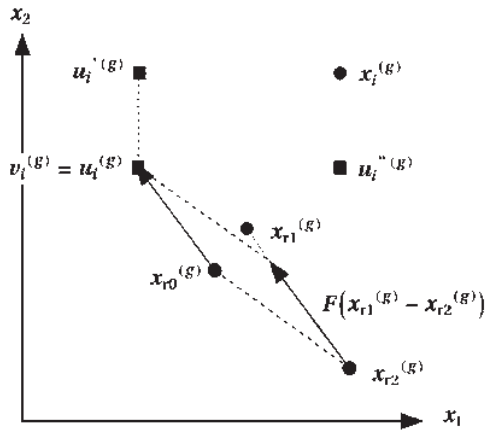


Figura 3.4: Los posibles individuos de prueba, $u_i^{'(g)}$, $u_i^{''(g)}$, cuando $v_i^{(g)}$ y $x_i^{(g)}$ son uniformemente cruzados.

3.4.4. Selección

Si el individuo $u_i^{(g)}$ tiene una evaluación de la función objetivo menor o igual que la evaluación del individuo objetivo $x_i^{(g)}$, el individuo $u_i^{(g)}$ reemplazará al individuo $x_i^{(g)}$ en la próxima generación; de otro modo, $x_i^{(g)}$ permanece en su lugar en la población por al menos una generación más. La Ecuación (3.7) muestra el proceso de selección.

$$x_i^{(g+1)} = \begin{cases} u_i^{(g)} \text{ si } f(u_i^{(g)}) \leq f(x_i^{(g)}) \\ x_i^{(g)} \text{ si no} \end{cases} \tag{3.7}$$

3.4.5. Criterio de parada

Un aspecto importante para un EA es decidir cuando detenerse. Con una buena capacidad de exploración y factores adecuados sabemos que estos algoritmos convergen a un valor óptimo con una probabilidad de uno cuando el tiempo tiende a infinito

[Torn y Zilinskas, 1989]. Sin embargo, esperar a tal convergencia es completamente imprácti-co. Por lo tanto el usuario necesita decidir en que condiciones terminará el algoritmo.

Decidir el criterio de parada depende de muchos factores, tales como la aplicación del algoritmo, la precisión deseada, costo y restricciones de tiempo. Para DE, algunos de los criterios de parada más comunes son:

- Máximo número de generaciones.
- Que la diferencia entre el mejor y peor valor de la función sea muy pequeña.
- El mejor valor de la función no se ha mejorado despues de un cierto número de generaciones.
- La distancia entre los individuos de la población es muy pequeña.

3.4.6. Algoritmo

Hasta ahora se han descrito los pasos involucrados en el proceso y las consideraciones que se deben tomar para la mejor elección de los valores de los parámetros. De manera general podemos decir que una vez que la población incial es generada, el proceso de mutación, cruza y selección se repite hasta que el óptimo es encontrado ó un criterio de terminación es alcanzado, por ejemplo que el número de generaciones alcance el máximo preestablecido, $g = g_{max}$.

La implementación más clásica de DE puede verse en el Algoritmo 3 y de la misma forma que con el algoritmo 2 este algoritmo funciona solamente para problemas de optimización sin restricciones.

3.5. Comparación de desempeño

Para poder comparar el desempeño entre los GAs y DE, se ha realizado un experimento en dos dimensiones, para que las poblaciones y sus perturbaciones puedan ser graficadas. La función objetivo utilizada para el experimento está dada por la Ecuación

Algoritmo 3 DE

Entrada: $f(x)$

Salida: x^*

–Generar la población inicial $P^{(0)}$ de forma aleatoria y hacer $g = 0$.

–**Repetir**

–Evaluar la aptitud de cada individuo en $P^{(g)}$.

–Realizar mutación diferencial mediante la Ecuación (3.5) con los índices $i \neq r_0 \neq r_1 \neq r_2$, para crear $v_i^{(g)}$.

–Efectuar cruzamiento uniforme con la Ecuación (3.6) entre $v_i^{(g)}$ y $x_i^{(g)}$ para generar $u_i^{(g)}$.

–Utilizar la Ecuación (3.7) para determinar $x_i^{(g+1)}$

–Hacer $g \leftarrow g + 1$

–Hasta alcanzar el criterio de parada

(3.8) y se compone de dos óptimos, haciendo las condiciones de análisis de convergencia más obvias en el experimento.

En la figura 3.5 se muestra que la función $f(x)$ produce dos óptimos en el espacio de búsqueda de la función (casi plana), los cuales se encuentran localizados en $[-11, -9]^T$ y $[11, 3]^T$. El valor para cada uno de los óptimos es 1.1 y 0.999998, respectivamente.

$$h(x_1, x_2) = \frac{x_1^2 + x_2^2}{0.001 + x_1^2 + x_2^2}$$

$$f(x_1, x_2) = h(x_1 + 11, x_2 + 9) + 1.1h(x_1 - 11, x_2 - 3) \tag{3.8}$$

Vista desde arriba, la función objetivo es bastante plana. Se escogió esta función para hacer el proceso de optimización más complicado. De hecho, cualquier método basado en gradiente tendrá serios problemas si su punto inicial es ubicado en la parte plana de la función, porque para un punto en la parte plana de $f(x)$, el gradiente $\nabla f(x) = 0$ y de

acuerdo a la condición suficiente de primer orden, nos encontramos en un mínimo o en un máximo de la función, pero en este punto, el determinante del hessiano $|\nabla^2 f(x)|$ es casi cero, creando una inconsistencia, porque esto indica que el hessiano es singular y no puede aportar más información para determinar si es un mínimo o es un máximo.

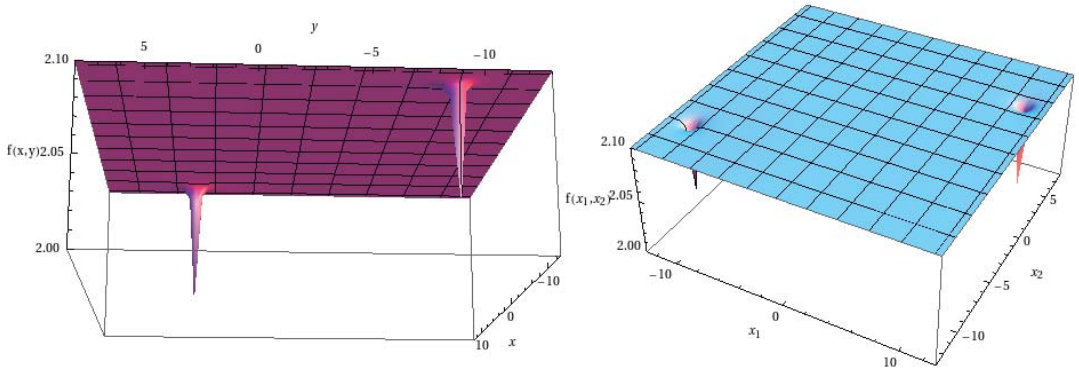


Figura 3.5: Función objetivo. a) Vista desde abajo y b)Vista desde arriba

Cada algoritmo fue iniciado con $N_{pop} = 50$, $g_{max} = 200$, y de una serie de experimentos realizados, se presenta el mejor comportamiento para cada uno. Para el caso de los GAs, los párametros utilizados fueron los siguientes: $Num_{Hijos} = 2$, $Mut_{prob} = 0.5$, $PointMut_{prob} = 0.5$, $Cr_{prob} = 0.5$, *TipoCr=1 Punto, Selección= Fitprop.* Para DE los parámetros utilizados fueron los siguientes: *F=0.9* y *Cr=0.5.*

De acuerdo con los datos mostrados en la Tabla 3.2, en las primeras generaciones los mejores individuos de los GAs se encuentran cerca del óptimo global [11,3], mientras que los mejores individuos de DE se encuentran cerca del óptimo local [-11,-9]. Sin embargo, ambos comparten una evaluación de la función objetivo similar.

A través del paso de las generaciones en el proceso evolutivo, los mejores individuos de los GAs quedan atrapados cerca del óptimo global, y es necesario que pase un gran número de generaciones para que la precisión de la solución se mejore. Por otro lado, a pesar de que en las primeras generaciones DE se encontraba cerca del óptimo local, su capacidad de exploración le permitió encontrar el óptimo global en unas pocas generaciones y a través del paso de más generaciones, la precisión continuó incrementándose.

Tabla 3.2: Comparación de desempeño entre los GAs y DE

Algoritmo	gen	$[x_1, x_2]$	$f(x)$
GAs	1	[12.5114, 0.506958]	2.09987
	2	[12.5114, 1.41661]	2.09977
	3	[12.5114, 1.41661]	2.09977
	18	[11.2306, 3.21964]	2.08926
	31	[11.0656, 3.21964]	2.07946
	84	[11.0656, 2.86057]	2.05555
	101	[11.0656, 2.92134]	2.00429
	128	[11.0656, 2.94628]	1.96571
	200	[11.0639, 2.94628]	1.96196
DE	1	[-8.53639, -8.88748]	2.09983
	2	[-8.53639, -8.88748]	2.09983
	3	[-10.6899, -8.58257]	2.09631
	16	[10.9254, 2.85175]	2.06145
	25	[10.9691, 3.15563]	2.05798
	31	[10.9233, 2.89432]	2.03905
	50	[11.0049, 3.00677]	1.07167
	78	[11.0002, 3.00031]	1.00015
	103	[11,3]	0.999998

La razón por la que los GAs no convergen con una buena precisión es por el mecanismo de selección utilizado. Dos padres se combinan y producen dos hijos. Estos dos hijos se encontrarán dentro del vecindario definido por los padres. Si la evaluación de un hijo es mejor que la evaluación de otro individuo en la población, el otro individuo será reemplazado por el hijo, reduciendo la diversidad genética de la población. Este efecto se repite una y otra vez a través del proceso evolutivo, hasta que la diferencia entre todos los individuos de la población es muy pequeña, por lo tanto, regularmente la población queda muy cerca del mejor individuo, y si el mejor individuo no se encuentra en el óptimo, para los GAs es muy complicado alcanzarlo.

Por otro lado, DE presenta un mecanismo de selección diferente. La selección aleatoria de los índices i, r_0, r_1 y r_2 preserva un comportamiento adecuado para proporcionar la participación de toda la población. Un nuevo individuo es creado a partir de tres individuos y será seleccionado solamente si su aptitud es mejor a la del individuo objetivo.

Esta estrategia guía la búsqueda mientras se preserva la riqueza genética de la población, en otras palabras, podemos decir que aunque el mejor individuo no se encuentre cerca o en el óptimo, la población será capaz de explorar más allá del mejor individuo y converger en la mejor solución, tal y como sucedió en el experimento realizado.

En la figura 3.6 se muestra el comportamiento de $f(x)$ a través del paso de las generaciones. Como se puede observar, después de la generación $g = 50$, DE se encuentra muy cerca de converger, y solo fueron necesarias algunas generaciones más para alcanzar $f(x^*)$. Mientras que, los GAs se quedaron estancados cerca del óptimo global, haciendo que la convergencia proceda de manera más lenta porque aún en la generación $g = 200$ no se ha podido encontrar $f(x^*)$.

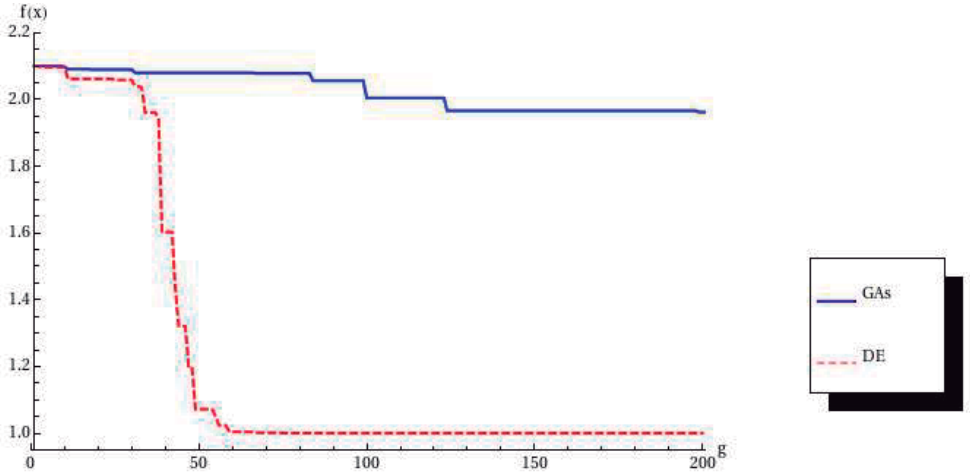


Figura 3.6: Función objetivo evaluada con los mejores individuos de cada generación

La velocidad de convergencia de DE se debe a la manera en la que explora su espacio de búsqueda. La exploración utilizada por DE está basada en diferencias de individuos; en tempranas generaciones, estas diferencias se encuentran en el rango de grandes a pequeñas. En generaciones posteriores, cuando la población está llegando a un punto de convergencia, el rango de las diferencias es pequeño y mientras más generaciones sucedan el rango se vuelve más pequeño, permitiendo una notable precisión de la solución. Esto es una ventaja para DE, debido a que no se necesita ajustar ningún parámetro de evolución.

Por otro lado las perturbaciones realizadas en los GAs, no están distribuidas cerca del origen, causando que los movimientos de los individuos perturbados se encuentren limitados a ciertas regiones. Este hecho decrementa la probabilidad de que un hijo o mutante se encuentre en el óptimo global.

Para mostrar esta propiedad, consideremos un individuo en alguna curva de nivel cercana al óptimo global. La mutación en los GAs selecciona algunos de los parámetros del individuo para realizar la mutación en esos genes en particular; para un caso de dos dimensiones, el individuo solamente se podría mover en una de las coordenadas (x_1 ó x_2). Por

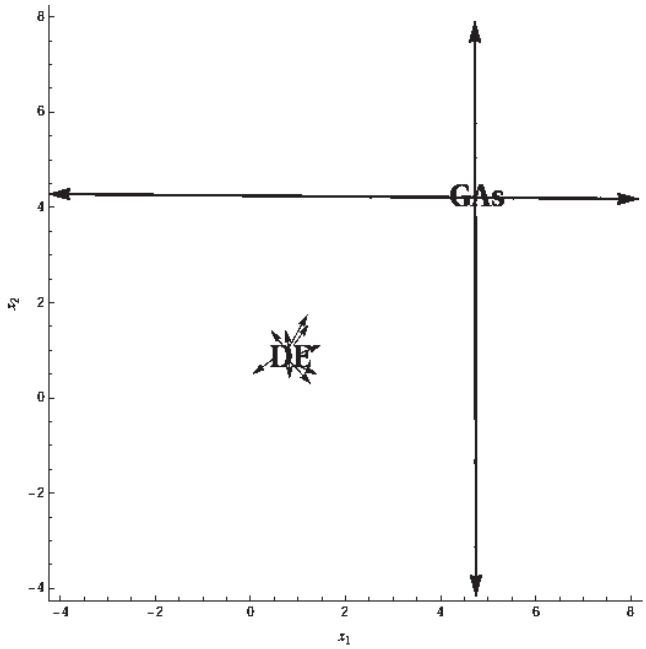


Figura 3.7: Esquemas de mutación utilizados por GA y DE

lo tanto, cualquier mutación a un parámetro de un individuo en dos dimensiones se encontrará en una"cruz" centrada en el individuo. Si el óptimo está fuera de la cruz de mutación, y la diferencias entre todos los individuos de la población es muy pequeña, es sumamente complicado moverse en dirección del óptimo. Para ilustrar este proceso, consideremos la Figura 3.7 como ejemplo, donde se muestran las curvas de nivel de alguna función objetivo y dos poblaciones distintas: GAs y DE. Toda la población de los GAs se encuentra en un

punto en particular, mientras que la población de DE se encuentra muy cerca de alcanzar el óptimo de la función. Las flechas horizontales y verticales que forman una cruz en el centro de la población de los GAs, muestran todos los posibles resultados para la mutación llevada a cabo por la población. Debido a que el óptimo de la función se encuentra fuera su área de influencia, la capacidad de exploración de los algoritmos genéticos no es muy útil, porque para alcanzar el óptimo de la función utilizando la mutación, los individuos de la población de los GAs se tendrían que desplazar verticalmente y luego horizontalmente, y quizás este proceso se repetiría un gran número de veces, debido a que la si la ubicación de la población de los GAs se encuentra como la mostrada en la Figura 3.7 y el óptimo se encuentra en dirección diagonal a la población, la única manera de llegar al óptimo es mediante una gran cantidad de movimientos verticales y horizontales.

Por otro lado, el proceso realizado por la mutación diferencial permite a la población de DE continuar moviéndose fácilmente hacia el óptimo de la función. Las flechas que se encuentran en el origen de la población de DE muestran la movilidad que mantiene la población. Como resultado, la calidad de las soluciones que ED presenta siempre son mejores que aquellas reportadas por los GAs. Ademas los GAs por lo general necesitan un empujón final utilizando algún método basado en gradiente.

3.6. Manejo de restricciones en los algoritmos evolutivos

Las restricciones típicamente hacen la optimización una tarea más complicada para los EAs, porque pueden crear regiones prohibidas en el espacio de búsqueda de la función objetivo, restringiendo el movimiento libre de la población [Storn y Price, 2005].

Dado que podemos generar una población inicial mediante el Algoritmo 1, estamos asegurando que la población inicial se encuentra dentro de Ω y por lo tanto, la población satisface todas las restricciones, solo debemos considerar las contribuciones que realizan los operadores evolutivos.

Debido a la capacidad de exploración que utilizan los EAs, y mediante la aplicación de estos operadores, se generan continuamente individuos que quizás violen alguna restricción, o se encuentren fuera de los límites permitidos, por lo tanto, en la siguiente

sección se realiza un análisis de las situaciones que provocan este comportamiento.

3.6.1. Operadores evolutivos

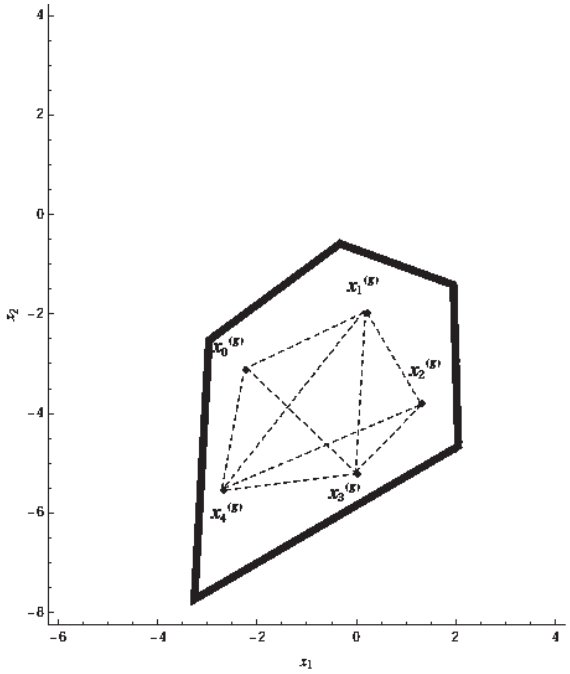
Tanto en los GAs como en DE los operadores evolutivos toman individuos de la población para generar otros individuos. En este proceso los nuevos individuos generados pueden no satisfacer las restricciones del problema y adicionalmente la evaluación de su función podría ser mucho menor que la de los individuos que sí satisfacen todas las restricciones, de tal manera que el algoritmo convergería a un punto infactible.

Debido al análisis realizado en la Sección 3.5, se propone DE como el algoritmo de optimización a utilizar y por lo tanto revisaremos como los individuos que se generan a través de la mutación diferencial y la cruza tienen la facultad de violar alguna o todas las restricciones.

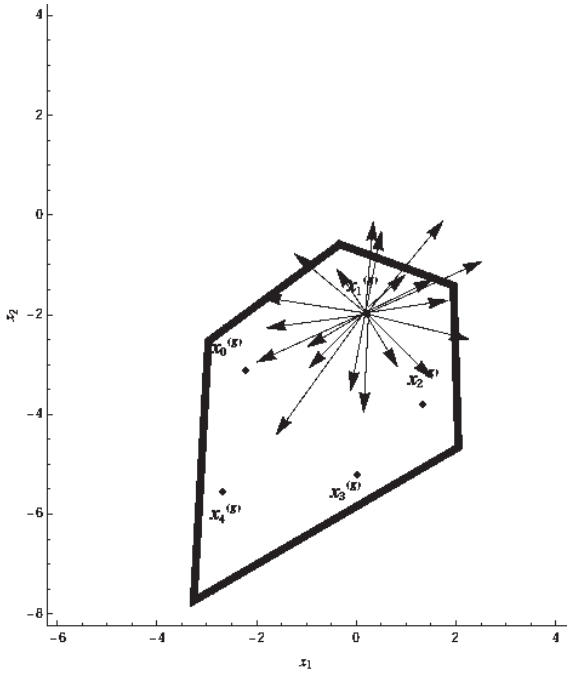
Hay que recordar la mutación diferencial funciona en base a diferencias entre individuos escogidos aleatoriamente. En la Figura 3.8(a) se muestran las posibles 20 diferencias que existen cuando $N_{pop} = 5$. Estas posibles diferencias son escaladas a la mitad ($F = 0.5$), y transportadas a un origen común ($x_1^{(g)}$) como en la Figura 3.8(b). Es sencillo notar el problema que surge al aplicar la operación de mutación diferencial, debido a que cuando la diferencia escalada de dos individuos escogidos aleatoriamente es añadida a un tercer individuo, esta adición puede rebasar los límites de Ω.

Existe un caso especial donde la mutación asegura no violar ningún límite. Esto sucede cuando los índices $r_2 = r_0$ y $F \in [0, 1]$, porque la Ecuación (3.5) se convierte a $v_i^{(g)} = x_{r_0}^{(g)} + F(x_{r_1}^{(g)} - x_{r_0}^{(g)})$, la cual es una representación paramétrica de la línea existente entre $x_{r_1}^{(g)}$ y $x_{r_0}^{(g)}$ por lo cual deducimos que si $x_{r_1}^{(g)}$ y $x_{r_0}^{(g)}$ se encuentran dentro de Ω, cualquier individuo generado se encontrará dentro. Sin embargo, con este enfoque se pierde la capacidad de exploración de DE, porque solamente se generarán individuos entre distintos pares de individuos (Figura 3.8(a)).

Otro operador evolutivo que afecta la convergencia de DE en un problema con restricciones es la cruza. Dado que los parámetros del vector generado son una combinación de los parámetros de dos vectores ($v_i^{(g)}$ y $x_i^{(g)}$) el problema es similar al de la mutación diferencial. La Figura 3.9 ilustra las cuatro posibles situaciones cuando se efectúa la cruza:



(a)



(b)

Figura 3.8: Violación de los límites de Ω mediante la operación de mutación diferencial. a)Posibles diferencias entre un conjunto de vectores , b)Escalamiento de las posibles diferencias centradas en un origen común.

El vector $u_i^{(g)}$ hereda directamente los parámetros de $v_i^{(g)}$, $u_i^{'(g)}$ es la combinación de un parámetro de $v_i^{(g)}$ y de otro parámetro de $x_i^{(g)}$, $u_i^{''(g)}$ es la situación contraria a $u_i^{'(g)}$, y finalmente donde $u_i^{(g)}$ hereda directamente los parámetros de $x_i^{(g)}$. En este caso el vector que viola las restricciones es $u_i^{''(g)}$.

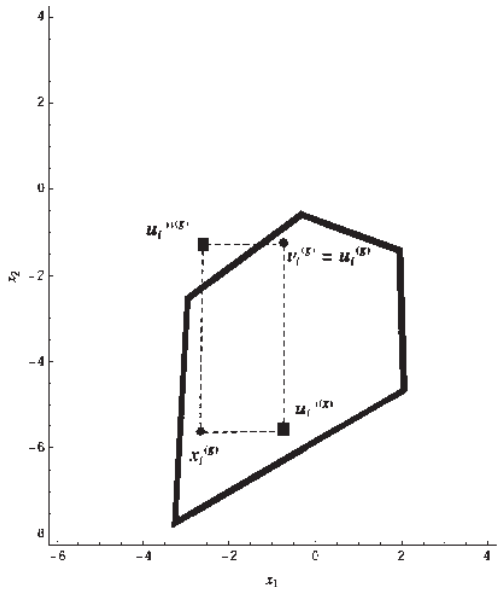


Figura 3.9: Violación de los límites de Ω mediante la cruza.

Dadas estas condiciones, la estrategia a utilizar para no salir de la región de factibilidad es utilizar el método propuesto en la Sección 2.7 mediante la Ecuación (2.23), el cual sirve para validar si una solución es generada dentro o fuera de la región de factibilidad. Sin embargo es importante mencionar que existen diversas técnicas para el manejo de restricciones en el contexto de los EAs. Michalewicz and Shoenauer [Michalewicz et al., 1994], han clasificado los manejos de restricciones para los EAs en cuatro categorías:

- Técnicas basadas en preservar las soluciones factibles
- Técnicas basadas en funciones de penalización
- Técnicas que distinguen entre soluciones factibles y soluciones infactibles.

- Técnicas híbridas.

Los primeros dos enfoques son sin lugar a duda los más populares, y por lo tanto solo se discutirán estos.

Las técnicas basadas en preservar las soluciones factibles regularmente son conocidas como técnicas de reparación. Es decir, que estas técnicas convierten los parámetros que hayan violado alguna restricción en un parámetro generado aleatoriamente dentro del rango permitido. Sin embargo, como se cambia radicalmente los valores de un parámetro, esto puede causar un comportamiento indeseado cuando las soluciones se encuentran en los límites de Ω [Bagdadhi, 2007].

El segundo enfoque popular para el manejo de restricciones es el uso de funciones de penalización. El objetivo es penalizar la evaluación de los individuos infactibles en favor de las soluciones factibles en el proceso de selección y reemplazo. La ventaja principal del uso de las funciones de penalización es su simplicidad; sin embargo, su gran desventaja es que los factores de penalización, los cuales determinan la magnitud del castigo, deben ser impuestos por el usuario y estos valores son dependientes del problema [Montes y Coello, 1998].

Los parámetros de penalización son también conocidos por influenciar directamente la convergencia del algoritmo. La "sobrepenalización" de los resultados que violaron las restricciones resulta en una exploración pobre del espacio de búsqueda y convergencia prematura a soluciones que usualmente son mínimos locales. Por otro lado una penalización muy pequeña resulta en una convergencia muy lenta hacia las soluciones factibles, y no existe garantía que una solución factible sea encontrada. En el Capítulo 4 se utilizarán funciones de penalización con el fin de comparar resultados.

3.7. Conclusiones del capítulo

Este capítulo presenta distintas maneras de resolver un problema de optimización, en las cuales sobresalen los Algoritmos Evolutivos, por su gran desempeño al momento de resolver este tipo de problemas.

Estos algoritmos son una gran familia conocida como EAs, y para poder implementar y aplicarlos a la solución de problemas es necesario conocer su naturaleza, sus operadores y su funcionamiento. Cabe destacar que entre los algoritmos pertenecientes a los EAs, existen diferencias notables como se mencionaron en la Sección 3.5 de este capítulo, las cuales deben ser analizadas para tener un mayor criterio sobre la elección del algoritmo adecuado.

En este contexto DE es presentado como un algoritmo robusto que permite la optimización global de problemas no-lineales con ventajas notables sobre los GAs, debido a sus pocos parámetros de ajuste y rápida convergencia.

Cuando no existen restricciones en la formulación del problema, el desempeño de los EAs es en general bastante confiable, sin embargo cuando las restricciones son añadidas al problema, la capacidad de exploración de los EAs se convierte en una desventaja, porque los vectores generados mediante los operadores evolutivos pueden violar las restricciones del problema. Por lo tanto es importante conocer cuales son las situaciones donde se violan los límites y así aplicar algún método o técnica útil para evitar que siga sucediendo.

En general este capítulo debe verse como la elección de la herramienta adecuada para resolver el problema planteado en el Capítulo 1, justificando la elección de dicha herramienta, sus características, la comparación con otra herramienta similar y analizando las posibles desventajas de la elección para poder plantear soluciones adecuadas a estas posibles desventajas.

Capítulo 4

Evolución Diferencial con NGFRS

4.1. Introducción

En los capítulos anteriores se describieron y analizaron distintos métodos utilizados para resolver un problema de optimización sujeto a restricciones. En el Capítulo 2, se desarrolló un algoritmo que permite capturar la forma geométrica de Ω, generando puntos que cumplen con todas las restricciones del problema. Este enfoque permite cambiar un problema sujeto a restricciones a uno libre de restricciones. Y en el Capítulo 3, se describieron las ventajas y desventajas de distintos algoritmos de optimización. Particularmente se realizó un análisis de desempeño entre los algoritmos GAs y DE, donde DE mostró ser un algoritmo confiable y robusto para la búsqueda del óptimo global.
El algoritmo DE pertenece a la familia de los EAs, los cuales por la naturaleza de sus procesos evolutivos, les permite convertirse fácilmente en híbridos. Diversas características de distintos algoritmos pueden ser incorporadas para producir nuevos y mejores algoritmos [Al-Sultan y Al-Fawzan, 1997].

Por lo tanto, se desea utilizar el algoritmo DE como optimizador base e incorporar el algoritmo NGFRS como parte de su funcionamiento modificando la manera en la que se generan los numeros aleatorios dentro de la region de factibilidad.

4.2. DE-NGFRS

En esta sección presentamos el algoritmo DE-NGFRS (*Differential Evolution based on Number Generation with the Feasible Region Shape*) como un algoritmo evolutivo híbrido que complementa la metaheurística de búsqueda y exploración de DE con el método de generación de números aleatorios NGFRS. Este complemento permite a DE mejorar su capacidad de exploración generando individuos con la forma de la región de factibilidad y guiar la búsqueda en espacios restringidos, validando cualquier individuo que sea generado mediante los operadores evolutivos.
Las características que se desea que este algoritmo posea son las siguientes:

- Generalidad. Capacidad de resolver un gran conjunto de problemas.
- Confiabilidad. Encontrar el óptimo global la mayoría de las ocasiones, con una precisión deseada.
- Eficiencia. Capacidad de exploración suficiente, mejorando la velocidad de convergencia del algoritmo.
- Simplicidad. El algoritmo deberá ser simple de comprender y de manipular. El número de parámetros deberá ser limitado para que el algoritmo se desempeñe correctamente sin necesitar demasiado afinamiento.
- Restricciones. Un manejo adecuado de restricciones que permita diferenciar entre una solución factible y una solución no-factible.

Cumpliendo con los requerimentos mencionados, se obtendrá un algoritmo robusto que permita encontrar el óptimo global en problemas de optimización sujetos a restricciones.

4.2.1. Descripción

Dada una función f sujeta a un conjunto de restricciones lineales, podemos encontrar el óptimo global utilizando el algoritmo DE-NGFRS. El funcionamiento de este algoritmo es comparable con el funcionamiento de DE, sin embargo, existen algunas diferencias

entre ellos, como el manejo de restricciones y la capacidad de exploración. A continuación se describen los pasos involucrados en el algoritmo DE-NGFRS.

Inicialización

Es necesario crear una población inicial que se encuentre dentro de Ω y como ya se analizó en capítulos anteriores, el mejor enfoque que podemos utilizar está dado por el algoritmo NGFRS. Se crean N_{pop} individuos que simulan la forma geométrica de Ω, y posteriormente son evaluados de acuerdo a su aptitud en el espacio de búsqueda (Seccion 2.7, algoritmo 1).

Evolución

Una vez creada y evaluada la población inicial, se puede proceder de dos maneras distintas:

1. Generar $u_i^{(g)}$ usando el esquema de DE (mutación diferencial y cruza)
2. Generar $u_i^{(g)}$ utilizando NGFRS.

La decisión de que método utilizar para generar $u_i^{(g)}$ está a cargo de un parámetro aleatorio $\gamma \sim U(0,1)$, el cual es comparado con una probabilidad definida por el usuario (en este caso la probabilidad es de 50 %). Si $\gamma \leq 0.5$, la generación está dada por mutación diferencial y cruza, por otro lado si $\gamma > 0.5$ se utiliza solamente NGFRS.

En la Figura 4.1(a) se muestra una población inicial con $N_{pop} = 9$ en el espacio de búsqueda de alguna función, donde el óptimo de la función se encuentra representado por una cruz. La población iniciará el proceso de evolución de acuerdo al valor de γ.

Nótese en 4.1(b) como $u_i^{(g)}$ es generado utilizando NGFRS con media $\mu = x_i^{(g)}$ y la covarianza de Ω. Como se puede observar, este enfoque tiene sus ventajas porque con una buena probabilidad un $u_i^{(g)}$ será generado a lo largo del elipsoide acercándose de manera más rápida al mínimo de la función.

En la Figura 4.1(c) las diferencias entre los individuos existentes en la población son acomodadas en un origen común. Posteriormente se realizará una cruza en el proceso y

ese será el nuevo individuo generado.

Si la población se encuentra encerrada en un espacio muy retirado del óptimo de la función, la capacidad exploratoria de NGFRS permitirá un rápido avance hacia el óptimo, mientras que cuando la población se este aproximando a un punto de convergencia las operaciones evolutivas realizadas por DE asegurarán la convergencia y precisión deseada.

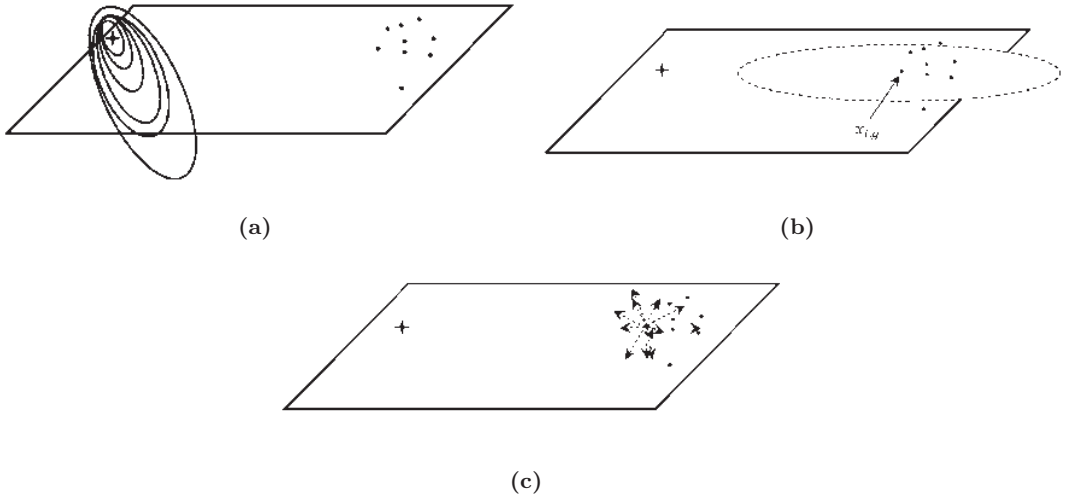


Figura 4.1: Movimiento de individuos hacia el óptimo a) Población inicial alejada del mínimo, b) Esquema NGFRS, c) Esquema DE

Selección y Finalización

Una vez generado $u_i^{(g)}$, su evaluación de la función objetivo será comparada con la evaluación de $x_i^{(g)}$, y el valor más pequeño será $x_i^{(g+1)}$. El proceso evolutivo y de selección se repite hasta que todos los individuos de la población hayan sido escogidos como individuos objetivo ($i = N_{pop}$). Y a su vez, todo el esquema mencionado se repite un número de generaciones ($g = g_{max}$) ó hasta cumplir con algún criterio de parada establecido.

4.2.2. Algoritmo

El Algoritmo 4, resume todos los pasos descritos en la Sección 4.2.1.

Algoritmo 4 DE-NGFRS

Entrada: q y $f(x)$

Salida: x^*

–Dado N_{pop}, F, C_r

–Crear la población inicial $P^{(g)} = [x_1^{(g)}, x_2^{(g)}, \ldots, x_n^{(g)}]$ con el algoritmo 1, con media μ_q y evaluar la función de aptitud $f(x)$ sin restricción, para cada miembro de la población.

–**Para** $g = 0, 1, 2, \cdots, N_{gen}$ {

  –**Para** $i = 0, 1, 2, \cdots, N_{pop}$ {

    –Genera $\gamma \sim U(0, 1)$

    –**Si** $\gamma < 0.5$ {

      –Calcula $v_i^{(g)}$ con la Ecuación (3.5)

      –Calcula la cruza $u_i^{(g)}$ con la Ecuación (3.6)

      –Selecciona $x_i^{(g+1)}$ con (3.7)

      –**Si** $x_i^{(g+1)}$ está fuera de Ω, $x_i^{(g+1)} = x_i^{(g)}$

    }

    –**Si no**, calcula $u_i^{(g)}$ con el algoritmo NGFRS con media $x_i^{(g)}$.

    –Selecciona $x_i^{(g+1)}$ con (3.7)

    –**Si** $x_i^{(g+1)}$ está fuera de Ω, $x_i^{(g+1)} = x_i^{(g)}$

  }

}

4.2.3. Aspectos a considerar

De acuerdo al teorema No-Free-Lunch (**NFL**), no existe ningún algoritmo capaz de resolver todos los problemas de optimización y que en promedio sea superior a cualquier buen competidor [Wolpert y Macready, 1997].

El teorema justifica por qué ante el conjunto de todos los problemas de optimización matemáticamente posibles, en promedio, los algoritmos se comportan exactamente de

la misma forma. Por lo tanto, la única manera en la que un algoritmo puede ser mejor que otro es cuando este se especializa en algún problema en específico.

Antes de comenzar con los experimentos es necesario considerar algunos aspectos para la correcta implementación y funcionamiento del algoritmo DE-NGFRS. Tales aspectos son la dimensionalidad del problema y tipo de restricciones.

Alta dimensionalidad

Un problema común que se presenta cuando se desea resolver problemas mediante EAs es la alta dimensionalidad. Generalmente el problema radica en que el tamaño del espacio de búsqueda se incrementa de manera exponencial y es necesario crear una gran cantidad de individuos para realizar la búsqueda. Por ejemplo, para 1D el espacio de búsqueda es una línea con 2^1 vértices, para 2D el espacio de búsqueda es un cuadro con 2^2 vértices y para 3D el espacio de búsqueda es un cubo con 2^3 vértices; de manera que dependiendo del número de dimensiones, el número de vértices del espacio de búsqueda es 2^D. Para un espacio de búsqueda con 2^D vértices, se esperaría que el valor de N_{pop} fuera al menos la misma cantidad de vértices, sin embargo, no es muy práctico utilizar este enfoque para el tamaño de la población, porque para dimensiones mayores a diez, sería necesario evaluar una gran cantidad de individuos por generación.

En la Figura 4.2 se puede observar el crecimiento de la población y el número total de evaluaciones puede calcularse mediante la Ecuación (4.1).

$$Evals = 2^D \times g_{max} \tag{4.1}$$

Por ejemplo, consideremos un problema donde $D = 15$, entonces $N_{pop} = 32,768$. Si el número de generaciones es $g_{max} = 100$, el número total de evaluaciones sería igual a $3,276,800$. Por otro lado, como se mencionó en la Sección 3.4.1, un valor común para el tamaño de la población es $10 \times D$. De acuerdo con esta proposición, si $D = 1$, entonces $N_{pop} = 10$. Por lo tanto, utilizando este enfoque el número total de evaluaciones puede calcularse con la Ecuación (4.2).

$$Evals = 10 \times D \times g_{max} \tag{4.2}$$

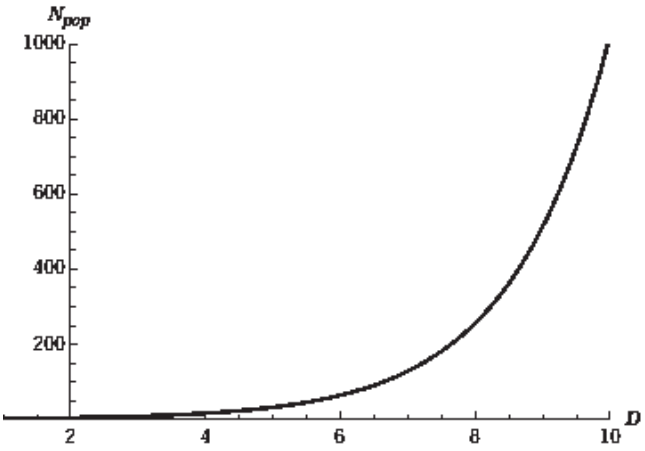


Figura 4.2: Crecimiento de la población con la dimension del problema

Utilizando el ejemplo anterior, si $D = 15$, entonces $N_{pop} = 150$. Si el número de generaciones es $g_{max} = 100$, entonces el número total de evaluaciones de la función objetivo sería igual a $150,000$. A pesar de que el tamaño de la población y número de evaluaciones no es tan grande como en la primera proposición de esta sección, la velocidad de convergencia de DE-NGFRS podría verse afectada si el problema tiene una alta dimensionalidad, debido al tiempo consumido para evaluar a cada individuo.

Tipos de restricciones

La validación de soluciones utilizada por el algoritmo DE-NGFRS solamente trabaja con espacios convexos. Por lo tanto, las restricciones aplicadas a los problemas a resolver deberán ser lineales. De acuerdo con la Sección 2.4, el conjunto resultante de la intersección de múltiples restricciones lineales tendrá la propiedad de ser convexo, sin embargo, como se verá en las siguientes secciones, en ocasiones calcular la intersección de las restricciones lineales puede representar un problema por si mismo.

4.3. Experimentos

Para probar el desempeño de nuestro algoritmo, se ha utilizado con cinco funciones. La primera es una función en dos dimensiones y de grado seis; la segunda es la función de Rosenbrock, y las tres funciones restantes son las funciones objetivo que pertenecen al problemas de Despacho Económico conocidos como IEEE14 [Ongsakul y Ruangpayoongsak, 2001], Wong [Wong y Fung, 1993] y Wood [Wood y Wollenberg, 1984].

A manera de comparación se utiliza DE con funciones de penalización y el programa Mathematica 8. DE con funciones de penalización es la técnica más utilizada para manejar las restricciones, sin embargo en la mayoría de las ocasiones, la solución se sesga debido a un factor de penalización, perdiendo tiempo de búsqueda y precisión. La formulación básica de las funciones de penalización está dada por la Ecuación (4.3)

$$F(x) = f(x) + \lambda \sum_{i=0}^{M-1} C(-a_i^T x - b_i) \tag{4.3}$$

donde λ es un factor de penalización, $C(y)$ es 1 si $y > \tau$ y 0 si no, y τ es un umbral.

Por otro lado, Mathematica contiene funciones dedicadas a la optimización de problemas, entre las cuales se encuentran las heurísticas *búsqueda aleatoria, SA* y *DE*, por lo tanto, Mathematica con DE también es utilizado como método de comparación. En las siguientes secciones se dan los detalles de los experimentos realizados.

4.4. Funciones de referencia

En esta sección se realizaron dos experimentos, utilizando dos funciones de referencia en dos dimensiones cada una. La primer función está sujeta a seis restricciones y la función de Rosenbrock está sujeta a cuatro restricciones.

4.4.1. Cálculo de intersección de restricciones lineales en 2D

Dado un conjunto de restricciones ψ (igualdades y desigualdades) de la forma $ax + by + c = 0$ y $ax + by + c \geq 0$, donde a y b son los coeficientes de x y y y c es el término independiente. Cada restricción representa un subespacio definido por el signo de la desigualdad y el cálculo de la región de factibilidad está dado por la intersección de este conjunto.

Es necesario calcular la intersección de cada una de las restricciones contra todas las demás. Las restricciones son revisadas por pares, y cada par de restricciones se coloca en un sistema de ecuaciones, donde el número de incógnitas es igual al número de ecuaciones. La manera más común de solucionar este sistema de ecuaciones es mediante la *eliminación gaussiana* [Strang, 1988]. Si las restricciones son geométricamente paralelas, la intersección entre ellas no sucederá en ningún punto.

Cada punto de intersección es almacenado y una vez que todas las restricciones son revisadas, todos los puntos de intersección encontrados son evaluados en cada una de las restricciones. Si algún punto falla en su evaluación este punto es desechado.

Finalmente, se calcula la envoltura convexa con algún algoritmo (por ejemplo, quickhull ó giftwrapping [O'Rourke, 1998]) del conjunto de puntos encontrados. Los pasos involucrados en el cálculo de la intersección de restricciones lineales en 2D se resumen en el Algoritmo 5.

4.4.2. Polinomio de sexto grado

Para este experimento se propone minimizar la función $f_1(x)$ dada por la Ecuación (4.4). La envoltura convexa de Ω se determinó calculando la intersección de las restricciones mediante el Algoritmo 5 y está dada por $q = \{[-23, -95], [-24, -94], [-20, -50], [19, 1], [20, -2], [10, -40]\}$, en la Figura 4.3, podemos observar gráficamente como la envoltura convexa de Ω es un polígono con seis vértices ubicado en las curvas de nivel de la función.

Algoritmo 5 Cálculo de la intersección de restricciones lineales en 2D

Entrada: Conjunto de restricciones ψ

Salida: Ω

$n \leftarrow$ Número de restricciones

$k \leftarrow 0$

–**Para** $j = 1, 2 \cdots, n-1$ {

–**Para** $i = j+1, \cdots, n$ {

–**Si** ψ_j y ψ_i no son paralelas {

–Aplicar eliminación gaussiana al sistema de ecuaciones entre ψ_j y ψ_i

– $k++$

– Almacenar la solución en g_k }

}

}

–**Para** $j = 1, 2 \cdots, n$ {

–**Para** $i = 1, 2, \cdots, k$ {

– Evaluar ψ_j $(ax + by + c \geq 0)$ con g_k y eliminar cada g_k que no cumpla con las restricciones

}

}

$\Omega \leftarrow$ Calcular la envoltura convexa de g.

$$f_1(x) = -3x_0^6 + 2x_0^5 - x_0 + 2x_1^3 + 45x_1 + 23 \tag{4.4}$$

$$s.a$$

$$x_0 + x_1 + 118 \geq 0, \quad 11x_0 - x_1 + 170 \geq 0,$$

$$-3x_0 - x_1 + 58, \quad 17x_0 - 13x_1 - 310 \geq 0,$$

$$-5x_0 + 3x_1 + 170, \quad -19x_0 + 5x_1 + 390 \geq 0$$

Se utilizó el algoritmo DE-NGFRS para minimizar esta función. Los resultados

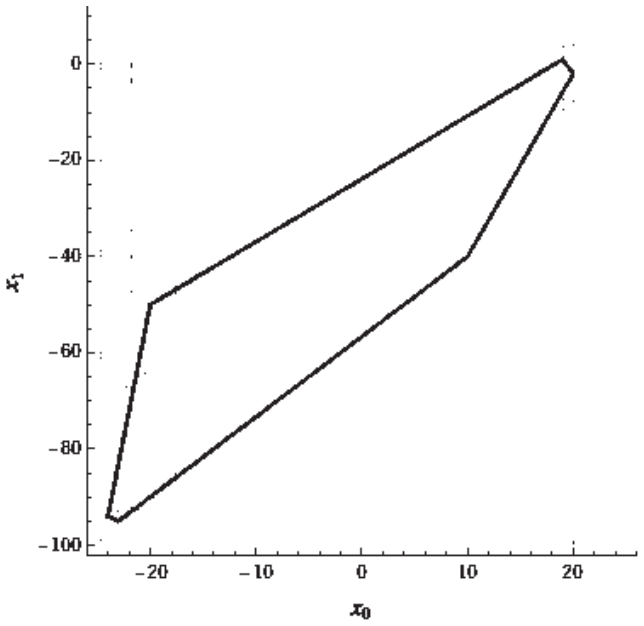


Figura 4.3: Región factible delimitada por las restricciones

se comparan contra DE con funciones de penalización y DE-Mathematica. En el caso del algoritmo DE con funciones de penalización (4.3), el costo asociado por incumplir una restricción fue $\lambda = 1 \times 10^{10}$. En la Tabla 4.1 se presentan el mejor caso, el promedio y el peor de 100 corridas del algoritmo DE-NGFRS, en el caso de DE-Mathematica siempre se encuentra una solución en un óptimo local y en el caso de funciones de penalización se tiene convergencia en algunos casos y en otros no.

Tabla 4.1: Polinomio de sexto grado

Algoritmo	**Valores**	$\boldsymbol{x}$	$\boldsymbol{f_1(x)}$
DE-NGFRS	Min	[-24.0000,-94.0000]	-5.90900×10^8
	Media	[-24.0000,-94.0000]	-5.90900×10^8
	Max	[-23.9983,-93.9818]	-5.90900×10^8
Mathematica	Siempre	[1.5250, 0.0000]	0.234956
Penalización	Min	[-24.0000,-94.0000]	-5.9090×10^8
	Media	$[-1.58 \times 10^{22}, -1.12 \times 10^{47}]$	-2.8228×10^{145}
	Max	$[-1.48 \times 10^{24}, -1.11 \times 10^{49}]$	-2.82282×10^{147}

4.4.3. Función de Rosenbrock

La función de Rosenbrock es una función utilizada como prueba de desempeño para múltiples algoritmos de optimización. En este experimento se presenta la función $f_2(x)$ como la función de Rosenbrock con restricciones, esta función está dada por la ecuación (4.5).

Los vértices del conjunto factible fueron calculados mediante el Algoritmo 5 y son $q = \{[2,4],[-39,101],[-40,100],[1,3]\}$, los cuales se pueden observar en la Figura 4.4. El costo para la función de penalización (4.3) fue de $\lambda = 100$. En la Tabla 4.2, se muestran los resultados; podemos notar que en todos los casos nuestra propuesta presenta mejores resultados. DE con funciones de penalización converge en algunos casos y Mathematica presenta resultados similares al algoritmo DE-NGFRS.

$$f_2(x) = (1 - x_0)^2 + 100(x_1 - x_0^2)^2 \tag{4.5}$$

$$s.a.$$

$$-97.0x_0 - 41.0x_1 + 358 \geq 0$$

$$x_0 - x_1 + 140.0 \geq 0$$

$$97.0x_0 + 41.0x_1 - 220 \geq 0$$

$$-x_0 + x_1 - 2.0 \geq 0$$

Tabla 4.2: Funcion de Rosenbrock

Algoritmo	**Valores**	$\boldsymbol{x}$	$\boldsymbol{f_2(x)}$
DE-NGFRS	Min	[1.99889, 3.99889]	0.998889
	Media	[1.99889, 3.99889]	0.998889
	Max	[1.99880, 3.99889]	0.998889
Mathematica	Siempre	[1.99888, 3.99888]	0.998889
Penalización	Min	[1.99889, 3.99889]	0.998889
	Media	[-3.76033, 17.30290]	48.9379
	Max	[-7.58911, 57.60030]	173.776

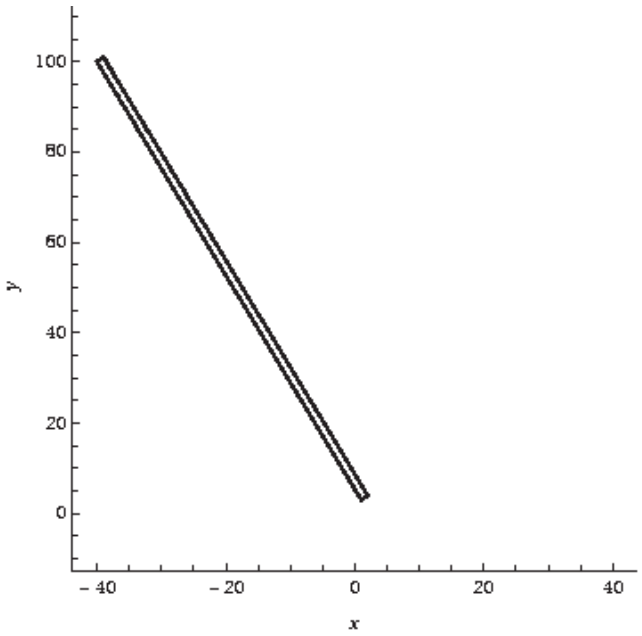


Figura 4.4: Función de Rosenbrock sujeta a una región factible

4.5. Despacho Económico

4.5.1. Introducción

La operación eficiente y óptima de un sistema eléctrico de generación de potencia activa siempre ha ocupado una importante posición en la industria eléctrica [Stevenson, 1982]. El conocimiento del flujo de cargas en un sistema eléctrico de potencia, permite hallar la potencia activa y reactiva que debe entregar cada unidad generadora para atender una demanda de potencia determinada. El reparto de cargas o potencias entre generadores depende de las condiciones de operación que se impongan. La demanda de potencia en un sistema eléctrico puede ser generada de diversas formas, y de todos los posibles repartos de carga que cumplan con esta demanda, nos interesa aquel que supone un mínimo costo de generación [Wood y Wollenberg, 1984].

4.5.2. Formulación

El problema de Despacho Económico (*Economic Dispatch*-**ED**) se define en términos de minimizar el costo total de generación activa sujeto a satisfacer la demanda total de potencia activa y las pérdidas de transmisión, mientras se mantiene la generación de potencia activa dentro de los límites [Stevenson, 1982].
Para un sistema con N unidades generadoras, el costo $c(P_i, \alpha_i)$, para cada generador se define como una función polinomial dada por la Ecuación (4.6).

$$c(P_i, \alpha_i) = \alpha_{i,0} + \alpha_{i,1}P_i + \alpha_{i,2}P_i^2 + \cdots = \sum_{j=0}^{N_c} \alpha_{i,j}P_i^j \tag{4.6}$$

donde P_i es la potencia activa para cada unidad generadora, α_i es el conjunto de coeficientes polinomiales, $\alpha_{i,j}$ es el j-ésimo coeficiente polinomial asociado a la i-ésima unidad generadora y N_c es el orden del polinomio. El costo total C es por supuesto la suma de costos de cada unidad generadora y el problema de minimizar el costo total puede ser formulado como un proceso de optimización no-lineal (4.7), sujeto a múltiples restricciones dadas por las ecuaciones (4.8) y (4.9).

$$min \quad C = \sum_{i=1}^{N} c(P_i, \alpha_i) \tag{4.7}$$

$$s.a. \quad \sum_{i=1}^{N} P_i = P_T \tag{4.8}$$

$$P_i^{min} \leq P_i \leq P_i^{max} \tag{4.9}$$

donde P_T es la potencia total activa demandada por las cargas y la transmisión del sistema, y P_i^{min} y P_i^{max} son los límites de generación para cada unidad en particular.

La restricción dada por la Ecuación (4.8) corresponde al balance de la potencia total activa que debe de existir en la red, y se refiere a ella como la Restricción de la Conservación de la Energía (*Energy Conservation Constraint*). Por otro lado, la restricción dada por la Ecuación (4.9) corresponde a un conjunto de límites de generación y se les conoce como Restricciones de Capacidad (*Capability Constraints*).

4.6. Delimitación de Ω en el Despacho Económico

Gran parte de las restricciones del problema de ED aseguran que las unidades generadores se encuentren dentro del rango permitido. Dependiendo de la dimensionalidad del problema, las restricciones dadas por la Ecuación (4.9) generan un hiper–cubo, donde cualquier solución generada dentro del hiper–cubo es una solución aceptada. Por otro lado, la Ecuación (4.8) determina un hiper–plano en el espacio que satisface la restricción de la conservación de la energía.

Nótese que tanto la Ecuación (4.8) y la Ecuación (4.9) son restricciones lineales, y como se vio en la Sección 2.4, el conjunto de soluciones permitidas se encuentran en la intersección del hiper–cubo y del hiper–plano.

Sin embargo, encontrar la intersección entre un hiper–cubo y un hiper–plano puede ser computacionalmente costoso, especialmente para problemas con altas dimensiones [Lara et al., 2009]. Por lo tanto, para calcular la intersección entre el hiper–cubo e hiper–plano se utiliza el enfoque utilizado por [Lara et al., 2009].

Este algoritmo calcula la intersección entre las restricciones del problema de ED, obteniendo una serie de vectores que son la envoltura convexa de Ω. El cálculo de la envoltura convexa de Ω mediante este enfoque, brinda las bases ideales para para trabajar con DE-NGFRS y solucionar el problema de ED.

4.7. Escenarios de Despacho Económico

Para esta prueba se presentan tres redes eléctricas correspondientes a IEEE14 (ver [Ongsakul y Ruangpayoongsak, 2001]), Wood (ver [Wood y Wollenberg, 1984]) y Wong (ver [Wong y Fung, 1993]).

Las funciones de costo para cada una de las redes está dada por las ecuaciones (4.10), (4.11) y (4.12) respectivamente. El conjunto de vértices de las envolturas convexas de las correspondientes regiones de factibilidad fueron calculadas con el algoritmo propuesto por [Lara et al., 2009], estos vectores se muestran en la Tabla 4.3. Gráficamente podemos observar las regiones de factibilidad (Ω_2 y Ω_3) de la segunda y tercera red respectivamente (debido a la dimensionalidad del problema no es posible graficar la primera red) en las

Figuras 4.5(a) y 4.5(b). Una vez que el conjunto de las envolturas convexas es calculado, se procede a resolver la versión del problema como si fuese uno sin restricciones, utilizando el Algoritmo 4.

En la Tabla 4.4 se muestra una comparación para las tres redes. En esta tabla mostramos que podemos resolver los problemas de ED en un tiempo muy inferior que los reportados en [Calderón et al., 2008] y con una mejor precisión como es el caso de la red de Wong utilizando el algoritmo DE-NGFRS.

$$
\begin{aligned}
f_{IEEE14}(P) = {} & 2 \times 10^{-5} + 0.003P_1 + 0.01P_1^2 \\
& +2 \times 10^{-5} + 0.003P_2 + 0.01P_2^2 \\
& +2 \times 10^{-5} + 0.003P_3 + 0.01P_3^2 \\
& +2 \times 10^{-5} + 0.003P_4 + 0.01P_4^2 \\
& +2 \times 10^{-5} + 0.003P_5 + 0.01P_5^2 \\
& s.a. \\
10 \leq P_1 \leq 80, \quad & 10 \leq P_2 \leq 60, \quad 10 \leq P_3 \leq 60, \\
& 10 \leq P_4 \leq 60, \quad 10 \leq P_5 \leq 80, \\
& P_1 + P_2 + P_3 + P_4 + P_5 = 300.5576
\end{aligned}
\tag{4.10}
$$

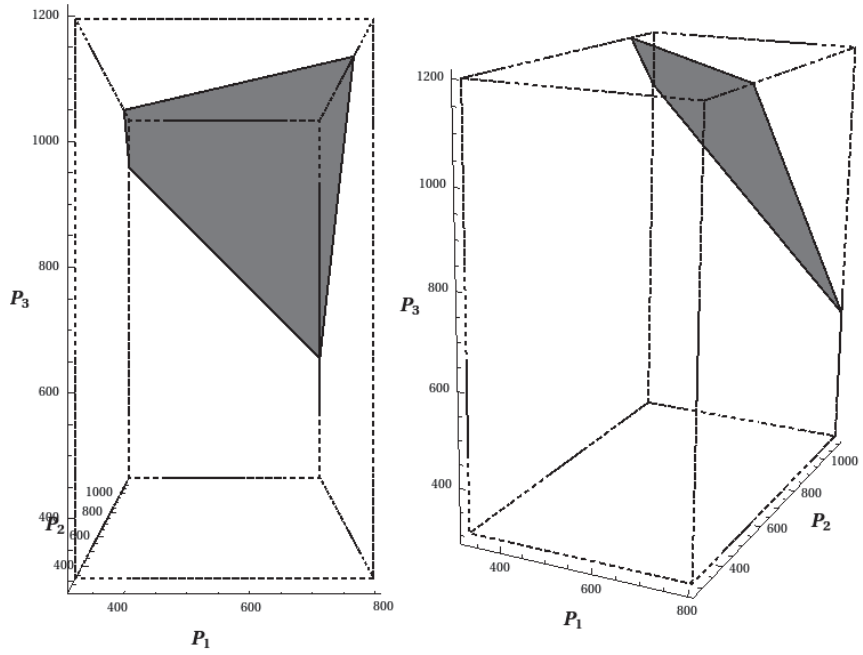
$$
\begin{aligned}
f_{Wood}(P) = {} & 749.55 + 6.950P_1 + 9.680 \times 10^{-4}P_1^2 + 1.270 \times 10^{-7}P_1^3 \\
& +1285.00 + 7.051P_2 + 7.375 \times 10^{-4}P_2^2 + 6.453 \times 10^{-8}P_2^3 \\
& +1531.00 + 6.531P_3 + 1.040 \times 10^{-3}P_3^2 + 9.980 \times 10^{-8}P_3^3 \\
& s.a \\
320 \leq P_1 \leq 800, \quad & 300 \leq P_2 \leq 1200, \\
275 \leq P_3 \leq 1100, \quad & P_1 + P_2 + P_3 = 2500
\end{aligned}
\tag{4.11}
$$

$$
\begin{aligned}
f_{Wong}(P) = 11.20 + 5.10238P_1 - 2.64290 \times 10^{-3}P_1^2 + 3.3333 \times 10^{-6}P_1^3 \qquad (4.12)\\
-632.00 + 13.01P_2 - 3.05714 \times 10^{-2}P_2^2 + 3.3333 \times 10^{-5}P_2^3\\
+147.144 + 4.28997P_3 + 3.08450 \times 10^{-}4P_3^2 - 1.7677 \times 10^{-7}P_3^3\\
s.a\\
100 \leq P_1 \leq 500, \quad 100 \leq P_2 \leq 500,\\
200 \leq P_3 \leq 1000, \quad P_1 + P_2 + P_3 = 1443.4
\end{aligned}
$$

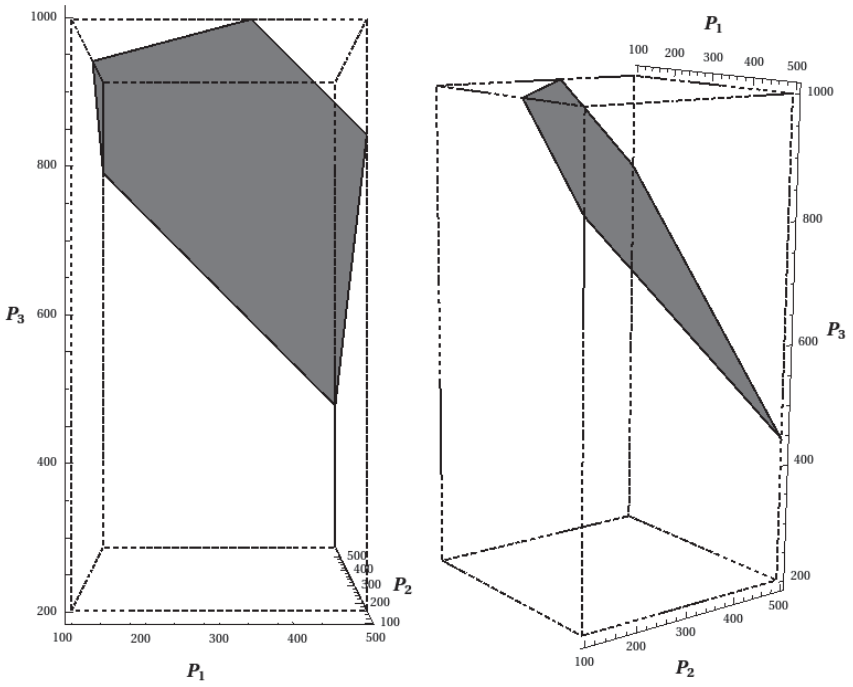
Tabla 4.3: Envolturas convexas calculadas por [Lara et al., 2009].

Red	**Vectores de potencia en KW**
IEEE 14	$q_1 = [40.5576, 60.0000, 60.0000, 60.0000, 80.0000]^T$ $q_2 = [80.0000, 20.5576, 60.0000, 60.0000, 80.0000]^T$ $q_3 = [80.0000, 60.0000, 20.5576, 60.0000, 80.0000]^T$ $q_4 = [80.0000, 60.0000, 60.0000, 20.5576, 80.0000]^T$ $q_5 = [80.0000, 60.0000, 60.0000, 60.0000, 40.5576]^T$
Wood	$q_1 = [320.0, 1080.0, 1100.0]^T$ $q_2 = [800.0, 600.0, 1100.0]^T$ $q_3 = [320.0, 1200.0, 980.0]^T$ $q_4 = [800.0, 1200.0, 500.0]^T$
Wong	$q_1 = [343.40, 100.00, 1000.00]^T$ $q_2 = [100.00, 343.40, 1000.00]^T$ $q_3 = [100.00, 500.00, 843.40]^T$ $q_4 = [500.00, 500.00, 443.40]^T$ $q_5 = [500.00, 100.00, 843.40]^T$

En la Tabla 4.5 se muestran los parámetros con los que se realizaron estas pruebas y el promedio y desviación estándar de la función de error de 100 corridas del algoritmo para estas cinco pruebas. En esta tabla se muestra en cada columna el nombre, el tamaño de la población N_{pop}, el número de generaciones N_{gen}, el parámetro de mutación F, el de cruzamiento C_r y la media de la función de error $\mu_{f(x)}$ y la desviación estándar $EstDev(f(x))$ para 100 experimentos. Note que la media de la función es consistente en los 100 experimentos con una desviación estándar muy baja.



(a) *Wood*



(b) *Wong*

Figura 4.5: Regiones de factibilidad para la red de Wood y la red de Wong, a)Dos vistas de la envoltura convexa de Ω_{Wood}, b)Dos vistas de la envolutura convexa de Ω_{Wong}

Tabla 4.4: Resultados comparativos para el Despacho Económico

Red	**Algoritmo**	**Demanda**	**Costo Generación**	**Tiempo (seg.)**	**Vector solución de Potencia Activa**
IEEE14	Ongsakul	300.5576	181.5724	-	[60.2788 60.0000 ... 60.0000 60.0000 60.2788]
	Calderon	300.5576	181.5724	4.276	[60.2789 59.9999 ... 59.9999 59.9997 60.2789]
	DE-NGFRS	300.5576	181.5724	0.811	[60.2788 60.0000 ... 60.0000 60.0000 60.2788]
Wood	Wood	2500.1000	22730.21669	-	[726.9000 912.8000 860.4000]
	Calderon	2500.0000	22729.32458	2.814	[725.0078 910.1251 864.8670]
	DE-NGFRS	2500.0000	22729.32458	0.027	[724.9915 910.1534 864.8551]
Wong	Wong	1462.4480	6639.50400	-	[376.1226 100.0521 986.2728]
	Calderon	1443.4000	6552.23790	2.842	[343.3980 100.0415 999.9604]
	DE-NGFRS	1443.4000	6552.09315	0.260	[343.4000 100.0000 1000.0000]

Tabla 4.5: Parámetros y resultados para todos los ejemplos

Ejemplo	N_{pop}	N_{gen}	F	C_r	$\mu_{f(x)}$	$EstDev(f(x))$
G6	50	500	0.95	0.90	-5.9090×10^{8}	0.01118
Rosenbrock	1500	500	1.2	0.90	0.9988	1.8058×10^{-15}
IEEE14	1000	1000	0.95	0.90	181.5724	3.8849×10^{-12}
Wood	50	700	0.95	0.90	22729.3246	2.9451×10^{-11}
Wong	500	700	0.95	0.90	6552.0931	1.6024×10^{-11}

4.8. Conclusiones del capítulo

En este capítulo se presentó el algoritmo DE-NGFRS como una combinación de distintas técnicas y algoritmos descritos en capítulos anteriores.Uno de los aspectos a tomar en cuenta es la manera en la que el algoritmo explora el espacio de búsqueda, porque mediante el algoritmo NGFRS evita un posible estancamiento en mínimos locales y aporta diversidad de soluciones más allá de la media de la población. Debido a esta capacidad de exploración, la velocidad de convergencia del algoritmo es superior a la presentada por los autores [Ongsakul y Ruangpayoongsak, 2001], [Calderón et al., 2008], [Wood y Wollenberg, 1984] y [Wong y Fung, 1993].

Por otro lado, DE aporta el factor de precisión. Mientras la población está llegando a un punto de convergencia, las diferencias entre individuos que maneja DE evitan cualquier tipo de convergencia prematura y continúan proporcionando mejores soluciones. Dada su

codificación y la manera en la que los individuos se mueven en el espacio de búsqueda, DE es capaz de encontrar soluciones altamente precisas.

De los experimentos realizados en este capítulo se puede deducir que DE-NGFRS tiene un comportamiento eficiente y robusto en comparación a otros algoritmos, porque de una serie de experimentos realizados para un conjunto de funciones de prueba, el óptimo global de la función es encontrado constantemente y las soluciones encontradas son altamente precisas.

Los resultados que presenta DE con funciones de penalización están altamente influenciados por el factor de penalización aplicado a las soluciones que violan alguna restricción; para cada problema es necesario cambiar la función de penalización, lo cual no es totalmente práctico y no garantiza obtener buenos resultados. Por otro lado, no es posible analizar completamente los resultados obtenidos por Mathematica, porque no se conoce el mecanismo interno que utiliza para encontrar la solución y sin importar la cantidad de veces que se optimice el problema, siempre reporta la misma solución. Por lo tanto, además de la solución, el único factor a comparar es el tiempo, el cual es considerablemente mayor a los otros algoritmos.

Debido a que el método de validación de soluciones factibles utilizado por DE-NGFRS utiliza la envoltura convexa de Ω, es indispensable calcular primero la envoltura convexa del conjunto, de otro modo el funcionamiento del algoritmo propuesto resulta considerablemente afectado. En el caso de los problemas de ED, el cálculo de la envoltura convexa no es una tarea sencilla, por lo tanto se utiliza el algoritmo propuesto por [Lara et al., 2009].

Capítulo 5

Conclusiones

5.1. Conclusiones generales

En este trabajo de tesis se ha implementado un algoritmo para la solución de problemas de optimización sujetos a restricciones lineales. Para tal fin, es necesario modelar el problema a resolver, incluyendo las restricciones y la función objetivo.

Dependiendo de la función objetivo y de las restricciones, un problema de optimización puede ser resuelto con distintas técnicas o algoritmos y los EAs han probado ser una buena herramienta para la resolución de problemas de optimización lineales y no-lineales, exhibiendo una ventaja sobre los métodos tradicionales basados en gradiente, especialmente para funciones objetivo discontinuas o no-diferenciables.

En el ámbito de los EAs, una forma de manejar las restricciones de un problema es añadiendo funciones de penalización a la función objetivo cuando un individuo se encuentra fuera Ω. Sin embargo, el uso de penalizaciones no es independiente del problema, es decir que para cada problema es necesario plantear una nueva función de penalización, lo cual reduce la funcionalidad de estas técnicas. Adicionalmente el uso de penalizaciones tiende a sesgar la solución, porque la mayor parte del tiempo se desperdicia descartando individuos que violan las restricciones y la población llega a un punto de no convergencia, por lo tanto las soluciones encontradas no siempre son las mejores.

Si se tiene una región de factibilidad y la manera en la que se generan individuos

dentro de esta no es la más adecuada, el proceso de búsqueda evolutiva puede verse afectado desde el momento de la generación de la población inicial, provocando una convergencia no deseada. En este trabajo se presentó un algoritmo llamado NGFRS, que permite generar una población de individuos que llena una área convexa delimitada por la intersección de restricciones lineales. Las mismas ideas de NGFRS han sido utilizadas como complemento al algoritmo DE para resolver problemas de optimización lineales y no-lineales; el algoritmo resultado es llamado DE-NGFRS. DE-NGFRS es un algoritmo híbrido que incorpora las técnicas de evolución utilizadas por DE y la generación de individuos dentro de Ω, los cuales siguen una distribución gaussiana que permite adaptarse a la forma geométrica de Ω, y por lo tanto, el desempeño que presenta DE al momento de resolver problemas se ve altamente mejorado en comparación con otras técnicas.

En problemas del mundo real, como el problema de ED, se encuentran restricciones que representan comúnmente relaciones que han de preservarse entre las variables del problema (energía, conservación del flujo, etc.) o limitaciones físicas (valores de generación mínimos y máximos). En consecuencia, resolver este tipo de problemas no es una tarea sencilla y requiere de una buena estrategía de búsqueda de la solución, como la que presenta DE-NGFRS.

Se ha demostrado empíricamente que DE-NGFRS exhibe un comportamiento superior en comparación a DE con funciones de penalización. Los resultados obtenidos en la solución de los problemas de ED usando DE-NGFRS fueron comparados con los resultados presentados por otros autores; DE- NGFRS ha probado tener más precisión en los resultados y ha mejorado la velocidad de búsqueda.

Por otro lado, existe software comercial como Mathematica que permite resolver problemas de optimización lineales y no-lineales sujetos a restricciones. Sin embargo, para el polinomio de sexto grado, Mathematica no fue capaz de encontrar el óptimo global y en todos los experimentos realizados se quedó estancado en un óptimo local. Adicionalmente, el tiempo que utiliza Mathematica para calcular la solución en todas las funciones de prueba es notablemente mayor al tiempo empleado por DE-NGFRS. Lamentablemente, el funcionamiento utilizado para encontrar la solución por parte de este software es desconocido, lo cual no permitió su estudio y su adecuada comparación.

5.2. Trabajos futuros

De este trabajo pueden surgir distintas líneas de investigación para exploración futura. Como se puede apreciar en la Figura 4.7, cuando se calcula la intersección entre distintas restricciones lineales, Ω es representado geométricamente como un poliedro. La cantidad de vértices y aristas que contenga dependerá de la cantidad de restricciones y de la manera en que estas se intersectan.

Si el número de restricciones es grande, es probable que el conjunto de puntos q que define la envoltura convexa de Ω también sea grande. Esto puede significar un problema, porque validar si una solución es factible o no-factible utilizando el método descrito en la Sección 2.7 consumirá más tiempo de cómputo. A medida que la cantidad de vértices aumenta, más tiempo de cómputo es consumido por el método. El motivo de que esto suceda se debe a dos razones en particular. Primero, recordemos que la validación de soluciones consiste en resolver un sistema de ecuaciones de la forma $Qa = P$, la matriz Q, está compuesta por el conjunto de vértices q y las dimensiones de esta matriz son $N \times M$, donde N es el número de filas y M el número de columnas. De acuerdo con la Ecuación (2.23) es necesario realizar la operación $[Q^T Q]^{-1}$, como parte de la solución del sistema, lo cual resulta en una matriz cuadrada de dimensiones $N \times N$. Por lo tanto, el calculo de la inversa de una matriz de estas dimensiones cuando N es grande consumirá al menos un tiempo de computo del orden $O(n^2)$.

Segundo, cuando se aplica la solución $a = [Q^T Q]^{-1} Q^T P$ el vector a es de dimensiones $1 \times N$, y para determinar si un punto se encuentra dentro de Ω, es necesario revisar que cada valor de a se encuentre entre el intervalo [0,1]. Por lo tanto, si N es grande, por cada punto revisado, es necesario realizar N comparaciones para verificar esta condición. Dado este problema, se podría desarrollar una nueva investigación que permitiera validar puntos cuando N es muy grande.

Una manera de volver más robusto el algoritmo propuesto es la adaptabilidad a resolver problemas de optimización sujetos a restricciones no-lineales. La consecuencia principal que puede generar tener restricciones no-lineales es la posibilidad de que no exista convexidad en el conjunto factible, lo cual invalida totalmente el enfoque utilizado en esta

investigación para generar puntos dentro de Ω y para determinar si un punto se encuentra dentro o fuera de Ω.

Por lo tanto, se propone una investigación que analice las propiedades de los conjuntos resultantes de la intersección entre restricciones no-lineales y que permita determinar cuando un punto se encuentre fuera o dentro de Ω.

Apéndice A

Cálculo de valores y vectores propios

Los eigenvectores y eigenvalores son vectores y números asociados a matrices cuadradas; juntos proveen una eigen-descomposición de una matriz, la cual analiza la estructura de esta matriz. Incluso cuando la eigen-descomposición no existe para todas las matrices cuadradas, tiene una expresión simple en particular para una clase de matrices comúnmente utilizadas en el análisis multivariante, tal como la matriz de correlación, la matriz de covarianza, etc. La eigen-descomposición de este tipo de matrices es importante en la optimización porque se utiliza para encontrar el máximo o el mínimo de funciones que involucran estas matrices.

A.1. Notación y definiciones

Existen diversas maneras de definir a los eigenvalores y a los eigenvectores; la manera más común define un eigenvector de una matriz A como un vector r que satisface la siguiente ecuación:

$$Ar = \lambda r \tag{A.1}$$

Esta ecuación involucra dos incógnitas λ y r. Al número λ se le llama *eigenvalor* de la matriz A, y el vector r es el *eigenvector* asociado. Notese que la Ecuación (A.1) es una

ecuación no-lineal; λ multiplica a r. Si pudiéramos encontrar λ, entonces la ecuación para r sería lineal. De hecho, podemos escribir λI r y cambiar el término al lado izquierdo como se muestra en la Ecuación (A.2):

$$(A - \lambda I)r = 0 \tag{A.2}$$

El número λ es un eigenvalor de A si y solo si $det(A\text{-}\lambda I) = 0$. Entonces ahora el cálculo de los eigenvalores se convierte en encontrar las raíces donde el determinante es cero. Para cada eigenvalor encontrado, resolver la Ecuación (A.2). Debido a que el determinante es cero, entonces existen otras soluciones diferentes a $r = 0$. Esos son los eigenvectores.

Tradicionalmente, ponemos el conjunto de eigenvectores de A en una matriz denotada R. Cada columna de R es un eigenvector de A. Los eigenvalores se encuentran en una matriz diagonal Λ, donde los elementos diagonales nos dan los eigenvalores (y los otros valores son ceros). Entonces podemos reescribir la Ecuación (A.1) como la Ecuación (A.3)

$$\begin{aligned} AR &= \Lambda R \\ A &= R\Lambda R^{-1} \end{aligned} \tag{A.3}$$

A.2. Descomposición de la matriz de covarianza

La matriz de covarianza Σ no es solo una manera conveniente de mostrar números. Como una matriz, tiene distintas propiedades importantes que se derivan del hecho de que la matriz de covarianza siempre es definida semi-positiva. La eigen-descomposición de esta matriz siempre existe, y tiene una forma en particular. La matriz se dice que es positiva semidefinida cuando se puede obtener el producto de la matriz por su transpuesta. Esto implica que la matriz también es simétrica. Así que formalmente, la matriz Σ es positiva semidefinida si puede obtenerse mediante:

$$A = XX^T \tag{A.4}$$

para una cierta matriz X (que contiene números reales). Otra de las características que posee la matriz de covarianza es que sus eigenvalores son siempre positivos o nulos, y eso

indica que los eigenvectores son ortogonales cuando cada uno de los eigenvalores son diferentes. Así mismo los eigenvectores están compuestos de valores reales (estas últimas dos propiedades son consecuencia de la simetría de la matriz [Strang, 1988]). Porque los eigenvectores que corresponden a diferentes eigenvalores son ortogonales, es posible almacenar todos los eigenvectores en una matriz ortogonal [1]. Esto implica la ecuación A.5

$$R^{-1} = R^T \tag{A.5}$$

Por lo tanto, a partir de (A.3) y (A.5), podemos obtener la Ecuación (A.6).

$$\Sigma = R\Lambda R^T \tag{A.6}$$

donde $R^T R = I$ son eigenvectores normalizados. De esta manera podemos descomponer la matriz de covarianza para utilizarla en el algoritmo NGFRS.

[1] Una matriz ortogonal es cuando el producto de esta matriz por su transpuesta es una matriz diagonal

Apéndice B

Solución por mínimos cuadrados

Cada vez que se necesita validar si un número aleatorio se encuentra dentro de la región de factibilidad es necesario resolver un sistema de ecuaciones de la forma $Ax = b$ de $n \times m$ donde n es el número de columnas y m es el número de filas o ecuaciones. Este cálculo es importante debido a que nos brinda la información necesaria para tomar en cuenta un número o descartarlo.

Un sistema $Ax = b$ puede o no puede tener solución. Si b no se encuentra en el espacio columna de $\mathcal{R}(A)$, el sistema es inconsistente y la eliminación gaussiana falla [Strang, 1988, pág. 153].
En la práctica, las ecuaciones inconsistentes surgen y tienen que ser resueltas. Una posibilidad es determinar x como parte del sistema e ignorar el resto; esto es difícil de justificar si todas las m ecuaciones vienen de la misma fuente, sin embargo en vez de esperar que no exista ningun error en algunas ecuaciones y grandes errores en otras, es mucho mejor escoger x tal que minimice el error promedio en las m ecuaciones. Existen muchas maneras de definir tal promedio, pero la más conveniente es la suma de los cuadrados.
Si existe una solución a $Ax = b$, el error mínimo será $\mathbb{E} = 0$.

La manera más rápida de resolver $Ax = b$ cuando no tiene una solución directa, es a través de la multiplicación A^T, lo cual produce la matriz $A^T A$ de coeficientes cuadrados y simétrica [Strang, 1988].

La solución por mínimos cuadrados a un sistema inconsistente $Ax = b$ de m ecuaciones con n incógnitas satisface la siguiente ecuación:

$$A^T Ax = A^T b \tag{B.1}$$

Si las columnas de A son linealmente independientes, entonces $A^T A$ es cuadrada, simétrica e invertible, y la solución al sistema será dada por la Ecuación (B.2)

$$x = (A^T A)^{-1} A^T b \tag{B.2}$$

Referencias

[Al-Sultan y Al-Fawzan, 1997] Al-Sultan, K. S. y Al-Fawzan, M. (1997). A tabu search hooke and jeeves algorithm for unconstrained optimization. *European Journal of Operations Research*, pages 198–208.

[Ali y Storey, 1994] Ali, M. M. y Storey, C. (1994). Modified controlled random search algorithms. *International Journal of Computer Mathematics*, 53:229–235.

[Bagdadhi, 2007] Bagdadhi, Z. K. (2007). *Diferential Evolution algorithms for constrained global optimization*. PhD thesis, University of Witwaterstand.

[Boyd y Vandenberghe, 2004] Boyd, S. y Vandenberghe, L. (2004). *Convex Optimization*. Cambridge.

[Calderón et al., 2008] Calderón, F., Esquivel, C. R. F., y Flores, J. J. (2008). A constraint-handling genetic algorithm to power economic dispatch. *MICAI 2008: Advances in Artificial Intelligence Lecture Notes in Computer Science Mexico City, October 2008*, pages 371–381.

[Chong y Zak, 2001] Chong, E. K. y Zak, S. H. (2001). *An Introduction to Optimization*. Wiley.

[Chowdhury y Rahman, 1990] Chowdhury, B. H. y Rahman, S. (1990). A review of recent advances in economic dispatch. In *IEEE Transactions, Power Syst.,*, pages 1248–1259.

[de Berg et al., 2010] de Berg, M., Cheong, O., Kreveld, M. V., y Overmars, M. (2010). *Computational Geometry*. Springer.

[Devore, 2008] Devore, J. L. (2008). *Probabilidad y Estadística*. Cengage Learning.

[Dorigo y Caro, 1999] Dorigo, M. y Caro, D. (1999). *The ant colony optimization metaheuristic*. McGraw-Hill.

[E. Zitzler, 2000] E. Zitzler, K. Deb, y. L. T. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Massachusetts Institute of Technology*.

[Fischer, 2010] Fischer, H. (2010). *A History of the Central Limit Theorem: From Classical to Modern Probability Theory*. Springer.

[Goldberg, 1989] Goldberg, D. E. (1989). *Genetic Algorithms in search, Optimization and machine learning*. Addison Wesley.

[Grinstead y Snell, 1997] Grinstead, C. M. y Snell, J. L. (1997). *Introduction to Probability*. American Mathematical Society.

[Guillen, 2007] Guillen, P. (2007). Envoltura convexa. Recurso en Internet: http://pier.guillen.com.mx/algorithms/07-geometricos/07.7-envoltura_convexa.htm.

[Haupt y Haupt, 2004] Haupt, R. L. y Haupt, S. E. (2004). *Practical Genetic Algorithms*. Wiley.

[Hillier y Lieberman, 2006] Hillier, F. S. y Lieberman, G. J. (2006). *Introducción a la Investigación de Operaciones*. McGraw-Hill.

[Kennedy y Gentle, 1980] Kennedy y Gentle (1980). *Statistical Computing*. Marcel Dekker.

[Kleywegt y Saphiro, 2000] Kleywegt, A. J. y Saphiro, A. (2000). *Stochastic optimization*. Georgia Institute of Technology.

[Lara et al., 2009] Lara, C., Flores, J. J., y Calderón, F. (2009). On the hyperbox-hyperplane intersection problem. *Infocomp Computer Journal Computer Science. Brazil December 2009*, 8(4):21–27.

[Lawler y Wood, 1966] Lawler, E. y Wood, D. (1966). *Branch and bound methods: A survey*. Operations Research.

[Lawson, 1986] Lawson, C. L. (1986). Properties of n-dimensional triangulations. *Computer-Aided Geometric Design*, 3:231–246.

[Lee y El-Sharkawi, 2008] Lee, K. Y. y El-Sharkawi, M. A. (2008). *Modern Heuristic Optimization Techniques, Theory and Applications to Power Systems*. Wiley-Interscience.

[Leffingwell, 2011] Leffingwell, D. (2011). *Agile software requeriments*. Addison Wesley.

[Lejarza y Lejarza, 2007] Lejarza, J. M. y Lejarza, I. M. (2007). Distribución normal. Recurso en Internet: http://www.uv.es/ceaces/pdf/normal.pdf.

[Michalewicz, 1995] Michalewicz, Z. (1995). A survey of constraint handling techniques in evolutionary computation methods. *Evolutionary Programming*, pages 135–155.

[Michalewicz et al., 1994] Michalewicz, Z., T.D., L., y Swaminathan, S. (1994). Evolutionary operators for continuous convex parameter spaces. *Proceedings of the 4th Annual Conference on Evolutionary Computation*, pages 84–97.

[Montes y Coello, 1998] Montes, E. M. y Coello, C. A. C. (1998). A survey of constraint-handling techniques based on evolutionary multiobjective optimization. *Evolutionary Computation Group*.

[Nocedal y Wright, 1999] Nocedal, J. y Wright, S. J. (1999). *Numerical Optimization*. Springer.

[Ongsakul y Ruangpayoongsak, 2001] Ongsakul, W. y Ruangpayoongsak, N. (2001). Constrained dynamic economic dispatch by simulated annealing/genetic algorithms. In *Power Industry Computer Applications, 2001. PICA 2001. Innovative Computing for Power - Electric Energy Meets the Market. 22nd IEEE Power Engineering Society International Conference on*, pages 207 – 212.

[O'Rourke, 1998] O'Rourke, J. (1998). *Computational Geometry in C*. Cambridge University Press.

[Papoulis y Pillai, 1999] Papoulis, A. y Pillai, S. U. (1999). *Probability, Random Variables and Stochastic Processes*. McGraw-Hill.

[Pinter, 1996] Pinter, J. D. (1996). *Global optimization in action: Continuous and Lipschitz optimization algorithms: Implementations and Aplications*. Kluwer Academic Publishers.

[Rockafellar y Wets, 1998] Rockafellar, R. T. y Wets, R. J.-B. (1998). *Variational Analysis*. Springer.

[Santos y Marianib, 2007] Santos, L. y Marianib, V. C. (2007). Improved differential evolution algorithms for handling economic dispatch optimization with generator constraints. *Energy Conversion and Management*, 48:1631–1639.

[Sivanandam y Deepa, 2008] Sivanandam, S. y Deepa, S. (2008). *Introduction to genetic algorithms*. Springer.

[Solodovnikov, 1980] Solodovnikov, A. S. (1980). *Systems of linear inequalities*. The University of Chicago Press.

[Spall, 2004] Spall, J. C. (2004). *Stochastic optimization*. Springer Heidelberg.

[Stevenson, 1982] Stevenson, W. (1982). *Elements of Power System Analysis*. McGraw Hill.

[Storn y Price, 1995] Storn, R. M. y Price, K. V. (1995). Differential evolution- a simple and efficient adaptative scheme for global optimization over continuous spaces. *Global optimization*.

[Storn y Price, 1997] Storn, R. M. y Price, K. V. (1997). Differential evolution- a simple and efficient heuristic for global optimization over continuous spaces. *Global optimization*, 11:341–359.

[Storn y Price, 2005] Storn, R. M. y Price, K. V. (2005). *Differential Evolution A practical Aproach to Global Optimization*. Springer.

[Strang, 1988] Strang, G. (1988). *Linear Algebra and its Applications*. Books Cole/Thompson Learning.

[Torn y Zilinskas, 1989] Torn, A. y Zilinskas, A. (1989). *Global Optimization*. Springer-Verlage.

[Trelea, 2003] Trelea, I. (2003). The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information Processing Letters*, 85(6):317–325.

[Wolpert y Macready, 1997] Wolpert, D. H. y Macready, W. G. (1997). No free lunch teorems for optimization. *IEEE Transactions on Evolutionary Computation*, pages 67–82.

[Wong y Fung, 1993] Wong, K. y Fung, C. (1993). Simulated annealing based economic dispatch algorithm. In *Generation, Transmission and Distribution, IEEE Proceedings C*, volume 140, pages 509 – 515.

[Wood y Wollenberg, 1984] Wood, A. y Wollenberg, B. (1984). *Power Generation, Operation, and Control*. John Wiley and Sons Inc.

[Zhang y Baritompa, 1993] Zhang, B., W. G. y Baritompa (1993). *Multidimensional Bisection: the performance and the context*. Journal of global optimization.