



CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS
DEL INSTITUTO POLITÉCNICO NACIONAL

Unidad Zacatenco

Departamento de Computación

Mecanismo de selección y control de una hiperheurística basada en
Evolución Diferencial para optimización en espacios restringidos

Tesis que presenta

José Carlos Villela Tinoco

para obtener el Grado de

Maestro en Ciencias

en la Especialidad de

Computación

Director de la Tesis

Dr. Carlos A. Coello Coello

México, D.F.

Octubre 2010

Agradecimientos

Al término de una nueva etapa de mi vida, quiero expresar mi profundo agradecimiento a quienes con su ayuda, apoyo y comprensión me alentaron a lograr esta hermosa realidad: Mis padres.

A mi hermana Karla quien me apoyó y colaboró en mi formación y que ahora comienza su carrera profesional.

A mi amigo Erick, con quien por dos años más compartimos buenos momentos.

A mis compañeros de sala de trabajo con los que coniví formando una gran amistad en cada uno de ellos.

Al Dr. Carlos A. Coello Coello por dejarme trabajar bajo su dirección y brindarme todo el apoyo necesario.

A Sofi por su gran apoyo y amistad la cual sobrepasa cualquier obligación laboral.

A mis sinodales, Dr. Luis Gerardo de la Fraga y al Dr. Gregorio Toscano Pulido, por revisar mi tesis y que gracias a sus comentarios me permitió obtener un mejor trabajo.

A cada uno de los doctores que me impartieron clase en un año lleno de esfuerzo y que gracias a su apoyo siempre salí adelante.

Al CINVESTAV por brindarme la oportunidad de cumplir mis estudios en esta gran institución.

Al CONACYT por el apoyo económico brindado a lo largo de estos dos años, permitiendo de esta manera poder cumplir mis estudios.

Resumen

El uso de heurísticas para la solución de problemas de alta complejidad se ha vuelto muy popular en años recientes, debido a su flexibilidad, eficacia y facilidad de uso. Sin embargo, esta popularidad ha propiciado a la vez el desarrollo de una amplia gama de heurísticas para búsqueda y optimización, muchas de las cuales tienen inspiración biológica (p.ej., los algoritmos evolutivos, la optimización mediante cúmulos de partículas, los sistemas inmunes artificiales, la optimización mediante colonias de hormigas, etc.). Tal diversidad de métodos suele provocar confusiones entre aquellos interesados en abordar un problema en particular y son muy pocos los estudios que intenten identificar ventajas o desventajas de una heurística con respecto a otras, en un problema (o clase de problemas) en particular.

Un enfoque alternativo para abordar esta problemática es el uso de las hiperheurísticas. El término *hiperheurística* fue introducido por primera vez por Peter Cowling, y se define como un enfoque que opera en un alto nivel de abstracción y gestiona la elección de una heurística de bajo nivel para ser aplicada en un momento dado, dependiendo de las características de la región del espacio de búsqueda que se esté explorando. La idea clave en las hiperheurísticas es utilizar un conjunto de heurísticas conocidas para transformar el estado de un problema. Esto implica que el usuario no tiene que elegir una heurística en particular para resolver su problema, sino que puede usar una combinación de varias de ellas, buscando combinar sus bondades y compensar las desventajas que algunas de ellas pudieran tener.

Pese a sus obvias ventajas, las hiperheurísticas han sido usadas principalmente en optimización combinatoria, y existe poco trabajo en torno a su aplicación en problemas de optimización en los cuales las variables se expresan mediante números reales. Si además el problema tiene restricciones, el uso de hiperheurísticas se torna aún más escaso. En esta tesis se aborda precisamente este tipo de problemas y se propone una nueva hiperheurística basada en variantes de la Evolución Diferencial (ED) para resolver problemas de optimización con restricciones en los cuales las variables son números reales. La principal contribución de esta tesis es un nuevo mecanismo de selección para coordinar los diferentes modelos de Evolución Diferencial incorporados a la hiperheurística propuesta.

El nuevo enfoque hiperheurístico aquí propuesto se compara con respecto al algoritmo denominado *Constrained Differential Evolution* (CDE) o “Evolución Diferencial Restringida”, usando un conjunto de problemas de prueba estándar tomados de la literatura especializada. Los resultados obtenidos indican que la hiperheurística propuesta constituye una alternativa viable para combinar distintos modelos de la Evolución Diferencial para resolver problemas de optimización con restricciones en espacios de búsqueda continuos.

Abstract

The use of heuristics for the solution of high complexity problems has become very popular in recent years, mainly because of their flexibility, efficacy and ease of use. However, this popularity has simultaneously fostered the development of a wide variety of heuristics for search and optimization, many of which have a biological inspiration (e.g., evolutionary algorithms, particle swarm optimization, artificial immune systems, ant colony optimization, etc.). Such a diversity of methods tends to confuse those interested in tackling a particular problem, and there are very few studies that attempt to identify advantages or disadvantages of a metaheuristic with respect to others, in a particular problem (or class of problems).

An alternative approach to tackle this problem is the use of hyperheuristics. The term *hyperheuristic* was originally introduced by Peter Cowling, and is defined as an approach that operates at a high level of abstraction and manages the selection of a low level metaheuristic to be applied at a certain moment, depending on the characteristics of the region of the search space which is being currently explored. The key idea when using hyperheuristics is to use a set of known heuristics in order to transform the state of a problem. This implies that the user does not have to choose one specific metaheuristic to solve his/her problem, but can instead use a combination of several of them, aiming to combine their advantages and to compensate the disadvantages that some of them may have.

In spite of their obvious advantages, hyperheuristics have been mainly used in combinatorial optimization, and there is little work regarding their use in optimization problems in which the decision variables are expressed by real numbers. Furthermore, if the problem has constraints, the use of hyperheuristics is even more scarce. In this thesis, we precisely deal with this type of problems, and we propose a new hyperheuristic based on Differential Evolution (DE) variants to solve constrained optimization problems in which the decision variables are real numbers. The main contribution of this thesis is a new selection mechanism to coordinate the different Differential Evolution models incorporated into the proposed hyperheuristic.

The new hyperheuristic approach introduced here is compared with respect to an algorithm called *Constrained Differential Evolution* (CDE), using a set of standard test problems taken from the specialized literature. The obtained results indicate that the proposed hyperheuristic constitutes a viable alternative to combine different Differential Evolution models with the aim of solving constrained optimization problems in continuous search spaces.

Índice general

Lista de figuras	XIII
Lista de tablas	XV
Lista de algoritmos	XVII
1. Introducción	1
1.1. Objetivos	2
1.2. Contribuciones	2
1.3. Organización del documento	2
2. Optimización	5
2.1. Introducción	5
2.2. Problema de optimización	5
2.3. Computación evolutiva	7
2.3.1. Algoritmos evolutivos	7
2.3.2. Principales paradigmas	9
Programación evolutiva	9
Estrategias evolutivas	9
Algoritmos genéticos	10
2.4. Evolución Diferencial	11
2.4.1. Algoritmo de la ED	11
2.4.2. Componentes de la ED	12
Mutación	12
Recombinación	12
Selección	13
2.4.3. Variantes de la ED	14
3. Hiperheurísticas	17
3.1. Introducción	17
3.2. Marco de una hiperheurística	19
3.3. Enfoques modernos en las hiperheurísticas	20
3.3.1. Hiperheurísticas basadas en selección aleatoria	20
3.3.2. Hiperheurísticas codiciosas	22
3.3.3. Hiperheurísticas basadas en meta-heurísticas	22
Hiperheurísticas basadas en algoritmos genéticos	23

Hiperheurísticas basadas en recocido simulado y búsqueda tabú . . .	23
3.3.4. Hiperheurísticas con aprendizaje	25
4. Manejo de Restricciones en los AE	27
4.1. Introducción	27
4.2. Funciones de penalización	28
4.2.1. Pena de muerte	29
4.2.2. Penalizaciones estáticas	29
4.2.3. Penalizaciones dinámicas	30
4.2.4. Penalizaciones adaptativas	31
4.3. Operadores y representaciones especiales	32
4.4. Separación de restricciones y objetivos	33
4.5. Métodos híbridos	33
4.6. Otros Métodos	34
4.6.1. Jerarquización estocástica	34
4.6.2. Evolución diferencial restringida	35
5. Propuesta de Hiperheurística Basada en ED	37
5.1. Modelo propuesto para espacios sin restricciones	37
5.1.1. Estudio comparativo	37
Población inicial	39
Resultados	42
5.1.2. Mecanismo de selección de heurísticas	43
5.2. Manejo de restricciones	50
6. Resultados Experimentales	55
6.1. Diseño experimental	55
6.1.1. Análisis estadístico	56
6.1.2. Gráficos de caja	56
6.1.3. Métodos estadísticos de cómputo intensivo	57
Método de <i>bootstrap</i>	58
Intervalos de confianza	59
6.2. Resultados experimentales	59
6.2.1. Parámetros de control	60
6.2.2. Resultados obtenidos	61
6.2.3. Comportamiento de las heurísticas de bajo nivel	68
7. Conclusiones y Trabajo Futuro	71
A. Funciones de Prueba sin Restricciones	75
B. Resultados Experimentales para Optimización sin Restricciones	79
C. Funciones de Prueba con Restricciones	83

D. Gráficas de Resultados	91
D.1. Histogramas de resultados	91
D.2. Gráficos de caja	95
D.3. Comportamiento de heurísticas de bajo nivel	99
D.4. Gráficas de convergencia	101
Bibliografía	103

Lista de figuras

2.1.	El mínimo de la función $f(\vec{x})$ es igual al máximo de la función $-f(\vec{x})$	6
2.2.	Ejemplificación de la región factible e infactible para un problema con dos variables de decisión.	7
2.3.	Diagrama de flujo de la Evolución Diferencial	14
2.4.	Ejemplo de los vectores de prueba de la Evolución Diferencial	15
2.5.	Recombinación discreta binomial y exponencial de la Evolución Diferencial	15
3.1.	Marco de una hiperheurística	18
5.1.	Comparativa de la entropía entre la secuencia de Halton y una distribución uniforme para: a) una variable aleatoria $X = \{1, 2, \dots, 100\}$ con distribución $p(x)$ y b) dos variables aleatorias X, Y con distribución $p(x, y)$ donde $X = Y$	41
5.2.	Trescientos puntos creados con: a) una distribución uniforme y b) la secuencia de Halton en \mathbb{R}^2 en el intervalo $[0, 1]$	42
5.3.	Promedio de mejoras por cada variante para: a) $D = 10$, b) $D = 30$, c) $D = 50$ y $D = 100$ variables de decisión con $R = 0.2$ para las cinco funciones de prueba, donde el eje x representa el índice de la variante de la ED de acuerdo a la tabla 5.1, de la pág. 39.	44
5.4.	Promedio de mejoras por cada variante para: a) $D = 10$, b) $D = 30$, c) $D = 50$ y $D = 100$ variables de decisión con $R = 0.8$ para las cinco funciones de prueba, donde el eje x representa el índice de la variante de la ED de acuerdo a la tabla 5.1, de la pág. 39.	45
5.5.	Ejemplo hipotético de los tres tipos de ruletas con cuatro heurísticas de bajo nivel: a) Ruleta original, b) Ruleta con inicio aleatorio y c) Ruleta con permutación	47
5.6.	Ejemplificación de los dos tipos de dirección de búsqueda del operador de mutación de la ED: a) hacia la zona factible \mathcal{F} y b) hacia la zona infactible	51
5.7.	Diagrama de flujo del proceso de división de la población	51
6.1.	Diagrama de caja indicando sus principales componentes	57
6.2.	Histograma de frecuencias de las soluciones para el problema g03 para: a) la muestra original y b) aplicando el método de <i>bootstrap</i> a la muestra original	64
6.3.	Gráficas de caja de las soluciones para el problema g03 para: a) la muestra original y b) aplicando el método de <i>bootstrap</i> a la muestra original	65
6.4.	Porcentaje de mejoras por cada variante para cada uno de los modelos de selección de heurísticas para las funciones g01 y g02	69

D.1. Porcentaje de mejoras por cada variante para cada uno de los modelos de selección de heurísticas para las funciones $g01 - g06$	99
D.2. Porcentaje de mejoras por cada variante para cada uno de los modelos de selección de heurísticas para las funciones $g07 - g13$	100
D.3. Gráficas de convergencia para las funciones $g01 - g06$	101
D.4. Gráficas de convergencia para las funciones $g07 - g13$	102

Lista de tablas

2.1. Principales modelos de la Evolución Diferencial	16
5.1. Conjunto de heurísticas de bajo nivel	39
5.2. Comparación de la aptitud media de 100 ejecuciones independientes con un valor de $R = 0.2$	42
5.3. Comparación de la aptitud media de 100 ejecuciones independientes con un valor de $R = 0.8$	42
6.1. Características de las 13 funciones utilizadas para optimización en espacios restringidos	60
6.2. Parámetros de control empleados en las pruebas experimentales para la hiperheurística	61
6.3. Parámetros de control empleados en las pruebas experimentales para el algoritmo CDE	61
6.4. Estadísticas con respecto a la solución obtenida $f(\vec{x})$ para CDE y mecanismo aleatorio descendente	62
6.5. Estadísticas con respecto a la solución obtenida $f(\vec{x})$ para R variable y R1	62
6.6. Estadísticas con respecto a la solución obtenida $f(\vec{x})$ para R2 y R3	62
6.7. Porcentaje de éxitos, mejor y peor solución de $f(\vec{x})$ para CDE y el mecanismo aleatorio descendente	63
6.8. Porcentaje de éxitos, mejor y peor solución de $f(\vec{x})$ para R variable y R1	63
6.9. Porcentaje de éxitos, mejor y peor solución de $f(\vec{x})$ para R2 y R3	63
6.10. Intervalos de confianza de la solución obtenida $f(\vec{x})$ para CDE, aleatorio descendente y R variable	66
6.11. Intervalos de confianza de la solución obtenida $f(\vec{x})$ para los tres tipos de ruleta	66
6.12. Estadísticas a través de un procedimiento de <i>bootstrap</i> para CDE y mecanismo aleatorio descendente	67
6.13. Estadísticas a través de un procedimiento de <i>bootstrap</i> para R variable y R1	67
6.14. Estadísticas a través de un procedimiento de <i>bootstrap</i> para R2 y R3	68
6.15. Promedio y mínimo de generaciones requeridas para el algoritmo CDE y los diversos mecanismos de selección	69
6.16. Comparativa entre el mecanismo de selección R2 y la Jerarquización Estocástica	70
B.1. Resultados experimentales para f_1	79
B.2. Resultados experimentales para f_2	80

B.3. Resultados experimentales para f_3	80
B.4. Resultados experimentales para f_4	80
B.5. Resultados experimentales para f_5	81
B.6. Resultados experimentales para las funciones f_6 a f_9 usando el mecanismo de selección de la ruleta con inicio aleatorio.	81
D.1. Histogramas de frecuencias de las soluciones de las 100 ejecuciones independientes para los algoritmos CDE, el mecanismo de selección aleatoria descendente y nuestra propuesta R2 como mecanismo de selección, la cual fue la que mejor desempeño tuvo.	94
D.2. Gráficas de caja de las soluciones de las 100 ejecuciones independientes para los algoritmos CDE, el mecanismo de selección aleatoria descendente y nuestra propuesta R2 como mecanismo de selección, la cual fue la que mejor desempeño tuvo.	98

Lista de algoritmos

1.	Esquema general de un algoritmo evolutivo	8
2.	Esquema general de la programación evolutiva	9
3.	Estrategia evolutiva de dos miembros (1+1)-EE	10
4.	Esquema general de un algoritmo genético	10
5.	Algoritmo de Evolución Diferencial ED/rand/1/bin	13
6.	Pseudocódigo de un algoritmo hiperheurístico básico	18
7.	Algoritmo de la jerarquización estocástica	35
8.	Hiperheurística con mecanismo de selección aleatorio descendente	38
9.	Secuencia de Halton	40
10.	Pseudocódigo de la ruleta propuesto por DeJong	46
11.	Mecanismo de selección de la hiperheurística propuesta	48
12.	Propuesta de hiperheurística basada en variantes de la ED	49
13.	Pseudocódigo del algoritmo de la ED para espacios restringidos	52
14.	Esquema general de un procedimiento de <i>bootstrap</i>	59

Introducción

Por **optimización** se entiende al acto de obtener el mejor resultado posible dadas ciertas circunstancias [59]. La optimización tiene una enorme utilidad en muchas ramas del conocimiento. En la actualidad, no existe ningún método de optimización para resolver todo tipo de problemas, por lo que a lo largo de los años se han desarrollado diversos métodos para resolver distintas clases de problemas.

El término **heurística** proviene del griego *heuriskein* (encontrar, descubrir, hallar). Desde un punto de vista científico, el término fue usado por primera vez por el matemático G. Polya quien lo empleó en su libro *How to solve it?* en 1957. Reeves [61] indica que una técnica heurística es un método que busca encontrar una solución casi óptima, a un costo razonable de cálculo sin ser capaz de garantizar la obtención del óptimo global. Por desgracia, puede incluso no ser posible establecer que tan cerca se encuentra una solución del óptimo global.

Los métodos de optimización se pueden clasificar de acuerdo a métodos **clásicos** en donde encontramos el método simplex, de Newton, Cauchy, etc.; y los métodos **metaheurísticos** en los que se incluyen los algoritmos evolutivos, el recocido simulado y cualquier método de búsqueda heurística. De forma muy general se puede decir que los métodos clásicos buscan y garantizan encontrar el valor óptimo a un problema pero sólo son aplicables a ciertas clases de problemas, mientras que los métodos heurísticos no garantizan encontrar el óptimo, pero tienen amplia aplicabilidad.

Sin embargo, no existe ningún método determinista que garantice converger siempre al valor óptimo de una función en problemas no lineales. Por esta razón, los mecanismos heurísticos han tenido un gran auge en los últimos años para dar solución a problemas muy difíciles. El constante incremento del poder de procesamiento de las computadoras personales ha hecho posible el desarrollo de metaheurísticas más elaboradas y eficaces.

En esta tesis se aborda un enfoque relativamente reciente denominado *hiperheurística*. Las hiperheurísticas combinan un conjunto de heurísticas y cuentan con un mecanismo de control que las coordine, buscando aplicar cada heurística en el momento más indicado. En este trabajo se propone el diseño e implementación de una hiperheurística para resolver problemas de optimización, cuyo conjunto de heurísticas de bajo nivel está conformado por variantes del algoritmo denominado Evolución Diferencial.

La **Evolución Diferencial** (ED) es un algoritmo poblacional de búsqueda directa el cual emplea combinaciones lineales de los individuos para generar nuevas soluciones. Existen en la literatura diferentes modelos de la ED y el objetivo principal de este trabajo es estudiar cada uno de ellos para identificar sus características principales y poder diseñar un mecanismo que permita coordinarlos a partir de ellas.

1.1. Objetivos

El objetivo general de esta tesis ha sido diseñar e implementar un mecanismo de control que coordine un cierto conjunto de algoritmos para resolver varios tipos de problemas, adaptándose a las características de cada uno de ellos. Para poder llevar a cabo el objetivo general, fue necesario cumplir los siguientes objetivos particulares:

- Estudiar cada una de las variantes originales de la Evolución Diferencial.
- Identificar los problemas y las ventajas de cada variante de la Evolución Diferencial que se puedan aplicar a cada tipo de problema.
- Diseñar un mecanismo de control que permita combinar cada una de las variantes a fin de resolver los problemas de manera más eficiente.

1.2. Contribuciones

Las aportaciones de este trabajo de tesis son:

1. Un estudio comparativo entre diversos modelos de Evolución Diferencial usando funciones estándar de la literatura especializada.
2. Una nueva hiperheurística basada en variantes de la Evolución Diferencial, tanto para problemas sin restricciones como en la presencia de éstas.
3. Dicha hiperheurística es evaluada usando funciones de prueba estándar reportadas en la literatura especializada. Los resultados se comparan con respecto a los del algoritmo de la Evolución Diferencial Restringida y a los de la hiperheurística con mecanismo de selección aleatorio descendente.

1.3. Organización del documento

El trabajo de esta tesis está organizado de la siguiente forma:

Capítulo 2. Optimización: Se da una descripción sobre los conceptos básicos de optimización y el planteamiento del problema general de optimización. También se da una introducción breve sobre el área de Computación Evolutiva indicando sus principales paradigmas. Finalmente, se da una descripción detallada del algoritmo de la Evolución Diferencial indicando sus principales componentes y modelos existentes.

Capítulo 3. Hiperheurísticas: Se presentan los orígenes del concepto de hiperheurística, el marco general que las rige y los diversos enfoques que han surgido en esta área relativamente nueva dentro de la computación evolutiva.

Capítulo 4. Manejo de Restricciones en los AEs: En este capítulo se estudian las principales técnicas para el manejo de restricciones que existen dentro de los Algoritmos Evolutivos (AEs), entre las que se incluyen: las funciones de penalización en sus diversas formas, operadores y representaciones especiales, separación de restricciones y objetivos y los métodos híbridos. Posteriormente, se habla sobre dos de los mecanismos de manejo de restricciones más representativos del área que son la Jerarquización Estocástica y la Evolución Diferencial Restringida.

Capítulo 5. Propuesta de Hiperheurística Basada en ED: En este capítulo se lleva a cabo el estudio comparativo de los modelos de la Evolución Diferencial para, posteriormente detallar la propuesta completa de la hiperheurística basada en Evolución Diferencial, tanto para problemas sin restricciones como para optimización en espacios restringidos, a partir de las características identificadas en el estudio hecho.

Capítulo 6. Resultados Experimentales: Se presentan los resultados obtenidos por la hiperheurística propuesta en esta tesis y un estudio comparativo empleando un conjunto de funciones de prueba estándar utilizadas en la optimización en espacios restringidos.

Capítulo 7. Conclusiones y Trabajo Futuro: Se muestran las conclusiones basadas en los resultados obtenidos así como las posibles líneas de investigación a seguir.

Apéndice A. Funciones de Prueba sin Restricciones: Se listan las funciones de prueba sin restricciones empleadas en este trabajo.

Apéndice B. Resultados Experimentales para Optimización sin Restricciones: En este apéndice se muestran los resultados experimentales obtenidos con las funciones de pruebas para espacios sin restricciones adoptadas.

Apéndice C. Funciones de Prueba con Restricciones: Se presentan las descripciones de los problemas de prueba empleados en la optimización numérica con restricciones.

Apéndice D. Gráficas de Resultados: Para el algoritmo CDE, el enfoque hiperheurístico con mecanismo de selección aleatorio descendente y nuestro mecanismo propuesto, se listan los histogramas de frecuencia, los gráficos de caja y el comportamiento de cada modelo de Evolución Diferencial dentro de un esquema hiperheurístico. Finalmente, se presentan las gráficas de convergencia para cada una de las funciones de prueba.

Optimización

En este capítulo se dará una introducción sobre los conceptos básicos de optimización así como el planteamiento del problema general de optimización. También se presentará una breve descripción sobre el área de computación evolutiva, señalando sus principales paradigmas y se dará una descripción detallada sobre el algoritmo denominado Evolución Diferencial, haciendo énfasis en los componentes y las diversas variantes existentes de esta heurística.

2.1. Introducción

La optimización es una de las áreas de las matemáticas que más aplicaciones tiene en la vida real. Los primeros problemas de optimización que se estudiaron y lograron resolver, son los llamados *problemas de programación lineal* y para los cuales existen actualmente diversos métodos matemáticos que aseguran encontrar el óptimo.

También existen algunos métodos para resolver problemas de optimización no lineal, pero requieren información adicional del problema, v.g., la mayoría requiere la primera derivada de la función objetivo. Sin embargo, no en todas las clases de problemas está disponible dicha información ya que la función objetivo no siempre es diferenciable o existe una discontinuidad en el punto óptimo, razón por la cual la derivada en dicho punto es inexistente.

Para tales casos, no se conoce ningún algoritmo que asegure encontrar el óptimo en un tiempo polinomial. Debido a lo anterior, para obtener soluciones a estos problemas en tiempos más cortos, lo único que puede hacerse es desarrollar heurísticas, tratando de que se comporten lo mejor posible [60].

2.2. Problema de optimización

Un problema de optimización es un planteamiento a una situación para la cual existen diferentes soluciones posibles. Debido a que el problema de minimizar $f(\vec{x})$ es equivalente al de maximizar $-f(\vec{x})$ (ver fig. 2.1), el problema general de optimización se puede formular como sigue:

Definición 1 *Encontrar los valores de las n variables de decisión $[x_1, x_2, \dots, x_n]^T$ denotadas por el vector $\vec{x} \in \mathcal{S}$, que satisfagan las m condiciones de desigualdad y las p condiciones*

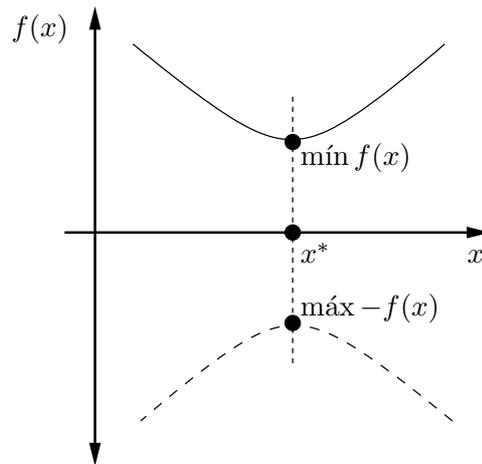


Figura 2.1: El mínimo de la función $f(\vec{x})$ es igual al máximo de la función $-f(\vec{x})$.

de igualdad que optimicen la función objetivo $f(\vec{x})$.

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

$$\begin{aligned} g_i(\vec{x}) &\leq 0, & i &= 1, 2, \dots, m \\ h_j(\vec{x}) &= 0, & j &= 1, 2, \dots, p \end{aligned}$$

De acuerdo a la definición 1, el **espacio de diseño** o **espacio de búsqueda**, denotado por \mathcal{S} , es el espacio cartesiano n -dimensional en donde cada uno de los ejes representa a cada una de las variables de diseño x_i y en el que cada punto sobre él representa una solución posible al problema. El vector \vec{x} es denominado **vector de diseño** y cada x_i ($i = 1, 2, \dots, n$) son valores escalares llamados **variables de decisión** o **variables de diseño**. En la mayoría de los problemas de optimización, las variables de diseño no pueden ser tomadas arbitrariamente, sino que deben satisfacer los requerimientos impuestos por las restricciones de desigualdad $g_i(\vec{x})$ e igualdad $h_j(\vec{x})$. Estas son funciones escalares $g_i : \mathbb{R}^n \rightarrow \mathbb{R}$ y $h_j : \mathbb{R}^n \rightarrow \mathbb{R}$ con $i = 1, 2, \dots, m$ y $j = 1, 2, \dots, p$ y no tienen propiedades especiales.

Definición 2 La región factible $\mathcal{F} \subseteq \mathcal{S}$ representa las soluciones posibles al problema y se define como:

$$\mathcal{F} = \{\vec{x} \in \mathbb{R}^n \mid g_i(\vec{x}) \leq 0, i = 1, 2, \dots, m \text{ y } h_j(\vec{x}) = 0, j = 1, 2, \dots, p\}$$

En otras palabras, al subconjunto de todos los puntos que satisfacen las restricciones del problema se le denomina **región factible** y, por lo tanto cada punto dentro de ésta representa una solución aceptable al problema. El subconjunto de todos los puntos infactibles es llamado **región infactible** (ver fig. 2.2, donde cada g_i representa las restricciones de desigualdad).

Se dice que $g_i(\vec{x})$ representa una **restricción activa** cuando $g_i(\vec{x}) = 0$ en el óptimo; en caso contrario, se denomina **restricción inactiva**. Por definición todas las restricciones de igualdad se consideran activas en todo el espacio de búsqueda.

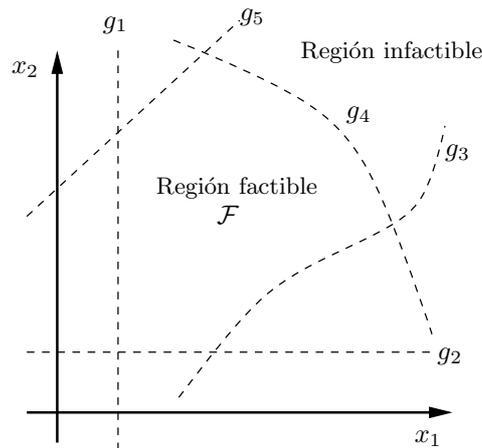


Figura 2.2: Ejemplificación de la región factible e infactible para un problema con dos variables de decisión.

2.3. Computación evolutiva

A lo largo de la historia varios científicos han contribuido al entendimiento del proceso evolutivo. El **Neodarwinismo** es la teoría o corriente científica referente a las teorías de la evolución que mantiene la esencia de la teoría de Charles Darwin sobre el origen de las especies, v.g., la evolución de las especies se produce mediante variaciones aleatorias de los individuos, producto de la selección natural; la teoría seleccionista (o del germoplasma) de August Weismann¹ y las leyes de la genética de Gregor Mendel.

El Neodarwinismo trata de aplicar los conocimientos genéticos al estudio de la evolución (v.g., los mecanismos genéticos que producen como resultado la evolución de una especie). Desde un punto de vista genético, la evolución es el cambio de las frecuencias genéticas de una población a lo largo del tiempo.

2.3.1. Algoritmos evolutivos

El término **computación evolutiva** engloba una serie de técnicas inspiradas biológicamente. Hoy en día existen una gran variedad de algoritmos evolutivos empleados para dar solución a diversos problemas. En términos generales, para simular el proceso evolutivo se requieren los siguientes elementos:

1. **Codificación de las estructuras:** se refiere a cómo representar las características de los “individuos” (i.e., las soluciones posibles al problema de nuestro interés).
2. **Función de aptitud:** básicamente corresponde a la función objetivo o a una variante de ésta.
3. **Operadores:** permiten modificar a los “individuos” (p. ej. cruza y mutación).

¹Teoría que asegura que la selección natural es el único mecanismo capaz de modificar el genotipo.

4. **Mecanismo de selección:** consiste en escoger a los mejores individuos para reproducirse, puede ser determinista o probabilístico.

Por individuo entendemos a una solución potencial del problema, el cual se encuentra representado según las necesidades del mismo. Cada individuo tiene asociado un nivel de aptitud definido por una función de aptitud que indica sus posibilidades de supervivencia. La computación evolutiva involucra métodos que son poblacionales, v.g., que trabajan con varias soluciones y no con una, tratando de evitar de esta manera quedar atrapados en óptimos locales.

La cruce es un operador que permite mantener la diversidad genética de una población y cuyo propósito es proveer un mecanismo para escapar de óptimos locales, así como desplazar a los individuos hacia zonas del espacio de búsqueda que no pueden ser alcanzadas por medio de otros operadores genéticos. Por otro lado, la recombinación consiste en la mezcla de la información de dos o más individuos mediante el entrecruzamiento de los cromosomas de los individuos a aparearse. Ambos operadores son estocásticos, i.e., se aplican con probabilidades definidas por el usuario. Normalmente se establece una probabilidad de mutación muy inferior a la probabilidad de cruce, ya que una probabilidad de mutación muy alta convertirá al proceso de búsqueda evolutivo en un proceso de búsqueda aleatorio.

Con base en el nivel de aptitud se escogen dos o más individuos, llamados padres, a los cuales se les aplican ciertos operadores (siendo los más comunes la recombinación y la mutación) para obtener nuevos individuos, denominados hijos, que mantienen algunas propiedades de sus antecesores y los cuales sobrevivirán o se eliminarán mediante un proceso de selección (probabilística o determinista). Este proceso se realiza con cada individuo de la población hasta formar la nueva población, repitiéndose el proceso un cierto número de iteraciones (generaciones).

En el algoritmo 1 se describe el pseudocódigo de un algoritmo evolutivo en donde la población actual $P_g = \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n\}$ representa el conjunto de soluciones candidatas. A partir de la población actual P_g se genera la población P'_g para que a partir de la unión de éstas se seleccionen a los individuos para generar la siguiente población P_{g+1} .

```

1  $G \leftarrow 0$ 
2 Inicializar  $P_g$ 
3 Evaluar( $P_g$ ) ( $f(\vec{x}_1), f(\vec{x}_2), \dots, f(\vec{x}_n)$ )
4 while Criterio de terminación no satisfecho do
5    $padres \leftarrow \text{seleccionarPadres}()$ 
6    $P'_g \leftarrow \text{operadorRecombinación}(padres)$ 
7    $P''_g \leftarrow \text{operadorMutación}(P'_g)$ 
8   Evaluar( $P''_g$ )
9    $P_{g+1} \leftarrow \text{seleccionar}(P_g \cup P''_g)$ 
10   $G \leftarrow G + 1$ 

```

Algoritmo 1: Esquema general de un algoritmo evolutivo

2.3.2. Principales paradigmas

Hoy en día se consideran como los principales paradigmas de la computación evolutiva a tres tipos de técnicas: *algoritmos genéticos*, *estrategias evolutivas* y *programación evolutiva*. A continuación se describirá brevemente cada uno de ellos.

Programación evolutiva

La Programación Evolutiva (PE) fue desarrollada por Lawrence J. Fogel para simular a la evolución como un proceso de aprendizaje [27]. En su forma original se usaba máquinas de estados finitos para representar a los individuos dentro una población.

En su forma original, este algoritmo opera sobre una secuencia de símbolos de un alfabeto finito cuyo objetivo es predecir el siguiente símbolo de la secuencia. No se ocupa un operador de recombinación, por lo cual las variaciones son únicamente producto de la mutación. La función de aptitud es una medida basada en la exactitud de la predicción.

En la PE no existe un proceso de selección de padres, puesto que todos los individuos de la población generan un descendiente. El mecanismo de supervivencia es simple: dada una población de μ individuos se generan μ descendientes. Entre las $\mu + \mu$ soluciones se efectúan una serie de torneos con la finalidad de seleccionar a los μ individuos que conformarán la siguiente generación.

```

1  $g \leftarrow 0$ 
2 Inicializar  $P_g$ 
3 Evaluar( $P_g$ )
4 while Criterio de terminación no satisfecho do
5    $P'_g \leftarrow \text{Mutar}(P_g)$ 
6   Evaluar( $P'_g$ )
7    $P_{g+1} \leftarrow \text{Seleccionar}(P_g \cup P'_g)$ 
8    $g \leftarrow g + 1$ 

```

Algoritmo 2: Esquema general de la programación evolutiva

Estrategias evolutivas

Las Estrategias Evolutivas (EE) fueron desarrolladas por Hans-Paul Schwefel e Ingo Rechenberg [65]. La versión original, denominada (1+1)-EE, usaba un solo padre para generar un solo hijo (EE de dos miembros); este hijo sobrevive si es mejor que el padre. Para la generación de hijos se usa una mutación que permite perturbar la solución actual usando una distribución normal con media cero y desviación estándar σ . El algoritmo 3 muestra el algoritmo básico de una estrategia evolutiva de dos miembros.

Existen estrategias evolutivas multimiembro. En ellas, a partir de μ padres se generan λ hijos. Para tales casos existen dos esquemas: si durante la selección sólo se toman en cuenta a los μ mejores individuos de los λ hijos se trata de una estrategia (μ, λ) ; por otro lado, si

```

1  $t \leftarrow 0$ 
2 Generar de manera aleatoria  $\langle x_1^t, x_2^t, \dots, x_n^t \rangle$ 
3 while Criterio de terminación no satisfecho do
4    $y_i^t = x_i^t + N(0, \sigma) \forall i \in \{1, 2, \dots, n\}$ 
5   if  $f(\vec{y}^t) \leq f(\vec{x}^t)$  then
6      $\vec{x}^{t+1} \leftarrow \vec{y}^t$ 
7   else
8      $\vec{x}^{t+1} \leftarrow \vec{x}^t$ 
9    $t \leftarrow t + 1$ 

```

Algoritmo 3: Estrategia evolutiva de dos miembros (1+1)-EE

se selecciona a los μ mejores individuos de la unión de los μ padres y los λ hijos se trata de una estrategia $(\mu + \lambda)$. En ambos casos la selección es determinista, por lo que sólo los mejores individuos pasan a la siguiente generación. A este tipo de estrategias se les conoce como estrategias evolutivas multimiembro.

Las EE simulan el proceso evolutivo a nivel de individuos, por lo que la recombinación es posible. Sin embargo, el principal operador de la EE es la mutación, jugando un papel secundario el operador de recombinación.

Algoritmos genéticos

Los Algoritmos Genéticos (AG) fueron desarrollados por John H. Holland [34]. Los Algoritmos Genéticos han sido usados ampliamente en optimización, convirtiéndose en una técnica muy popular. En su forma original, el AG codifica a los individuos en forma de cadena binaria (cromosoma). Sin embargo, en la actualidad existen una gran variedad de formas de representación, así como métodos de selección de padres y operadores de recombinación y mutación. El algoritmo 4 muestra el esquema general de un algoritmo genético.

```

1  $g \leftarrow 0$  Inicializar  $P_g$ 
2 Evaluar( $P_g$ ) ( $f(\vec{x}_1), f(\vec{x}_2), \dots, f(\vec{x}_n)$ )
3 while Criterio de terminación no satisfecho do
4   seleccionarPadres()
5    $P'_g \leftarrow \text{Cruza}(\text{padres})$ 
6    $P''_g \leftarrow \text{Mutación}(P'_g)$ 
7   Evaluar( $P''_g$ )
8    $P_{g+1} \leftarrow P''_g$ 
9    $g \leftarrow g + 1$ 

```

Algoritmo 4: Esquema general de un algoritmo genético

A diferencia de las estrategias evolutivas en los algoritmos genéticos el operador de recombinación es el principal, jugando un papel secundario el operador de mutación. La cruza

redistribuye los segmentos de información de los individuos a sus descendientes, permitiendo de esta manera que los segmentos con información útil se propaguen a lo largo de las generaciones. La mutación puede destruir esta información valiosa debido a las variaciones aleatorias que se causan.

2.4. Evolución Diferencial

La Evolución Diferencial (ED) fue propuesta como una nueva heurística para optimización de funciones no lineales y no diferenciables, la cual fue elaborada por Rainer Storn y Kenneth Price en 1995 [69]. Surgió por los intentos de Kenneth Price por resolver el polinomio de Chebychev, en el que se utilizaron vectores de la población a ser perturbados.

La ED es un método estocástico de optimización global, que usa una codificación de punto flotante para optimización global sobre espacios continuos. La ED crea soluciones nuevas mediante la combinación de un padre con otros individuos de la misma generación. Un candidato sustituye al padre sólo si tiene un valor de aptitud (*fitness*) mejor. El algoritmo cuenta con tres parámetros de control: el factor de amplificación del vector diferencia F , el parámetro de control de cruce R y el tamaño de la población N .

El algoritmo original de la ED mantiene los tres parámetros de control fijos durante todo el proceso de búsqueda. Sin embargo, existe una gran falta de conocimiento sobre cuáles deberían ser los parámetros ideales de control de la ED para una función determinada [47, 72]. La necesidad de cambiar dichos parámetros de control durante el proceso de optimización ha sido confirmado empíricamente [8].

2.4.1. Algoritmo de la ED

La Evolución Diferencial es un algoritmo de búsqueda directa que utiliza N vectores de dimensionalidad D . La versión original de la ED puede definirse con base en los siguientes conceptos:

1. La población:

$$P_{x,g} = (\vec{x}_{i,g}) \quad i = 0, 1, \dots, N-1, \quad g = 0, 1, \dots, G_{max}$$

$$\vec{x}_{i,g} = [x_0, x_1, \dots, x_{D-1}]^T$$

en donde N denota el número máximo de vectores que conforman la población, g es el contador de generaciones, G_{max} es el número máximo de generaciones y D es la dimensionalidad, i.e., el número de variables de decisión del problema.

2. Inicialización de la población :

$$\vec{x}_{i,j} = x_{j,L} + U(0,1) \cdot (x_{j,U} - x_{j,L}) \quad \forall i \in \{0, 1, \dots, N-1\}, \quad \forall j \in \{0, 1, \dots, D-1\}$$

en donde $x_{j,L}$ y $x_{j,U}$ denotan los límites inferior y superior del j -ésimo componente, respectivamente. $U(0,1)$ retorna un número aleatorio distribuido uniformemente en el

rango $[0, 1]$. Es importante mencionar que los individuos iniciales deben estar distribuidos aleatoriamente de alguna manera ya que las diferencias de los vectores objetivo conducen el comportamiento de la ED.

Después de la inicialización, la población es sujeta a un proceso iterativo de **mutación**, **recombinación** y **selección** durante G_{max} iteraciones (generaciones). Para tales efectos, la ED emplea operadores de mutación y recombinación para generar un solo descendiente o *vector candidato* \vec{u}_i por cada vector padre \vec{x}_i . A continuación se describen dichos operadores.

2.4.2. Componentes de la ED

La idea fundamental detrás del algoritmo de la ED es un mecanismo para generar los vectores de prueba. La cruce y mutación se utilizan para generar nuevos vectores (vectores de prueba) mientras que la selección elige cuál de estos vectores sobrevivirán a la próxima generación.

Mutación

La mutación se refiere al proceso de generar variaciones aleatoriamente a una o más variables de decisión de un vector solución dado. El objetivo de este esquema es generar movimientos que desplacen a los vectores solución en la dirección y magnitud correcta. El éxito de esta tarea depende directamente de la función de distribución empleada para la generación de dichas perturbaciones.

En su forma más simple, la ED genera un vector de mutación sumando una diferencia ponderada de un par de vectores aleatoriamente seleccionados de la población. Mediante este operador, para cada vector objetivo $\vec{x}_{i,G}$, se genera un vector de mutación de acuerdo a:

$$\vec{v}_{i,G+1} = \vec{x}_{r_1,G} + F (\vec{x}_{r_2,G} - \vec{x}_{r_3,G}) , r_1 \neq r_2 \neq r_3 \neq i \quad (2.1)$$

en donde r_1, r_2 y $r_3 \in [1, N]$ son seleccionados aleatoriamente. F es un parámetro real en el intervalo $[0, 2]$ el cual controla la amplificación del vector diferencia $\vec{x}_{r_2,G} - \vec{x}_{r_3,G}$. El vector base $\vec{x}_{r_1,G}$ puede ser determinado de diferentes formas las cuales se mencionarán más adelante.

Recombinación

La mutación es el operador encargado de generar nuevas direcciones de búsqueda como se mencionó anteriormente. La recombinación, por su lado, es un proceso complementario que permite el intercambio de información de los vectores a sus descendientes.

En sus primeras versiones, la ED emplea un procedimiento de recombinación discreto y no uniforme. El vector de dirección $\vec{x}_{i,G}$ se mezcla con el vector mutado $\vec{v}_{i,G+1}$, utilizando el siguiente esquema, para de esta manera obtener el vector de prueba $\vec{u}_{i,G+1}$

$$\vec{u}_{i,G+1} = [u_{1i,G+1}, u_{2i,G+1}, \dots, u_{Di,G+1}]^T$$

donde:

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1} & \text{si } U(0,1) \leq R \text{ o } j = j_{rand} \\ x_{ji,G} & \text{en otro caso} \end{cases} \quad (2.2)$$

donde $R \in [0, 1]$ es el parámetro de control de cruza.

Se puede observar que cuando $U(0,1) \leq R$ o $j = j_{rand}$ el valor de la j -ésima variable de decisión es una combinación lineal de los vectores aleatoriamente seleccionados (\vec{x}_{r_1} , \vec{x}_{r_2} y \vec{x}_{r_3}); de otra manera, este valor es heredado directamente del padre. La condición $j = j_{rand}$ se incluye con el fin de asegurar que \vec{u}_i difiera al menos en un parámetro de \vec{x}_i .

Selección

La ED usa un mecanismo de selección codicioso, el cual está definido por:

$$\vec{x}_{i,G+1} = \begin{cases} \vec{u}_{i,G+1} & \text{si } f(\vec{u}_{i,G+1}) < f(\vec{x}_{i,G}) \\ \vec{x}_{i,G} & \text{en otro caso} \end{cases} \quad (2.3)$$

Después de que cada vector \vec{u}_i es evaluado en la función objetivo, los valores de $f(\vec{u}_i)$ y $f(\vec{x}_{i,G})$ son comparados, eligiéndose a $\vec{x}_{i,G+1}$ como el vector que posea un mejor valor de aptitud dependiendo del problema (minimización o maximización).

Los procesos de recombinación y selección se ilustran en la figura 2.3 mientras que en la figura 2.4 se muestra un ejemplo hipotético en dos dimensiones aplicando una cruza uniforme. Una vez determinados los individuos que conforman la siguiente generación, el ciclo se repite hasta que el problema sea resuelto o todos los vectores objetivo converjan a un punto o no se logre ninguna mejora después de varias iteraciones. El algoritmo 5 muestra a la ED en su forma original.

```

1   $G \leftarrow 0$ 
2  Inicializar  $P_{x,G}$ 
3  while Criterio de terminación no satisfecho do
4      for  $i \leftarrow \{0, \dots, N-1\}$  do
5          Seleccionar  $r_1, r_2, r_3 \in \{0, \dots, N-1\}$  aleatoriamente, donde  $r_1 \neq r_2 \neq r_3$ 
6          Seleccionar  $j_{rand} \in \{0, \dots, D-1\}$  aleatoriamente
7          for  $j \leftarrow \{1, \dots, D-1\}$  do
8              if  $U[0,1] < R$  o  $j = j_{rand}$  then
9                   $u_{i,j} \leftarrow x_{r_3,j,G} + F(x_{r_1,j,G} - x_{r_2,j,G})$ 
10             else
11                  $u_{i,j} \leftarrow x_{i,j,G}$ 
12             if  $f(\vec{u}_i) \leq f(\vec{x}_{i,G})$  then
13                  $\vec{x}_{i,G+1} \leftarrow \vec{u}_i$ 
14   $G \leftarrow G + 1$ 

```

Algoritmo 5: Algoritmo de Evolución Diferencial ED/rand/1/bin

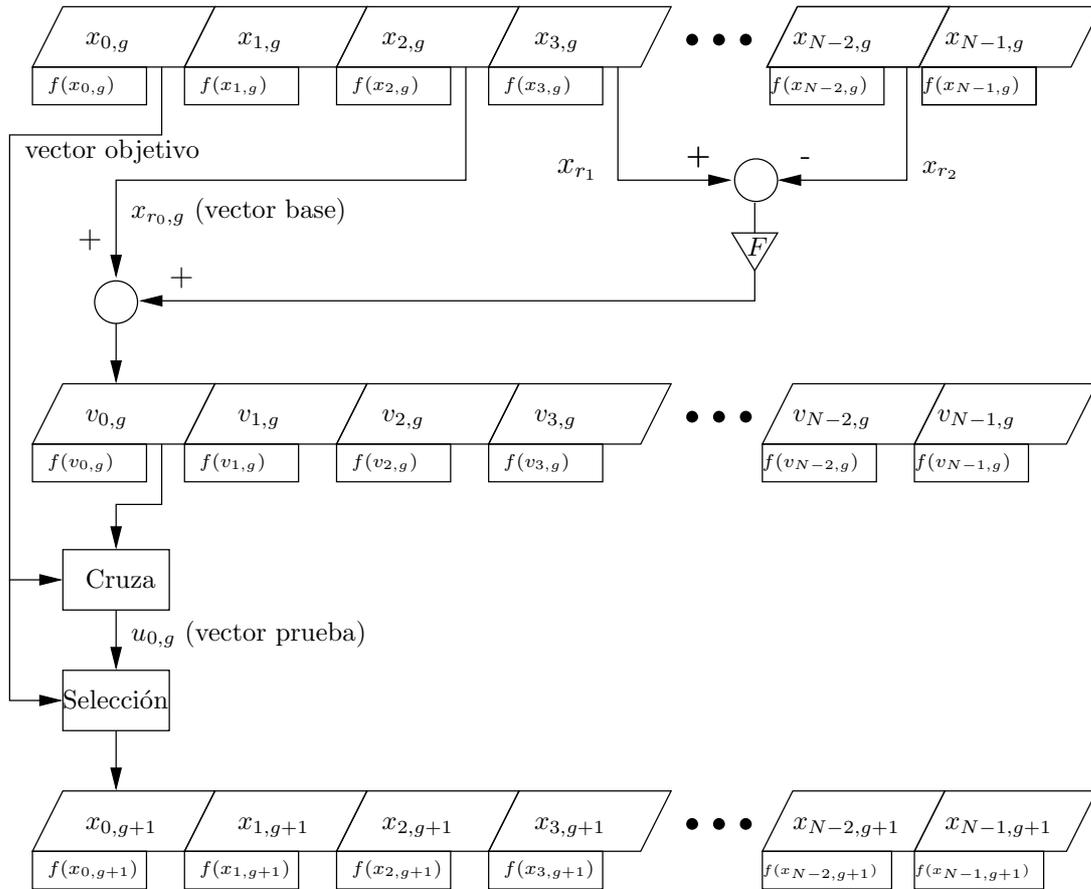


Figura 2.3: Diagrama de flujo de la Evolución Diferencial

2.4.3. Variantes de la ED

En la literatura se han propuesto diversas estrategias para los operadores de mutación y recombinación de la ED. Por tal motivo, se estableció una nomenclatura que denomina las diferentes variantes existentes [58]. Por ejemplo, para la versión original de la ED su nomenclatura correspondiente es **ED/rand/1/bin**, en donde *ED* establece que se trata de una versión del algoritmo de la ED, *rand* indica el tipo de selección (en este caso en particular se realiza una selección aleatoria en los vectores que conforman el vector de mutación), el valor *1* señala el número de pares de vectores que conforman la diferencia y finalmente *bin* establece el proceso de recombinación binomial.

Inicialmente se propusieron dos procedimientos de recombinación denominados: exponencial (*exp*) y binomial (*bin*). Estos controlan la frecuencia con la cual los parámetros del vector mutación \vec{v} son seleccionados para formar parte de la solución descendiente \vec{u} . En ambas la recombinación se realiza en cada una de las D variables de decisión. Para la recombinación binomial, cuando al elegir un número aleatorio entre $(0, 1)$ éste sea menor o igual que el valor R la variable se toma del vector mutación \vec{v} , en caso contrario se toma del padre original \vec{x} . Para la recombinación exponencial, las variables se toman del vector mutación \vec{v} hasta que la primera ocurrencia en que el número aleatorio entre $(0, 1)$ supera

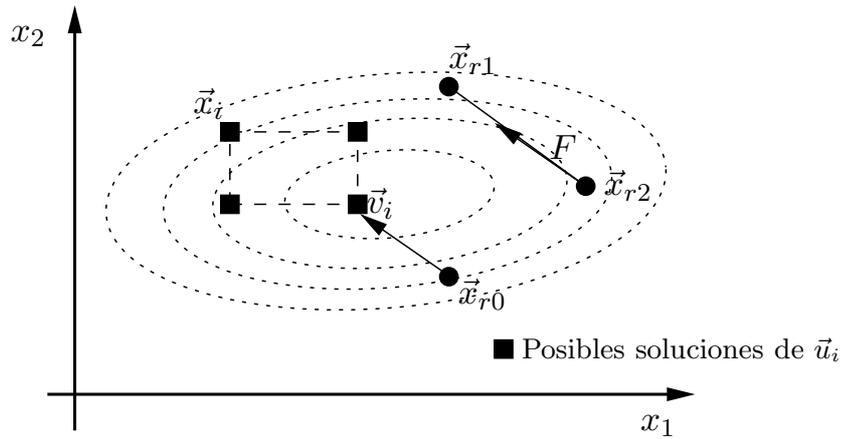


Figura 2.4: Ejemplo de los vectores de prueba de la Evolución Diferencial

el valor R , en este momento las variables restantes se toman del padre original \vec{x} . La cruce exponencial es similar a la cruce de un punto [29], mientras que la cruce binomial es similar a la cruce uniforme en los algoritmos genéticos [71].

En la figura 2.5 se ilustran ambos procesos, donde $U(0, 1)$ representa una variable aleatoria distribuida en el intervalo $[0, 1]$ y el vector \vec{u} es el resultado del proceso de recombinación de los vectores \vec{v} y \vec{x} . Para el ejemplo mostrado en la figura 2.5, en el caso de la cruce binomial las variables 1, 3 y 4 fueron tomadas del vector \vec{v} (cuando $U(0, 1) \leq R$) mientras que las variables 2 y 5 fueron tomadas del vector \vec{x} (cuando $U(0, 1) > R$). Para el caso de la cruce exponencial, la primer ocurrencia en que $U(0, 1) > R$ fué en la variable 4, por esta razón las primeras tres variables son tomadas del vector \vec{v} mientras que las restantes son tomadas del vector \vec{x} .

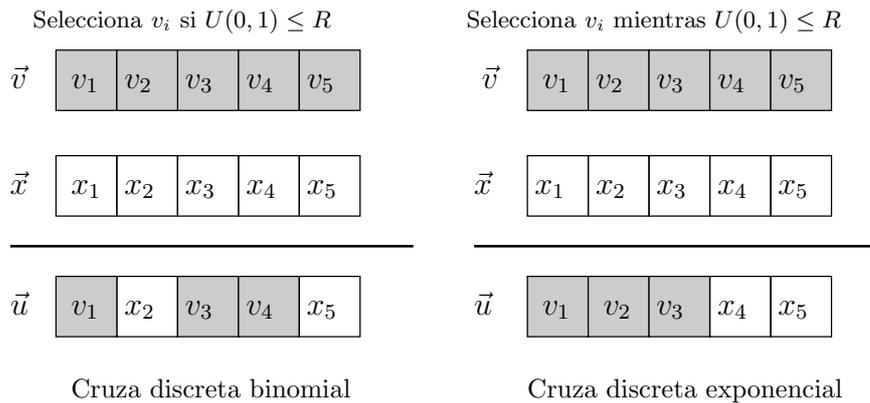


Figura 2.5: Recombinación discreta binomial y exponencial de la Evolución Diferencial

Un ejemplo más lo constituiría un algoritmo de ED que seleccione aleatoriamente a cuatro vectores que componen al vector mutación y que son recombinados con un proceso exponencial perturbando a la mejor solución hasta el momento. Este esquema se denomina

ED/best/2/exp.

La recombinación discreta es un proceso variante a la rotación. Por tal motivo, se han propuesto otros esquemas de recombinación invariantes a la rotación; el más empleado es la recombinación aritmética. La forma del vector descendiente \vec{u} se muestra en la ecuación (2.4), donde K es un factor que representa el sesgo de la cruce; cuando $k \rightarrow 0$ el vector $\vec{u} \rightarrow \vec{x}$. Por otro lado, cuando $K \rightarrow 1$ el vector $\vec{u} \rightarrow \vec{v}$. Este esquema se denomina **ED/current-to-rand/1**, donde *current-to-rand* indica una combinación lineal de los vectores \vec{x} y \vec{v} .

$$\vec{u}_i = \vec{x}_i + K * (\vec{x}_{r_3} - \vec{x}_i) + F * (x_{r_1} - x_{r_2}) \quad (2.4)$$

Las diversas estrategias que pueden adoptarse en el algoritmo de la ED dependen del tipo de problema al que se aplique. Como se mencionó anteriormente, las estrategias se basan en el vector a perturbar, el número de vectores considerados para la perturbación y, finalmente, el tipo de cruce a usar. En la tabla 2.1 se listan los principales modelos de la Evolución Diferencial propuestos en la literatura [58].

Nomenclatura	Modelo
rand/p/bin	$u_{i,j} = \begin{cases} x_{r_3,j} + F * \sum_{k=1}^p (x_{r_1,j}^p - x_{r_2,j}^p) & \text{si } U(0,1) < R \text{ o } j = j_{rand} \\ x_{i,j} & \text{en caso contrario} \end{cases}$
rand/p/exp	$u_{i,j} = \begin{cases} x_{r_3,j} + F * \sum_{k=1}^p (x_{r_1,j}^p - x_{r_2,j}^p) & \text{mientras } U(0,1) < R \text{ o } j = j_{rand} \\ x_{i,j} & \text{en caso contrario} \end{cases}$
best/p/bin	$u_{i,j} = \begin{cases} x_{r_{best},j} + F * \sum_{k=1}^p (x_{r_1,j}^p - x_{r_2,j}^p) & \text{si } U(0,1) < R \text{ o } j = j_{rand} \\ x_{i,j} & \text{en caso contrario} \end{cases}$
best/p/exp	$u_{i,j} = \begin{cases} x_{r_{best},j} + F * \sum_{k=1}^p (x_{r_1,j}^p - x_{r_2,j}^p) & \text{mientras } U(0,1) < R \text{ o } j = j_{rand} \\ x_{i,j} & \text{en caso contrario} \end{cases}$
current-to-rand/p	$u_{i,j} = \begin{cases} x_{i,j} + F * (x_{r_3,j} - x_{i,j}) + F * \sum_{k=1}^p (x_{r_1,j}^p - x_{r_2,j}^p) & \text{si } U(0,1) < R \text{ o } j = j_{rand} \\ x_{i,j} & \text{en caso contrario} \end{cases}$
current-to-best/p	$u_{i,j} = \begin{cases} x_{i,j} + F * (x_{r_{best},j} - x_{i,j}) + F * \sum_{k=1}^p (x_{r_1,j}^p - x_{r_2,j}^p) & \text{si } U(0,1) < CR \text{ o } j = j_{rand} \\ x_{i,j} & \text{en caso contrario} \end{cases}$

Tabla 2.1: Principales modelos de la Evolución Diferencial

La estrategia a adoptar para cada problema se suele determinar de manera empírica, aunque existen ciertos lineamientos generales al respecto en la literatura especializada [70]. Evidentemente, una estrategia que funcione bien para una cierta clase de problemas puede no ser la más adecuada cuando se aplique a una clase distinta de problemas.

Hiperheurísticas

En este capítulo se profundizará en torno al origen e inspiración del concepto de hiperheurística y se proporcionará un marco general de este tipo de técnica, además se hará revisión de diversos enfoques recientes que existen para el mecanismo de selección de las heurísticas de bajo nivel.

3.1. Introducción

Para muchos de los problemas del mundo real, una búsqueda exhaustiva de todas las posibles soluciones no es una alternativa viable para resolverlos, debido a que el espacio de búsqueda podría ser demasiado grande y/o accidentado. Es común entonces utilizar algún tipo de enfoque heurístico, sacrificando la garantía de encontrar la solución óptima pero reduciendo el tiempo computacional promedio de manera significativa. A lo largo de los años se ha desarrollado una enorme cantidad de algoritmos heurísticos, muchos de los cuales son específicos a ciertas clases de problemas de optimización.

El término *heurística* se utiliza para referirse a algoritmos de búsqueda general. Durante varios años, diversos autores intentaron argumentar la superioridad de algunas heurísticas con respecto a otras en todas las clases de problemas. Esta práctica comenzó a desaparecer cuando, Wolpert y MacReady publicaron su teorema del “*No Free Lunch*” [75], que demostró que, cuando se promedian sobre todos los problemas definidos en un determinado espacio de búsqueda finito, todos los algoritmos de búsqueda heurística presentan el mismo rendimiento. En otras palabras, no es posible diseñar una heurística que sea mejor que las demás en todas las clases de funciones.

Sin embargo, mucho antes de que el teorema se publicara, estaba claro que el uso de heurísticas individuales, por muy valiosas que fueran, podrían presentar interesantes peculiaridades y/o limitaciones. Por consiguiente, surgió la idea de poder combinar de alguna manera varias heurísticas de tal manera que unas pudieran compensar las debilidades de otras. Las **hiperheurísticas** han surgido como parte de este nuevo enfoque. El término hiperheurística fue introducido por Cowling et al. en [16], en donde las define de la manera siguiente:

Definición 3 *Enfoques que funcionan a un nivel más alto de abstracción que las heurísticas y gestionan la elección de lo que debería ser un método heurístico de bajo nivel para ser*

aplicado en un momento dado, dependiendo de las características de la región del espacio de soluciones en exploración.

A partir de esta definición, un algoritmo hiperheurístico básico podría ser el siguiente:

```

/* Conjunto de heurísticas */
1  $H = \{h_1, h_2, \dots, h_n\}$ 
2  $i \leftarrow 0$ 
3 while Criterio de terminación no satisfecho do
4   Seleccionar la heurística  $h_j$  más adecuada, donde  $j \in \{1, 2, \dots, |H|\}$ 
5   Transformar el estado actual  $S_i$  al estado  $S_{i+1}$  aplicando  $h_j$ 
6    $i \leftarrow i + 1$ 

```

Algoritmo 6: Pseudocódigo de un algoritmo hiperheurístico básico

La idea básica que reside detrás de las hiperheurísticas consiste en utilizar un conjunto de métodos heurísticos que actúan sobre un problema determinado (*heurísticas de bajo nivel*), de tal forma que en cada momento la hiperheurística decida cuál es el mejor de ellos para resolver el problema. En otras palabras, las hiperheurísticas operan en el **espacio de las heurísticas** mientras que las heurísticas trabajan directamente sobre el **espacio de las soluciones** del problema. Estas pueden ser descritas como estrategias diseñadas para controlar la aplicación de un conjunto de heurísticas durante el proceso de búsqueda y suelen verse como cajas negras (ver fig. 3.1).

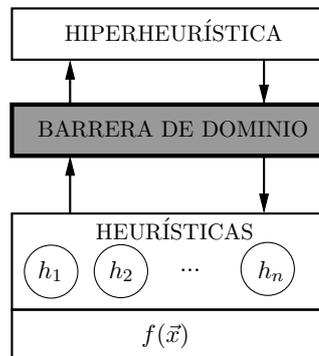


Figura 3.1: Marco de una hiperheurística

Esto significa que la hiperheurística no es la encargada de buscar la mejor solución al problema. Lo que hace, es seleccionar en cada paso del proceso de búsqueda las heurísticas de bajo nivel más prometedoras que sean potencialmente capaces de mejorar la solución actual. Por otra parte, si no hay mejora, la hiperheurística debe ser capaz de diversificar la búsqueda a otras regiones mediante la selección de otras heurísticas, buscando de esta manera poder escapar de óptimos locales.

Comúnmente suelen confundirse los términos metaheurística e hiperheurística; sin embargo la diferencia entre ellos es importante. Las primeras trabajan sobre problemas en

particular además de que incorporan información adicional. Su éxito se debe a este hecho, ya que sus implementaciones utilizan conocimiento y propiedades del problema mismo. Otro inconveniente que presentan las metaheurísticas es el ajuste de sus parámetros, a los cuales suelen ser muy sensibles. En virtud de lo anterior, las metaheurísticas deben ser desarrolladas y usadas por expertos los cuales tengan suficiente conocimiento y experiencia tanto del dominio del problema como de la metaheurística empleada.

Por su lado, las hiperheurísticas trabajan a un nivel más alto de abstracción y comúnmente no incluyen información alguna sobre el dominio del problema, por lo que son desarrollados como métodos de optimización general y de esta manera pueden aplicarse a una gama más amplia de problemas. Idealmente, una hiperheurística sólo requiere como parámetros el conocer el número de heurísticas de bajo nivel así como el tipo de función objetivo a optimizar (mín $f(\vec{x})$ o máx $f(\vec{x})$) [12].

El propósito de desarrollar hiperheurísticas viene del hecho de poder proporcionar un marco general que ofrezca soluciones de buena calidad para un mayor número de problemas de optimización. Otra motivación viene del hecho de que el desempeño de varias heurísticas puede variar dependiendo de las características específicas del problema. De hecho, el uso de ciertas heurísticas puede ir variando dependiendo de la etapa actual en el proceso de búsqueda. Las diferentes heurísticas a usar se pueden coordinar de tal forma que combinen sus mecanismos de exploración (diversificación) con los de explotación de buenas regiones del espacio de búsqueda (intensificación).

3.2. Marco de una hiperheurística

Como se ha señalado anteriormente, las hiperheurísticas trabajan sobre un conjunto de heurísticas de bajo nivel. La motivación detrás de este enfoque sugiere que un algoritmo hiperheurístico que haya sido desarrollado para un cierto dominio de un problema particular pueda extenderse fácilmente a otros dominios simplemente reemplazando el conjunto de heurísticas de bajo nivel y la función de evaluación.

En la figura 3.1 se mostró el marco general de una hiperheurística. En dicho marco se muestra una barrera entre las heurísticas de bajo nivel y la hiperheurística que tiene como objetivo que el conocimiento del problema no debe cruzar esta barrera, impidiendo de esta manera brindar a la hiperheurística conocimiento del dominio bajo el cual está trabajando. La hiperheurística cuenta con n heurísticas a las que puede recurrir para ir transformando el estado actual en el estado siguiente.

Existe por supuesto una interfaz bien definida entre la hiperheurística y las heurísticas de bajo nivel. Dicha interfaz debe considerar los siguientes aspectos:

1. La interfaz debe ser estándar, esto es, sólo debe requerirse de una interfaz para comunicar la hiperheurística con el conjunto de heurísticas. De lo contrario se requerirá de una interfaz independiente para cada heurística, con la finalidad de facilitar el paso de los datos (p. ej., el valor de la función objetivo y el tiempo requerido por la heurística) entre las heurísticas de bajo nivel y la hiperheurística (y viceversa). La interfaz

también tiene la obligación de definir si el resultado arrojado por la heurística de bajo nivel debe aplicarse al estado actual del proceso de búsqueda o sólo informar el efecto que tendría si se aplicara dicho cambio, permitiendo de esta manera decidir por parte de la hiperheurística qué heurística o conjunto de heurísticas están autorizadas a modificar la solución actual.

2. La interfaz debe facilitar el desarrollo hacia otros dominios. Cuando se requiera dar solución a un nuevo problema, el usuario sólo tiene que suministrar el conjunto de heurísticas de bajo nivel así como la función de evaluación. Si las heurísticas de bajo nivel siguen la interfaz estándar de la hiperheurística, ésta no debe ser alterada de ninguna manera, por lo que debe ser capaz de resolver el problema tan pronto la información haya sido introducida. Es decir, como se ha señalado anteriormente, se trata de contar con un mecanismo que funcione a un nivel más alto de abstracción que las metaheurísticas.

3.3. Enfoques modernos en las hiperheurísticas

Actualmente existen diversos mecanismos de selección diseñados para las hiperheurísticas, los cuales, según Chakhlevitch y Cowling [12], se pueden clasificar en cuatro grandes grupos:

1. Hiperheurísticas basadas en selección aleatoria
2. Hiperheurísticas codiciosas
3. Hiperheurísticas basadas en metaheurísticas
4. Hiperheurísticas con aprendizaje

En las siguientes secciones se dará una descripción breve de cada una de ellas.

3.3.1. Hiperheurísticas basadas en selección aleatoria

Las hiperheurísticas basadas en selección aleatoria han sido el enfoque de selección más antiguo ya que éste es el mecanismo más simple y fácil de aplicar. En este mecanismo de selección se elige al azar una heurística de bajo nivel y siempre se le aplica, incluso si no produce ninguna mejora a la solución actual del problema.

La principal desventaja de un enfoque puramente aleatorio es que la calidad de la solución obtenida depende de la posibilidad de seleccionar una “buena” secuencia de heurísticas de bajo nivel. Con el fin de evitar quedar atrapados en regiones pobres del espacio de búsqueda es necesario realizar modificaciones a este enfoque. Estas modificaciones consisten en afectar la regla de no aceptar siempre todos los movimientos malos. En [16] y [17], Cowling et al. comparan diferentes versiones del mecanismo de selección aleatorio aplicado a un problema práctico de programación de reuniones. Dichos mecanismos se mencionan a continuación:

- *Aleatorio simple.*- Escoge aleatoriamente la heurística a aplicar en cada iteración.

- *Aleatorio descendente*.- Selecciona una heurística aleatoriamente y la aplica repetidamente conforme el resultado se va mejorando y se detiene hasta que ningún otro movimiento produzca una mejora.
- *Permutación aleatoria*.- Crea una permutación inicial de las heurísticas y en cada iteración aplica la heurística por cada una de las posiciones de la permutación.
- *Permutación aleatoria descendente*.- Es idéntico al anterior sólo que la heurística se aplica hasta que el resultado de aplicar una cierta heurística deje de mejorar.

Los resultados de los experimentos llevados a cabo en [12] indican que la hiperheurística con el mecanismo *aleatorio descendente* se desempeña mejor que la versión simple. En ese estudio se menciona que la única posibilidad de poder quedar atrapados en un óptimo local ocurre en las primeras etapas de la búsqueda cuando ningún otro método heurístico de bajo nivel es capaz de mejorar dicha solución. Esta situación resulta ser más probable cuando el número de heurísticas de bajo nivel es pequeño. Por ello proponen introducir un nivel de aceptación con el cual una heurística que no produzca mejora sólo será aceptada con cierto valor de probabilidad.

Recientemente, han surgido en la literatura un conjunto de hiperheurísticas basadas en métodos de Monte Carlo. En general, estos enfoques usan una probabilidad para la aceptación de una nueva solución que no mejora, la cual va disminuyendo cuando la diferencia δ en los valores de las funciones objetivo de la nueva solución y la mejor solución se incrementan (considerando minimización). Esta probabilidad puede ser calculada de diferentes maneras. Ayob y Kendall [3] aplican una hiperheurística de Monte Carlo para la optimización de componentes electrónicos en un tablero de circuito impreso. Cuentan con seis heurísticas de bajo nivel las cuales representan los movimientos de intercambio de componentes y, consideran funciones lineales y exponenciales de δ para definir las probabilidades de aceptación en caso de que el movimiento generado por la heurística de bajo nivel no sea bueno. En su trabajo concluyen que usando la función exponencial obtienen los mejores resultados, además de que incluyen el tiempo que tarda la heurística para el cálculo de las probabilidades.

Kendall y Mohamad en [40] y [41] introdujeron otro hiper-método heurístico que también se centra en el criterio de aceptación incorporado al método de selección aleatorio. Usaron el algoritmo Gran Diluvio (*Great Deluge*) [22]. Es un algoritmo muy similar al método de descenso empinado (*hill-climbing*) en el cual se simula a una persona estando bajo un gran diluvio y la cual tenderá a moverse a direcciones más altas con la esperanza de encontrar un lugar en donde el nivel del agua no la alcance. La hiperheurística comienza con un nivel inicial. En cada paso, el movimiento que produzca un valor de aptitud menor que el del nivel actual es aceptado. Es decir, se selecciona una heurística a ser aplicada y, si el valor retornado por ésta es mejor que el valor de umbral, es aceptada. En cada paso, este nivel es disminuido de acuerdo a un factor (considerando minimización). Otro criterio de aceptación es que el movimiento es aceptado siempre y cuando el valor retornado por la heurística esté razonablemente cercano a la mejor solución actual.

En general, las heurísticas aleatorias son sencillas de implementar y pueden aplicarse a

cualquier problema de optimización y los resultados alcanzados por éstas pueden ser usados como puntos de referencia para evaluar el rendimiento de otros enfoques hiperheurísticos.

3.3.2. Hiperheurísticas codiciosas

Las hiperheurísticas codiciosas seleccionan y aplican en cada iteración la heurística de bajo nivel que produce la mayor mejora en el valor de la función objetivo, o bien, aquella que produzca el menor deterioro si no se mejora la solución previa. Existen dos estrategias codiciosas que pueden usarse: una acepta sólo mejoras de la heurística de bajo nivel mientras que la segunda acepta tanto movimientos que mejoran como aquellos que no mejoren a la solución actual. La segunda versión es mejor ya que impide a la hiperheurística detenerse pronto quedando atorada en un óptimo local. Además, cabe señalar que cualquier hiperheurística codiciosa va a requerir de una evaluación de cada heurística de bajo nivel con el fin de seleccionar la mejor, lo cual hace que este mecanismo sea mucho más lento que una hiperheurística basada en selección aleatoria.

La principal desventaja del mecanismo codicioso es su limitada capacidad para explorar con eficacia el espacio de búsqueda dejando muchas regiones con soluciones potencialmente fuertes no visitadas. Para superar los problemas asociados con óptimos locales, Cowling y Chakhlevitch en [14] y [15] diseñaron un esquema que combina mecanismos codiciosos y aleatorios para la selección de las heurísticas de bajo nivel. En este enfoque, un mecanismo aleatorio se encarga de escoger aleatoriamente una heurística de un conjunto de las mejores obtenidas mediante un proceso codicioso. La longitud de la lista de heurísticas candidatas puede ser ajustada con el fin de lograr una buena relación entre la intensificación y la diversificación en la búsqueda. Los autores consideran cuatro versiones de hiperheurística codiciosa utilizando tanto listas estáticas como dinámicas para un problema de programación de horarios.

En general, las hiperheurísticas codiciosas pueden ser aplicadas fácilmente a problemas de optimización, debido a su simplicidad y su alto nivel de generalidad. Sin embargo, su baja velocidad las hace desfavorables para problemas donde encontrar una buena solución es un factor importante.

3.3.3. Hiperheurísticas basadas en meta-heurísticas

Una metaheurística es un algoritmo de búsqueda para dar una solución a problemas complejos. Tomando en cuenta su aplicación exitosa para resolver problemas de optimización difíciles del mundo real, surge la cuestión si estos algoritmos son igual de eficientes al trabajar sobre un espacio de búsqueda de heurísticas de bajo nivel, lo cual resulta un tema de investigación interesante. Varios enfoques metaheurísticos y sus híbridos han sido probados como mecanismos de selección de las hiperheurísticas en los últimos años y se les suele denominar *hiperheurísticas basadas en metaheurísticas*.

En [68] Storer et al. desarrollaron una hiperheurística basándose en tres metaheurísticas populares: recocido simulado, búsqueda tabú y un algoritmo genético. En su enfoque definen dos espacios, el primero de ellos se refiere al espacio de búsqueda del problema mismo mientras que el segundo espacio se refiere al espacio de las heurísticas. Probaron su enfoque en

varios problemas de programación de horarios aplicando las tres metaheurísticas al espacio de búsqueda de las heurísticas. La principal aportación de este trabajo fue la introducción del concepto de *espacio heurístico*, que consiste en el espacio de búsqueda creado por el vecindario formado por cada una de las heurísticas de bajo nivel, el cual resulta ser la base para cualquier hiperheurística basada en el uso de metaheurísticas.

Hiperheurísticas basadas en algoritmos genéticos

Los primeros intentos de desarrollo de un método de búsqueda de buenas soluciones están relacionados con los Algoritmos Genéticos (AG) cuyas soluciones son codificadas en estructuras denominadas *cromosomas*. Dicha codificación por lo general se hace a través de cadenas binarias, permutaciones, vectores de números reales, etc. Sin embargo, las soluciones a muchos problemas del mundo real usan una estructura muy compleja lo que hace extremadamente difícil la codificación de dichas soluciones. Otras desventajas de una codificación son la gran longitud que puede llegar a tomar el cromosoma para representar dichas soluciones y la necesidad de incorporar operadores específicos para operar sobre dichas soluciones.

Las hiperheurísticas basadas en AG han servido como punto de partida para crear nuevas hiperheurísticas donde su mecanismo de selección está inspirado en otras metaheurísticas. Las primeras ideas en codificar indirectamente el método heurístico a aplicar a un AG fue propuesto por Norenkoy [54] y Fang et al. [26].

En [16], Cowling, et al. diseñaron un enfoque hiperheurístico basado en un AG, llamado *hyper-GA* probándolo en un problema del mundo real. El cromosoma de este AG representa una secuencia de números enteros que corresponde a cada una de las heurísticas de bajo nivel, por lo que la longitud del cromosoma será del mismo tamaño que el número de heurísticas de bajo nivel. En este enfoque, el AG opera a un nivel superior y se encarga de generar secuencias de las heurísticas de bajo nivel para, de esta manera, poder ser aplicadas en el orden que aparecen en el cromosoma. Cada individuo del AG codifica una permutación de posiciones, las cuales representan las posibles secuencias utilizando un operador de cruce de un punto y un operador de mutación que se encarga de intercambiar aleatoriamente dos heurísticas de bajo nivel. Esta hiperheurística produce soluciones significativamente mejores que las heurísticas individuales tanto en calidad de las soluciones como en tiempo de CPU requerido.

Hiperheurísticas basadas en recocido simulado y búsqueda tabú

El recocido simulado [1] y la búsqueda tabú [28] son otros dos enfoques que pueden ser utilizados para controlar la búsqueda en el espacio de las heurísticas, decidiendo en cada iteración qué algoritmo de bajo nivel debe ser aceptado o rechazado para poder modificar la solución actual del problema, dependiendo del valor de la nueva solución una vez que ésta haya sido aplicada.

Bai y Kendall [4] desarrollaron una hiperheurística basada en recocido simulado para resolver un problema de asignación de espacios en estanterías en el cual combinan 12 heurísticas de bajo nivel reportando resultados de alta calidad. Chakhlevitch [11] propuso otro

enfoque basado en un recocido simulado para un problema de programación de ventas, en el cual demuestra que su hiperheurística produce resultados más consistentes además de mostrar que este enfoque es menos sensible a la elección de la solución inicial.

Las hiperheurísticas basadas en búsqueda tabú han cobrado cierto interés en los últimos años. Kendall y Mohd Hussin [42] consideran una hiperheurística basada en una búsqueda tabú simple para resolver un problema de asignación de horarios. Su hiperheurística gestiona 13 heurísticas de bajo nivel, bajo un esquema en el que una heurística se convierte en tabú, tan pronto como se aplica a la solución actual, independientemente de si mejora la solución o no, por lo que dicho método no podrá volver a ser aplicado mientras se encuentre en la lista tabú, incluso si en una ejecución posterior conduce a una mejor solución. En [43] Kendall y Hussin Mohd consideran dos versiones más avanzadas de su enfoque original. En la primera versión utilizan un mecanismo descendente, v.g., aplican repetidamente la heurística mientras mejore la solución previa y se convierte en tabú hasta que deja de producir mejores soluciones. En la segunda versión aceptan una heurística de bajo nivel que no mejore la solución sólo si ésta se encuentra dentro de un límite determinado.

Adicionalmente, Burke et al. [9,10] propusieron otra hiperheurística con selección mediante búsqueda tabú. Esta técnica jerarquiza a cada una de las heurísticas usadas. Al inicio de la ejecución cada heurística comienza con el menor rango ($r_k = r_{min}$ donde $k = 1, 2, \dots, n$ y n es el número de heurísticas). Conforme cada heurística produce un mejor movimiento, su rango se incrementa sin exceder éste un valor máximo r_{max} ($r_k = r_k + \alpha$). Cada vez que una heurística no produce un mejor movimiento su rango se decrementa y, de manera análoga al caso anterior, este valor no puede ser menor que el valor mínimo definido ($r_k = r_k - \alpha$). Adicionalmente, ésta se adiciona a la lista tabú. También se requiere de un parámetro que indique el tiempo de duración máxima que una heurística se encuentre dentro de la lista tabú, la cual consiste del número de iteraciones máximas que una heurística se encuentre en ella, para que una vez excedido este tiempo, la heurística sea borrada de dicha lista.

Cowling y Chakhlevitch en [14,15] presentan diferentes versiones de una hiperheurística basada en búsqueda tabú. Esta hiperheurística trabaja sobre un gran conjunto de heurísticas de bajo nivel (casi 100). Si la heurística conduce a un valor de función objetivo mejor, siempre es aceptada; en caso contrario, se le coloca en la lista tabú. Los autores probaron varias versiones, dejando fija la lista tabú y cambiándola dinámicamente. En su trabajo reportan que la hiperheurística basada en una lista tabú logró conseguir mejores resultados a los obtenidos por otros métodos hiperheurísticos basados en una selección aleatoria y codiciosa.

En general, las hiperheurísticas basadas en metaheurísticas han demostrado ser muy eficaces en la solución de problemas del mundo real, incluso superando a métodos de última generación adaptados a cada problema en particular. Sin embargo, al igual que los enfoques tradicionales de las metaheurísticas, se requiere de un ajuste fino en sus parámetros (temperatura en el recocido simulado, tamaño de la lista tabú, probabilidades de cruce y mutación, etc). Aunque las hiperheurísticas basadas en estos enfoques han demostrado no ser muy sensibles a dichos parámetros no hay garantía de que una hiperheurística funcionará de igual manera para diferentes tipos de problemas con la misma configuración en sus parámetros. Recordemos que uno de los principales objetivos de una hiperheurística

es proporcionar un marco general para la rápida solución a dominios diferentes, v.g., una hiperheurística debe estar libre de la introducción de parámetros y debe ser de fácil aplicación a un nuevo problema sin la necesidad de realizar modificaciones significativas.

En virtud de lo anterior, se han hecho esfuerzos por desarrollar métodos que incorporen técnicas de aprendizaje para la elección de las heurísticas dentro de una hiperheurística. A continuación se revisará este nuevo enfoque.

3.3.4. Hiperheurísticas con aprendizaje

En este grupo se incluyen las hiperheurísticas que emplean diversas técnicas de aprendizaje de los resultados que han aportado cada heurística de bajo nivel a lo largo del proceso de búsqueda. Esto es, una hiperheurística con aprendizaje selecciona una heurística de bajo nivel en cada iteración basada en la eficacia acumulada de cada heurística desde el inicio de la ejecución hasta el estado actual.

Uno de los mecanismos más populares de aprendizaje que se han empleado en el marco de una hiperheurística se basa en los principios de aprendizaje por refuerzo [38]. La idea general de esta técnica es “recompensar” a la mejor heurística de bajo nivel en cada iteración y “castigar” a la de peor desempeño aumentando o disminuyendo su probabilidad de ser seleccionada. Estas probabilidades van cambiando y adaptándose conforme la búsqueda progresa, reflejando la eficacia de la heurística de bajo nivel en cualquier etapa del proceso de búsqueda.

Nareyek en [53] presenta un método adaptativo basado en el aprendizaje por refuerzo. En su trabajo, Nareyek investigó diferentes esquemas de selección de heurísticas de bajo nivel en el que cada heurística tiene un peso asignado. El peso de cada heurística es modificado inmediatamente después de que ésta haya sido invocada y se haya evaluado su desempeño, elevándolo si mejora el valor de la función objetivo, o disminuyéndolo en caso contrario. Dichos pesos se encuentran acotados por un nivel superior e inferior. En su trabajo se presentan dos enfoques de selección. El primero de ellos es la ruleta, en el cual se selecciona una heurística al azar con una probabilidad proporcional a su peso. El segundo método simplemente selecciona la heurística con el máximo peso.

Cowling et al. [16] introducen un enfoque hiperheurístico basado en la jerarquización de heurísticas de bajo nivel. En este método, la información reciente sobre la evolución de cada heurística es acumulada en una función de elección. La selección de cada heurística de bajo nivel depende del valor actual de dicha función de elección. Dicha función consiste de una suma ponderada de tres componentes que reflejan el desempeño de cada heurística: el resultado obtenido por la heurística en sus dos últimas ejecuciones y el tiempo requerido por la heurística para obtener dicho resultado. Los dos primeros componentes proporcionan la intensificación de la búsqueda mientras que el tercero se incluye para la diversificación. Los pesos de los componentes expresan su importancia en la función de elección la cual es recalculada en cada iteración. La idea general de esta función es que una heurística puede ser escogida por la solución que ésta ha brindado en las últimas ejecuciones o bien por la eficacia que ha presentado.

Las limitaciones que ofrece el trabajo presentado en [16] es que se requiere de un periodo de entrenamiento durante el cual las heurísticas son seleccionadas al azar con el fin de poder inicializar los valores de la función de elección y que los pesos de elección deben ser introducidos manualmente por el usuario. Para superar estas limitaciones, Cowling et al. [17] desarrollaron un procedimiento adaptativo que ajusta automáticamente los parámetros de la función de elección durante la búsqueda.

El desarrollo de hiperheurísticas mediante un modelo paralelizable resulta una idea prometedora. Biazzi et al. [7] realizaron un trabajo enfocado hacia estas nuevas plataformas, en el cual se examina una hiperheurística basada en un modelo de islas, las cuales se comunican entre sí llegando a la conclusión de que esta hiperheurística distribuida resultó ser un buen optimizador. Los autores usaron 8 heurísticas, de las cuales, 6 corresponden a versiones diferentes de la Evolución Diferencial, una a la optimización mediante cúmulo de partículas y una última consistente en una búsqueda aleatoria.

Como se estudió en este capítulo, existen muchos mecanismos de selección que han sido diseñados en los últimos años. Sin embargo, la mayoría de estos trabajos han sido empleados para solucionar problemas de optimización combinatoria en los cuales su efectividad ha sido comprobada en comparación con el uso de heurísticas individuales. En [7] se presenta el diseño de una hiperheurística para resolver problemas de optimización numérica usando un modelo de islas, siendo éste el primer intento de usar hiperheurísticas para resolver este tipo de problemas como lo mencionan los autores.

En virtud de lo anterior, resulta interesante poder diseñar un mecanismo hiperheurístico secuencial, para poder resolver problemas de optimización numérica e incluso con la presencia de restricciones, y poder de esta manera estudiar su comportamiento a fin de medir su eficiencia con respecto al uso de heurísticas que han demostrado obtener buenos resultados trabajando de manera independiente. Este es precisamente el tema que abordamos en esta tesis, usando como motor de búsqueda a la Evolución Diferencial.

Manejo de Restricciones en los AE

En este capítulo se hablará sobre los mecanismos más populares para el manejo de restricciones en los Algoritmos Evolutivos (AE). Dada su importancia en el área, se proporcionará una descripción más detallada del algoritmo de Jerarquización Estocástica. Adicionalmente, se incluye una breve descripción de un algoritmo basado en Evolución Diferencial para espacios restringidos.

4.1. Introducción

Muchos de los problemas de optimización que existen en el mundo real están sujetos a un conjunto de restricciones. Las técnicas de programación matemática utilizadas en la solución a este tipo de problemas cuentan con varias limitaciones. Entre la más importante se puede mencionar que muchas de estas técnicas requieren que la función objetivo y las restricciones sean diferenciables. También suelen requerir que tanto la función objetivo como las restricciones se puedan expresar de manera algebraica. De hecho, las condiciones de optimalidad de Kuhn-Tucker en las que se basan muchos de estos métodos, no se cumplen en el caso general de optimización no lineal lo que hace que el problema general de optimización no lineal permanezca como un área abierta de investigación.

Los Algoritmos Evolutivos (AE) han sido empleados exitosamente en la solución de una gran variedad de problemas de optimización incluyendo varios con características como las antes descritas. Sin embargo, los AE son técnicas de búsqueda sin restricciones, de tal suerte que resulta atractivo poder incorporar esquemas de manejo de restricciones a dichos algoritmos.

Coello en [13] propone una taxonomía para las principales técnicas de manejo de restricciones en los AE. Dicha clasificación se lista a continuación:

- Funciones de penalización
- Operadores y representaciones especiales
- Separación de restricciones y objetivos
- Métodos híbridos

A continuación se hará una breve introducción a cada uno de los métodos antes mencionados.

4.2. Funciones de penalización

Esta es la técnica de manejo de restricciones que más comúnmente se utiliza para incorporar las restricciones de un problema a los Algoritmos Evolutivos. La idea de esta técnica es transformar un problema de optimización con restricciones en uno sin restricciones al agregar a la función objetivo un grado de penalización resultado de sumar un cierto valor cuando las restricciones, tanto de igualdad como de desigualdad, son violadas. Existen dos clases de penalización:

1. **Penalización exterior**.- en este caso se permiten soluciones infactibles al inicio del proceso de búsqueda y se busca guiar al algoritmo hacia la región factible del espacio de búsqueda ajustando el valor de penalización.
2. **Penalización interior**.- a diferencia del caso anterior, en este caso únicamente se permiten soluciones factibles al inicio del proceso de búsqueda, con la finalidad de poder mantener la búsqueda en el interior de la región factible. Por lo tanto, la penalización actúa como una barrera que impide que las soluciones salgan de la región factible.

En muchos problemas, generar soluciones que estén sólo en la región factible es un problema extremadamente difícil, razón por la cual en los AE se ha optado principalmente por usar esquemas de penalización exterior.

La función de penalización exterior se define de la siguiente forma:

$$\phi(\vec{x}) = f(\vec{x}) \pm \left[\sum_{i=1}^m r_i \cdot G_i + \sum_{j=1}^p c_j \cdot L_j \right] \quad (4.1)$$

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

Para las funciones G_i y L_j las formas más comunes son:

$$G_i = \max[0, g_i(\vec{x})]^\beta \quad (4.2)$$

$$L_j = |h_j(\vec{x})|^\gamma \quad (4.3)$$

donde β y γ normalmente toman los valores de uno o dos. Sin embargo, es necesario mencionar que la mayoría de los mecanismos para el manejo de restricciones en los AE únicamente trabajan en problemas con restricciones de desigualdad. Sin embargo, una restricción de igualdad puede transformarse en una de desigualdad de la siguiente forma:

$$|h_j(\vec{x})| - \varepsilon \leq 0 \quad (4.4)$$

donde ε representa la tolerancia permitida.

El principal problema de las funciones de penalización es la necesidad de ajustar los factores de penalización r_i y c_j con el fin de lograr el correcto funcionamiento del algoritmo. Si la penalización es muy alta o muy baja, entonces el problema puede ser muy difícil de resolver por un Algoritmo Evolutivo.

Richardson et. al. [62] definieron algunas reglas para diseñar una función de penalización. Algunas de estas reglas son las siguientes:

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

Se han propuesto diferentes esquemas de penalización en la literatura. A continuación se explican brevemente las principales que engloban a estos esquemas.

4.2.1. Pena de muerte

La pena de muerte consiste en el rechazo total de las soluciones infactibles y es la forma más simple de manejar restricciones. Una variante relativamente popular de este método consiste en asignar una aptitud de cero a un individuo que no sea factible, y tomar el valor de la función de aptitud para los que sí lo sean. Esta técnica es muy eficiente porque no es necesario re-calcular las restricciones ni la función objetivo. Sin embargo, no es recomendable en el caso general debido a que la búsqueda puede quedar estancada al no poderse generar soluciones factibles. Michalewicz en [48] concluyó que esta técnica sólo puede usarse en espacios de búsqueda convexos y en aquellos casos en los que la zona factible constituya una parte razonablemente grande del espacio de búsqueda total.

4.2.2. Penalizaciones estáticas

En este esquema los factores de penalización permanecen sin cambio durante el proceso de búsqueda. Esto no resulta muy recomendable ya que conforme el proceso evolutivo avanza, varía la porción explorada de la región factible y es posible que ya no interese explorar dicha región en las etapas finales de la búsqueda. Otro inconveniente que presenta este mecanismo se debe al hecho de que los factores son generalmente dependientes del problema y en algunos casos el usuario debe definir un número alto de coeficientes de penalización.

Homaifar, Lai y Qi [35] propusieron un enfoque en el cual usan varios niveles de violación de las restricciones y un coeficiente de penalización que es elegido de tal manera que aumente

a medida que se alcanzan niveles más altos de violación de restricciones. El individuo es evaluado de acuerdo a la siguiente ecuación:

$$\phi(\vec{x}) = f(\vec{x}) + \sum_{i=1}^m (R_{k,i} \times \max [0, g_i(\vec{x})^2]) \quad (4.5)$$

donde $R_{k,i}$ son los coeficientes usados, m es el número de restricciones (donde las restricciones de igualdad son transformadas a restricciones de desigualdad), $f(\vec{x})$ es la función objetivo y $k = 1, 2, \dots, l$ donde l es el número de niveles de violación definidos por el usuario. El principal inconveniente de este enfoque es el gran número de parámetros requeridos, ya que por cada m restricciones se requieren definir $m(2l + 1)$ parámetros.

Kuri [45] introdujo otro enfoque basado en una penalización estática. Un individuo es evaluado de acuerdo a la siguiente ecuación:

$$\phi(\vec{x}) = \begin{cases} f(\vec{x}) & \text{si la solución es factible} \\ K - \sum_{i=1}^s (\frac{K}{m}) & \text{de lo contrario} \end{cases} \quad (4.6)$$

donde s es el número de restricciones no satisfechas, m es el total de restricciones y K es una constante (donde $K = 1 \times 10^9$ en los experimentos reportados en [45]). Nótese que cuando un individuo es infactible, el valor de la función objetivo no es calculado y todos los individuos que violen el mismo número de restricciones recibirán el mismo grado de penalización, independientemente de qué tan cerca estén de la región factible.

4.2.3. Penalizaciones dinámicas

En este esquema, a diferencia del caso anterior, los factores de penalización se ajustan conforme avanza el proceso de búsqueda. Normalmente, los factores de penalización aumentan con respecto al tiempo (generaciones). Algunos investigadores han asegurado que este mecanismo funciona mejor que las penalizaciones estáticas. Sin embargo, los problemas presentes en las penalizaciones estáticas también aplican a las penalizaciones dinámicas ya que resulta de igual manera difícil en este caso derivar buenas funciones de penalización y producir buenos factores de penalización.

Joines y Houck [37] propusieron una técnica en la cual los individuos son evaluados en cada generación t incrementando la penalización conforme el proceso evolutivo avanza. Usan la siguiente ecuación para determinar la aptitud de cada individuo:

$$\phi(\vec{x}) = f(\vec{x}) + (C \times t)^\alpha \times SVC(\beta, \vec{x}) \quad (4.7)$$

donde C , α y β son constantes definidas por el usuario y SVC es definida como:

$$SVC(\beta, \vec{x}) = \sum_{i=1}^n D_i^\beta(\vec{x}) + \sum_{j=1}^p D_j(\vec{x}) \quad (4.8)$$

donde:

$$D_i(\vec{x}) = \begin{cases} 0 & g_i(\vec{x}) \leq 0 \\ |g_i(\vec{x})| & \text{de lo contrario} \end{cases}$$

y

$$D_j(\vec{x}) = \begin{cases} 0 & -\epsilon \leq h_j(\vec{x}) \leq \epsilon \\ |h_j(\vec{x})| & \text{de lo contrario} \end{cases}$$

Kazarlis y Paredis [39] realizaron un estudio detallado de una función dinámica de penalización de la forma:

$$\phi(\vec{x}) = f(\vec{x}) + V(g) \times \left(A \sum_{i=1}^m (\delta_i \cdot w_i \cdot \Phi(d_i(S))) + B \right) \times \delta_s \quad (4.9)$$

donde A es un factor de “severidad”, m es el número total de restricciones, δ_i es 1 si la i -ésima restricción es violada y 0 en otro caso, w_i es un factor de peso para la i -ésima restricción, $d_i(S)$ es una medida del grado de la restricción i introducido por la solución S , $\Phi_i()$ es una función de esta medida, B es un factor umbral de penalización, δ_s es un factor binario (1 si S es infactible y 0 en otro caso) y $V(g)$ es una función creciente de $g \in (0, \dots, 1)$ (la generación actual).

Usando un conjunto de funciones, Kazarlis y Paredis experimentaron con diferentes formas para la función $V(g)$ encontrando que el mejor rendimiento lo proporcionaba la función de la forma:

$$V(g) = \left(\frac{g}{G} \right)^2 \quad (4.10)$$

donde G es el número total de generaciones.

4.2.4. Penalizaciones adaptativas

Para este tipo de mecanismos se usa información generada por el proceso de búsqueda. En esta clase de propuesta se regula de manera más “inteligente” la forma de los factores de penalización. Un aspecto interesante de estos métodos es que tratan de mantener una combinación de soluciones factibles e infactibles en la población, tratando de evitar que todos los individuos sean factibles o infactibles.

Bean y Hadj-Alouane [5, 30] desarrollaron un método que usa una función de penalización la cual usa retroalimentación del proceso de búsqueda. Cada individuo es evaluado de acuerdo a la siguiente ecuación:

$$\phi(\vec{x}) = f(\vec{x}) + \lambda(t) \left[\sum_{i=1}^n g_i^2(\vec{x}) + \sum_{j=1}^p |h_j(\vec{x})| \right] \quad (4.11)$$

donde $\lambda(t)$ es actualizado en cada generación t de la siguiente manera:

$$\lambda(t+1) = \begin{cases} \left(\frac{1}{\beta_1} \right) \cdot \lambda(t) & \text{caso 1} \\ \beta_2 \cdot \lambda(t) & \text{caso 2} \\ \lambda(t) & \text{de lo contrario} \end{cases} \quad (4.12)$$

donde $\beta_1, \beta_2 > 1$, $\beta_1 > \beta_2$ y $\beta_1 \neq \beta_2$. Caso 1 y caso 2 denotan las situaciones donde el mejor individuo en las últimas k generaciones fue siempre factible o nunca lo fue, respectivamente.

En otras palabras, el componente de penalización $\lambda(t+1)$ para la generación $t+1$ se decrementa si todos los mejores individuos de las últimas k generaciones fueron factibles. De lo contrario, se incrementa si todos ellos fueron infactibles.

Smith y Tate [66] propusieron un enfoque en el que la magnitud de la penalización es dinámicamente modificada de acuerdo al valor de aptitud de las mejores soluciones encontradas hasta el momento. Un individuo es evaluado usando la siguiente fórmula.

$$\phi(\vec{x}) = f(\vec{x}) + (B_{factible} - B_{all}) \sum_{i=1}^n \left(\frac{g_i(\vec{x})}{\text{NFT}(t)} \right) \quad (4.13)$$

donde $B_{factible}$ es el mejor valor de función objetivo conocido hasta la generación t , B_{all} es el mejor valor de la función objetivo no sancionado en la generación t , $g_i(\vec{x})$ es la cantidad de restricciones violadas, k es una constante que ajusta la severidad de la violación y NFT es el llamado *Umbral de Cercanía Factible*, el cual es definido como el umbral de distancia entre la región factible el cual es definido por el usuario y que considera que la búsqueda está razonablemente cerca de la zona factible.

4.3. Operadores y representaciones especiales

Algunos autores han decidido desarrollar esquemas de representación para problemas en particular en los que en ocasiones un esquema genérico puede no ser apropiado. Dicho cambio de representación trae como consecuencia la necesidad de diseñar operadores especiales que trabajen de manera similar que los operadores tradicionales.

Michalewicz [49] propuso el *Algoritmo Genético para Optimización Numérica de Problemas Restrictivos* (**GENOCOP** por sus siglas en inglés). Este algoritmo inicia con una solución factible y sus operadores ejecutan combinaciones lineales de los padres seleccionados con el fin de que los descendientes sean factibles también. La principal desventaja de este método es la necesidad de una solución factible, lo que en muchos problemas es algo muy difícil de obtener, como se ha discutido anteriormente. Además, únicamente maneja restricciones lineales y espacios convexos por lo cual su utilidad es muy limitada, debido a que existen técnicas de programación matemática que resuelven estos problemas garantizando obtener la solución óptima en tiempos razonablemente cortos.

Kowalczyk [44] propuso el uso de consistencia en las restricciones, asegurándose que las soluciones producidas por un AG sólo sean soluciones factibles. Kowalczyk usó una representación de valores reales y definió operadores genéticos especiales y un procedimiento especial para inicializar la población. En su trabajo indica que su enfoque puede ser usado en combinación con cualquier otro método de manejo de restricciones, además de indicar que en muchos casos incluir soluciones parcialmente factibles puede ser preferible para orientar la búsqueda de una manera más adecuada.

4.4. Separación de restricciones y objetivos

Existen varias propuestas en las cuales las restricciones y la función objetivo se consideran de manera independiente, con lo cual no se combina la información de la función objetivo con el grado de violación de las restricciones. Paredis [57] propuso una técnica basada en un modelo co-evolutivo en el cual existen dos poblaciones; la primera de ellas contiene las restricciones y la segunda las posibles soluciones (factibles e infactibles). Paredis indicó que su aproximación es similar a una función de penalización auto-adaptada.

Hinterding y Michalewicz [33] propusieron un enfoque llamado **CONGA** (*Constraint based Numeric Genetic Algorithm*). La idea es realizar la búsqueda en dos fases. En la primera fase, la búsqueda se concentra en encontrar individuos factibles (asumiendo que no hay ninguno en la población inicial). En esta etapa no se usa la información de la función objetivo, sólo se usa la proporcionada por el grado de violación de las restricciones. Conforme el número de individuos factibles crece, la búsqueda se enfoca en ajustar a los mejores de ellos.

También hay varios enfoques que se basan en conceptos de optimización multiobjetivo [56]. Dado que una restricción puede verse como un objetivo en el que hay que minimizar el grado de violación, se puede agregar la función objetivo al conjunto de restricciones, siendo todas éstas los objetivos por optimizar.

4.5. Métodos híbridos

En esta técnica se incluyen métodos que combinan dos o más tipos de técnicas para el manejo de restricciones. Adeli y Cheng [2] propusieron un Algoritmo Evolutivo que utiliza multiplicadores de Lagrange combinados con una función de penalización, usando una función de aptitud de la forma:

$$\phi(\vec{x}) = f(\vec{x}) + \frac{1}{2} \sum_{j=1}^m \gamma_j \{ [g_j(\vec{x}) + \mu_j]^+ \}^2 \quad (4.14)$$

$$[g_j(\vec{x}) + \mu_j]^+ = \text{máx} \{ 0, g_j(\vec{x}) + \mu_j \}$$

donde $\gamma_j > 0$. μ_j son parámetros asociados a la j -ésima restricción y m es el número total de restricciones.

Kim y Myung [57] propusieron otro algoritmo evolutivo que emplea una función extendida de Lagrange la cual garantiza la generación de soluciones factibles durante el proceso de búsqueda. Durante la primera fase del algoritmo el objetivo a minimizar es el siguiente:

$$\phi(\vec{x}) = f(\vec{x}) + \frac{C}{2} \left(\sum_{i=1}^n (\text{máx} [0, g_i])^2(\vec{x}) + \sum_{j=1}^p |h_j(\vec{x})|^2 \right) \quad (4.15)$$

donde C es una constante. Una vez que la primera fase termina, (o sea, cuando el nivel de violación de restricciones se ha reducido tanto como el usuario requiera), se inicia la segunda fase. Durante esta última fase se usan los multiplicadores de Lagrange de una manera similar a la propuesta de Adeli y Cheng [2].

4.6. Otros Métodos

Existe otra serie de técnicas para el manejo de restricciones basadas en otros tipos de heurísticas tales como el Sistema Inmune Artificial [18, 55] y la Colonia de Hormigas [21]. Sin embargo, la que muchos consideran como una de las técnicas de manejo de restricciones más efectivas de la actualidad es la denominada *Jerarquización Estocástica* [63]. A continuación proporcionamos una descripción de este método así como otro igualmente efectivo que se propuso específicamente para el algoritmo de Evolución Diferencial y que es de gran relevancia para este trabajo de tesis.

4.6.1. Jerarquización estocástica

Fue propuesta por Thomas Runarsson y Xin Yao [63], y tiene como objetivo balancear la influencia de la función objetivo y el grado de violación de las restricciones al momento de asignar el valor de aptitud a un individuo. Este método de manejo de restricciones ha sido uno de los más exitosos que se han propuesto a la fecha y sigue siendo un método de referencia contra el cual los nuevos métodos de manejo de restricciones deben compararse.

El aspecto más interesante de esta propuesta es que no requiere definir los factores de penalización, ya que introduce directamente una jerarquización de los individuos de la población a través de un procedimiento similar al ordenamiento de la burbuja. La comparación entre dos individuos adyacentes puede ser con base en la función objetivo o el grado de penalización que presentan. Tal elección se lleva a cabo de manera aleatoria mediante una probabilidad P_f definida por el usuario. Los autores determinaron experimentalmente que se logran los mejores resultados cuando $0.4 \leq P_f \leq 0.5$. Este procedimiento se describe en el algoritmo 7, donde $U(0, 1)$ es un generador de números aleatorios uniformemente distribuidos entre 0 y 1 y λ es el número de individuos a jerarquizar (definido también por el usuario). La función de penalización adoptada por este método es la siguiente:

$$\psi(\vec{x}) = \sum_{i=1}^m \max\{0, g_i(\vec{x})\}^2 + \sum_{j=1}^p |h_j(\vec{x})|^2 \quad (4.16)$$

En su forma original, este método usa una estrategia evolutiva multimiembro como motor de búsqueda, la cual realiza mutaciones generadas con una distribución normal de probabilidades. Además, adopta un coeficiente que regula el tamaño de paso por cada variable de decisión en cada individuo. En caso de que alguna variable quede fuera del rango permitido después del proceso de mutación, ésta toma el valor del límite correspondiente. Cabe mencionar que no se emplea ningún operador de recombinación.

Los autores presentaron posteriormente una versión mejorada del algoritmo [64] permaneciendo el mismo mecanismo para el manejo de restricciones pero cambiando el motor de búsqueda empleado. La diferencia se encuentra en el proceso de mutación en el cual se suavizan los tamaños de paso, además de que se incluye una recombinación discreta del tipo de la adoptada en la Evolución Diferencial, en caso de que las variables generadas después del proceso de mutación hayan quedado fuera de los límites del problema.

```

1  $P = \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_\lambda\}$ 
2 for  $i \leftarrow \{1, 2, \dots, \lambda\}$  do
3   for  $j \leftarrow \{1, 2, \dots, \lambda - 1\}$  do
4     if  $\psi(\vec{x}_j) = \psi(\vec{x}_{j+1}) = 0$  o  $U(0,1) < P_f$  then
5       if  $f(\vec{x}_j) > f(\vec{x}_{j+1})$  then
6         └ Intercambiar( $\vec{x}_j, \vec{x}_{j+1}$ )
7       else
8         if  $\psi(\vec{x}_j) > \psi(\vec{x}_{j+1})$  then
9           └ Intercambiar( $\vec{x}_j, \vec{x}_{j+1}$ )
10    if No hubo intercambio then
11      └ Terminar()

```

Algoritmo 7: Algoritmo de la jerarquización estocástica

Los autores reportan resultados realizando 350,000 evaluaciones de la función objetivo. La validación la realizan con trece funciones de prueba las cuales se encuentran en el apéndice C de esta tesis.

4.6.2. Evolución diferencial restringida

En la literatura especializada la mayoría de los algoritmos evolutivos para optimización en espacios restringidos usan funciones de penalización, y si nos restringimos únicamente al algoritmo de la ED, éste no es la excepción. Sin embargo, Lampinen et al. [46] propusieron una modificación al proceso de selección de ED con la finalidad de poder manejar restricciones. Esta propuesta evita usar funciones de penalización, además de que sólo modifica el proceso de selección manteniendo la estructura del algoritmo original de la ED.

En la ecuación (4.17) se muestra el operador de selección propuesto entre el vector de descendientes \vec{u}_i y el i -ésimo individuo en la población $\vec{x}_{i,G}$, para determinar al i -ésimo individuo de la siguiente generación. En este caso se considera un problema de minimización con m restricciones de desigualdad (asumiendo que las restricciones de igualdad han sido transformadas).

$$\vec{x}_{i,G+1} = \begin{cases} \vec{u}_i & \text{Si } \begin{cases} \forall j \in \{1, \dots, m\} : g_j(\vec{u}_i) \leq 0 \text{ y } g_j(\vec{x}_{i,G}) \leq 0 \text{ y } f(\vec{u}_i) \leq f(\vec{x}_{i,G}) \\ \forall j \in \{1, \dots, m\} : g_j(\vec{u}_i) \leq 0 \text{ y } \exists j \in \{1, \dots, m\} g_j(\vec{x}_{i,G}) > 0 \\ \exists j \in \{1, \dots, m\} : g_j(\vec{u}_i) > 0 \text{ y } \forall j \in \{1, \dots, m\} : \max[g_j(\vec{u}_i), 0] \leq \max[g_j(\vec{x}_{i,G}), 0] \end{cases} \\ \vec{x}_{i,G} & \text{de lo contrario} \end{cases} \quad (4.17)$$

Empleando el criterio anterior, el vector \vec{u}_i es seleccionado para la siguiente generación si:

- Satisface todas las restricciones y provee un valor de la función objetivo igual o mejor.
- Si \vec{u}_i es factible y $\vec{x}_{i,G}$ no lo es.

- Es infactible pero provee una violación de restricción menor o igual para cada una de las m restricciones.

Jiménez et al. [36] y Deb et al. [20] propusieron un mecanismo muy similar al de Lampinen en [46]. En este caso, se adopta un torneo binario con las reglas siguientes:

- Si las dos soluciones son factibles, aquella con mejor valor de aptitud gana.
- Si una solución es infactible y la otra es factible, la solución factible gana.
- Si ambas soluciones son infactibles, aquella con menor grado de violación de restricciones gana.

El cambio ocurre en la última regla. En este caso, cuando las dos soluciones son infactibles se selecciona a \vec{u}_i si no viola a lo más una restricción más que el vector \vec{x}_i . Este criterio es análogo al concepto de optimalidad de Pareto empleado en optimización multi-objetivo.

Por lo anterior, el criterio de la ecuación (4.17) únicamente acepta un reemplazo en la población cuando se obtienen mejoras en todas las restricciones, limitando mucho su capacidad de exploración, puesto que es muy poco probable que el operador de recombinación genere soluciones que mejoren todas las restricciones. Además, el mecanismo de manejo de restricciones no es capaz de explotar de manera eficiente la frontera de la zona factible puesto que existe una presión de selección muy alta hacia las zonas factibles del espacio de búsqueda, teniendo como principal desventaja no incluir ningún mecanismo de diversidad entre individuos factibles e infactibles.

Como consecuencia de lo anterior, el algoritmo de Evolución Diferencial Restringida requiere de muchas evaluaciones de la función objetivo en la mayoría de los problemas de prueba. Por esta razón, como se discutirá en el siguiente capítulo, en nuestra propuesta se ha decidido hibridizar el algoritmo de Evolución Diferencial en sus diversos modelos con el mecanismo de ordenamiento de los individuos propuesto en la Jerarquización Estocástica, con el objetivo de guiar las direcciones de búsqueda tanto a la región factible como a la región infactible y poder explorar la frontera entre ambas regiones.

Propuesta de Hiperheurística Basada en ED

En este capítulo se presenta la descripción completa de la hiperheurística propuesta en esta tesis, la cual se basa en Evolución Diferencial. Veremos la propuesta tanto para problemas sin restricciones como para aquellos que sí las tienen.

La primera parte del capítulo trata acerca del nuevo modelo de hiperheurística propuesto para optimización global sin restricciones, haciendo énfasis en el mecanismo de selección de heurísticas de bajo nivel. En la segunda parte, se detalla la incorporación del mecanismo de restricciones a la propuesta.

5.1. Modelo propuesto para espacios sin restricciones

5.1.1. Estudio comparativo

Como se mencionó en la sección 2.4.3, en la pág. 14, existen diversas variantes del algoritmo de la Evolución Diferencial (ED). En [74] se realiza un estudio comparativo de los diversos modelos de ED considerando los resultados obtenidos y el número de evaluaciones de la función objetivo requeridas, concluyendo que el modelo **ED/rand/1/bin** muestra muy buen desempeño en la mayoría de las funciones de prueba. Sin embargo, el estudio presentado tiene la gran desventaja de requerir una configuración específica de parámetros de la ED para cada una de las funciones de prueba empleadas.

Aunque en la literatura especializada se pueden encontrar estudios de los modelos de la Evolución Diferencial, para nuestro trabajo es necesario realizar un análisis de las diferentes variantes de ED trabajando en conjunto y no de manera independiente, ya que el comportamiento de éstas al trabajar colectivamente puede no ser el mismo que al ser empleadas de manera individual para resolver un problema. Para ello se implementó el mecanismo de selección **aleatorio descendente** propuesto en [16]. Este es uno de los mecanismos de selección más sencillos de implementar, y se eligió con el fin de encontrar las características y comportamientos reales de los diferentes modelos de ED trabajando en conjunto para resolver diversos problemas sin restricciones. En el algoritmo 8 se muestra el pseudocódigo de la hiperheurística implementada para evaluar el rendimiento de las variantes en conjunto.

```

/* Conjunto de heurísticas */
1  $heurísticas = \{h_1, h_2, \dots, h_n\}$ 
  /* Inicialización */
2  $iter \leftarrow 0$ 
3  $G \leftarrow 0$ 
4 Inicializar  $P_{x,G}$ 
5 while Criterio de terminación no satisfecho do
  /* Selección aleatoria de la heurística a usar */
6    $sel \leftarrow U(1, n)$  donde  $n$  es el número de heurísticas
7   repeat
8     aplicar_Heurística( $h_{sel}$ )
9      $G \leftarrow G + 1$ 
10  until no mejore la solución previa ;

```

Algoritmo 8: Hiperheurística con mecanismo de selección aleatorio descendente

Para evaluar dicho rendimiento, se utilizó un conjunto de funciones de prueba estándar, tomadas de artículos donde se usan algoritmos evolutivos para resolver problemas de optimización numérica sin restricciones [77]. El estudio consistió en efectuar 100 ejecuciones para cada una de las funciones de prueba que se describen en el Apéndice A al final de este documento.

Como criterio de terminación se empleó un valor de error absoluto de 1×10^{-11} entre la mejor y la peor solución de la población actual. Se fijó el número máximo de generaciones en 10,000, con el propósito de que el algoritmo lograra converger cuando el número de variables de decisión del problema aumentaba. Se hicieron experimentos variando el tamaño de población con 25, 50, 75 y 100 individuos, a partir de estos valores se determinó experimentalmente que el algoritmo ofrecía los mejores resultados al usar un tamaño de población $N = 50$. Para el parámetro R se efectuaron pruebas con valores de 0.2 y 0.8, con el objetivo de observar el comportamiento al variar este parámetro cerca del límite inferior y superior.

Como se vio en la sección 2.4, la Evolución Diferencial (ED) emplea un parámetro de control F que regula las magnitudes de las diferencias del vector mutación. En la literatura, el valor de este parámetro se define como $F \in [0, 1)$ [70]. Pero de manera experimental, en [51] se determinó que el comportamiento del algoritmo no sufre efectos adversos al generar de manera aleatoria el valor de F en el intervalo $[0.3, 0.9]$ durante cada generación. De tal suerte, en nuestra propuesta adoptamos este criterio haciéndose innecesario establecer un valor predeterminado para este parámetro.

Las funciones de prueba usadas para realizar el estudio comparativo son escalables en el número de variables de decisión D usadas, siendo el valor óptimo de la función el mismo para cada uno de los problemas. Para nuestro estudio se optó por usar cuatro valores para el número de variables de decisión: 10, 30, 50 y 100, con el objeto de poder observar el comportamiento global del algoritmo ante el incremento de las variables de decisión.

Finalmente, el conjunto de heurísticas de bajo nivel consta de doce variantes de la ED.

En la tabla 5.1 se muestra cada una de las variantes empleadas como heurística de bajo nivel de la hiperheurística.

Variantes con cruza <i>exp</i>		Variantes con cruza <i>bin</i>	
h_1	ED/best/1/exp	h_7	ED/best/1/bin
h_2	ED/rand/1/exp	h_8	ED/rand/1/bin
h_3	ED/current-to-best/1/exp	h_9	ED/current-to-best/1/bin
h_4	ED/best/2/exp	h_{10}	ED/best/2/bin
h_5	ED/rand/2/exp	h_{11}	ED/rand/2/bin
h_6	ED/current-to-rand/1/exp	h_{12}	ED/current-to-rand/1/bin

Tabla 5.1: Conjunto de heurísticas de bajo nivel

Población inicial

Como en todo algoritmo poblacional es necesario generar la población inicial que esta constituida por un conjunto de individuos que representan las posibles soluciones al problema. Esta puede ser creada de muy diversas formas, desde generar aleatoriamente cada gen para cada individuo, utilizar alguna función o generar alguna parte de cada individuo y luego aplicar una búsqueda local [29]. Es importante garantizar que dentro de la población inicial se tenga la diversidad necesaria para tener una representación de la mayor parte del espacio de búsqueda o al menos evitar la convergencia prematura.

Las secuencias de baja discrepancia, también denominadas secuencias **cuasi-aleatorias** o **sub-aleatorias**, ofrecen un grado de uniformidad más alto con respecto a la distribución uniforme de un conjunto de números pseudo-aleatorios común. En un conjunto de puntos de baja discrepancia, cada nuevo punto es generado por un mecanismo que considera la restricción de alta correlación con números precedentes, evitando los fenómenos de agrupamiento y baja cobertura de las secuencias de puntos generados por mecanismos pseudo-aleatorios.

El mecanismo clásico para la generación de secuencias de baja discrepancia fué presentado por van der Corput en 1935 para el caso unidimensional y continúa hoy en día siendo la base para varios generadores de secuencias cuasi-aleatorias [76]. Las secuencias de Halton [31] y Hammersley [32] son una generalización de la secuencia de van der Corput para espacios multidimensionales, en donde se utiliza un número primo como base por cada dimensión del espacio a muestrear. La secuencia de Sobol' [67] se define mediante una generalización de la secuencia de van de Corout y tiene una forma similar a la secuencia de Halton, pero utiliza una única base para todas las dimensiones e incorpora un mecanismo de reordenamiento de elementos.

En este trabajo se propone el uso de una secuencia de baja discrepancia para generar la población inicial de nuestro algoritmo haciendo uso de la secuencia de Halton. Supongamos que se requiere generar al i -ésimo vector de la secuencia de Halton en \mathbb{R}^n . El primer paso es generar n primos relativos p_1, p_2, \dots, p_n , comúnmente los primeros n números primos. Para construir el i -ésimo vector de la secuencia, se considera la representación en base p de i , la

cual toma la forma $i = a_0 + pa_1 + p^2a_2 + \dots + p^ra_r$. Cada coordenada del vector está en el intervalo $[0, 1]$ y se obtiene de la siguiente manera:

$$r(i, p) = \frac{a_0}{p} + \frac{a_1}{p^2} + \frac{a_2}{p^3} + \dots + \frac{a_r}{p^{r+1}} \quad (5.1)$$

De tal manera, el i -ésimo vector de la secuencia de Halton está dado por:

$$\langle r(i, p_1), r(i, p_2), \dots, (i, p_n) \rangle \quad (5.2)$$

Input: i (i -ésimo vector), j (j -ésimo componente del vector i)
Output: $c \in [0, 1]$ (j -ésima componente en la secuencia de Halton)

```

1 begin
  /* Definir los primeros n números primos */
2  primos[n] ← [2, 3, 5, ...]
3  p1 ← primos[j]
4  p2 = p1
5  c ← 0
6  repeat
7    x ← i % p1
8    c ← c + x/p2
9    i ← i/p1
10   p2 ← p2 · p1
11  until i ≤ 0 ;
12  return c
13 end

```

Algoritmo 9: Secuencia de Halton

El algoritmo 9, muestra el pseudo-código para obtener la j -ésima componente del vector i de la secuencia de Halton. El valor c que regresa el algoritmo está en el intervalo $[0, 1]$ el cual puede ser mapeado a un intervalo distinto mediante la siguiente ecuación:

$$c' = \text{intervalo}_{inferior} + c \cdot (\text{intervalo}_{superior} - \text{intervalo}_{inferior}) \quad (5.3)$$

La entropía constituye una medida que puede interpretarse de dos formas: como el grado de desorden o la incertidumbre asociada a una variable aleatoria X . Su cálculo se realiza a partir de su distribución de probabilidad $p(x)$ mediante la siguiente ecuación:

$$H(X) = \sum_{i=1}^n p(x_i) \log_a \left(\frac{1}{p(x_i)} \right) = - \sum_{i=1}^n p(x_i) \log_a (p(x_i)) \quad (5.4)$$

donde $0 \leq H(X) \leq \log_a(n)$, X representa la variable aleatoria y tiene un conjunto de valores definidos x_1, x_2, \dots, x_n con probabilidades respectivas $p(x_1), p(x_2), \dots, p(x_n)$. La entropía conjunta de un conjunto de variables aleatorias independientes X_1, X_2, \dots, X_m esta dada por:

$$H(X_1, X_2, \dots, X_m) = H(X_1) + H(X_2) + \dots + H(X_m) \quad (5.5)$$

Si las variables aleatorias X_1, X_2, \dots, X_m se distribuyen de forma idéntica, entonces:

$$H(X_1, X_2, \dots, X_m) = mH(X_1) \quad (5.6)$$

Se considera que las unidades de entropía se dá en logaritmos, los cuales se pueden tomar con respecto a cualquier base. Si se toma en base dos la unidad se denomina *bit*, para la base diez la unidad correspondiente se denomina *dit* o unidad de Hartley y si se toma en base e la unidad correspondiente se denomina *nat*.

En la figura 5.1 se muestra el comportamiento de la entropía para las secuencias uniforme y de Halton, variando el número de muestras que conforman a una ($m = 1$) y dos variables aleatorias ($m = 2$) respectivamente. De acuerdo a la figura se puede observar que la secuencia de Halton presenta un valor mayor de entropía que la secuencia uniforme, mostrando que la generación de las muestras producidas por la secuencia de Halton tiende a ser mas dispersa con respecto a la distribución uniforme de un conjunto de números pseudo-aleatorios común.

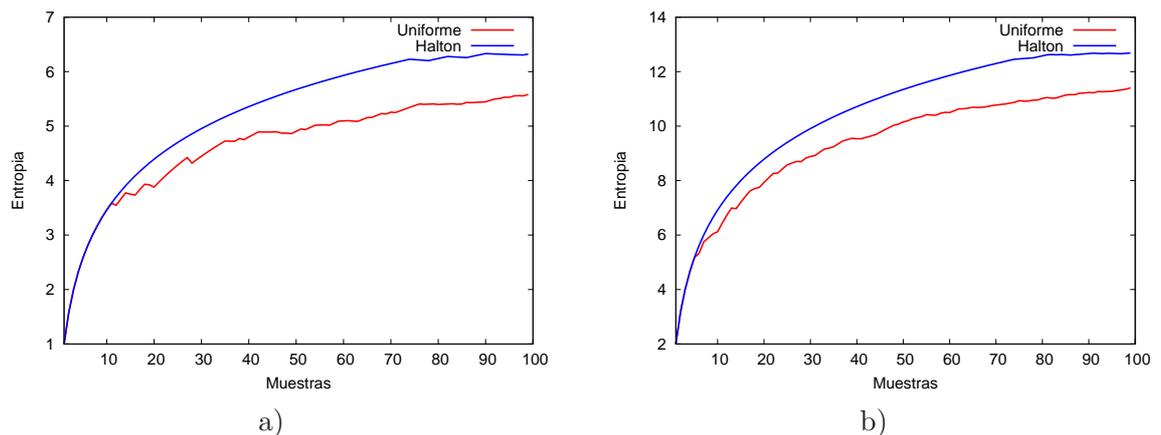


Figura 5.1: Comparativa de la entropía entre la secuencia de Halton y una distribución uniforme para: a) una variable aleatoria $X = \{1, 2, \dots, 100\}$ con distribución $p(x)$ y b) dos variables aleatorias X, Y con distribución $p(x, y)$ donde $X = Y$

En la figura 5.2 se presenta de manera gráfica un conjunto de puntos generados a partir de dos distribuciones, una distribución uniforme y otra usando la secuencia de Halton. Como se observa, la secuencia de Halton tiene una mejor distribución de puntos con respecto a la distribución uniforme. En los experimentos realizados, notamos que al inicializar la población usando la secuencia de Halton se producían mejores resultados con respecto a una inicialización con una distribución uniforme al requerir de una menor cantidad de generaciones para encontrar la solución óptima a los problemas. Por la razón anterior y la propiedad de distribución de la secuencia de Halton descartamos la distribución uniforme para inicializar la población de nuestro algoritmo de la Evolución Diferencial, quedando la generación de esta a partir de la secuencia de Halton.

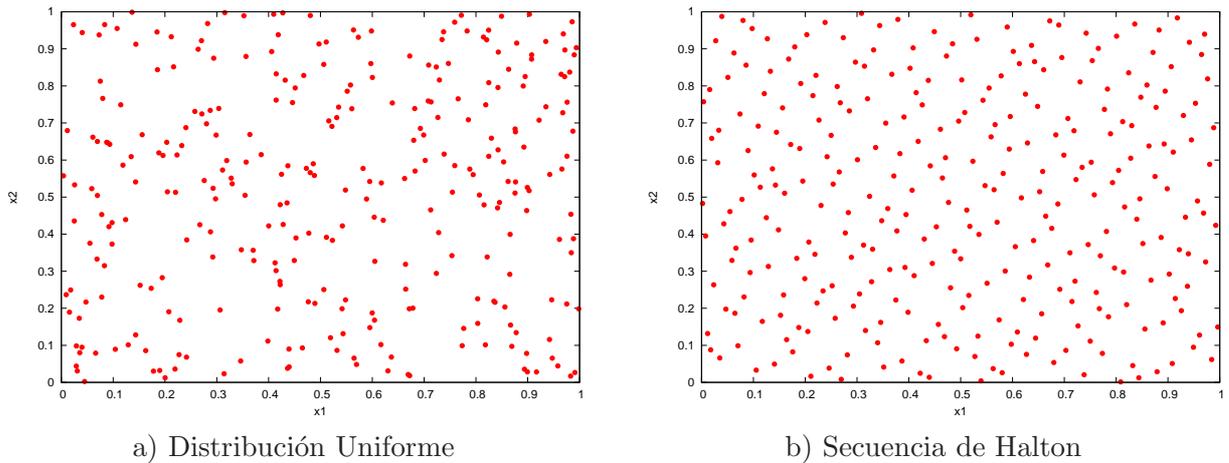


Figura 5.2: Trescientos puntos creados con: a) una distribución uniforme y b) la secuencia de Halton en \mathbb{R}^2 en el intervalo $[0, 1]$

Resultados

En las tablas 5.2 y 5.3 se muestran los resultados promedio de 100 ejecuciones independientes para los cuatro valores de número de variables de decisión, usando valores de R , la constante de recombinación para la ED, de 0.2 y 0.8, respectivamente. De ambas tablas se observa que, en términos generales, con un valor $R = 0.2$ se obtienen las mejores soluciones para cada problema. Para el caso de $R = 0.8$, se ofrecen resultados muy pobres para f_1 , f_3 y f_4 , acentuándose más este problema al incrementar el número de variables de decisión. Aunque este deterioro ocurre también con $R = 0.2$; dicho parámetro permite encontrar mejores soluciones.

D	f_1	f_2	f_3	f_4	f_5
10	3.68×10^{-12}	5.83×10^{-13}	8.14×10^{-14}	5.37×10^{-13}	6.18×10^{-13}
30	1.20×10^{-11}	4.62×10^{-12}	4.37×10^{-12}	7.40×10^{-4}	4.44×10^{-12}
50	2.14×10^{-11}	7.71×10^{-12}	1.49×10^0	9.86×10^{-4}	8.02×10^{-12}
100	3.98×10^{-8}	3.78×10^{-5}	3.08×10^1	1.39×10^{-9}	1.58×10^{-11}

Tabla 5.2: Comparación de la aptitud media de 100 ejecuciones independientes con un valor de $R = 0.2$

D	f_1	f_2	f_3	f_4	f_5
10	5.11×10^{-12}	7.93×10^{-13}	2.69×10^0	4.70×10^{-2}	1.36×10^{-12}
30	2.11×10^{-11}	2.00×10^{-11}	1.21×10^1	5.42×10^{-3}	6.12×10^{-12}
50	1.33×10^0	9.95×10^{-12}	2.24×10^1	7.38×10^{-3}	1.47×10^{-11}
100	2.40×10^0	1.52×10^{-11}	5.48×10^1	2.89×10^{-11}	3.26×10^0

Tabla 5.3: Comparación de la aptitud media de 100 ejecuciones independientes con un valor de $R = 0.8$

Con el fin de predecir el comportamiento de cada una de las variantes de Evolución Diferencial durante el proceso de búsqueda, se realizó un estudio sobre el número de veces en el que cada variante producía un *mejor movimiento* al ser aplicada, v.g., contabilizar el

número de veces en el que la heurística seleccionada mejoraba la solución actual. En las figuras 5.3 y 5.4 se muestra el promedio de mejoras de cada variante de ED empleadas como heurísticas de bajo nivel con los valores de R iguales a 0.2 y 0.8, respectivamente y para cada uno de los distintos números de variables de decisión.

A partir del comportamiento observado en las gráficas de las figuras 5.3 y 5.4, se concluye que el parámetro R de la Evolución Diferencial juega un papel muy importante en cada una de las variantes al trabajar conjuntamente para resolver problemas de optimización. Como se puede observar, un valor de $R = 0.2$ tiende a beneficiar a las variantes que usan cruce binomial, teniendo como consecuencia que los modelos con cruce exponencial tengan un desempeño pobre al no producir gran cantidad de buenos movimientos durante el proceso de búsqueda. El caso contrario sucede cuando $R = 0.8$, siendo más evidente cuando el número de variables de decisión tiende a ser mayor, pues estrategias con cruce exponencial producen mayor cantidad de buenos movimientos siendo muy pobre el comportamiento de variantes con cruce binomial.

También podemos notar que para cada conjunto de variantes, tanto con cruce exponencial como binomial, existen ciertas estrategias que sobresalen sobre las demás. Para el caso de las variantes que usan cruce exponencial, **ED/best/1/exp**, **ED/best/2/exp** y **ED/current-to-rand/1/exp** exhiben comportamientos mejores que el resto de los modelos con cruce exponencial. Para el conjunto de variantes con cruce binomial **ED/best/1/bin**, **ED/current-to-best/1/bin** y **ED/current-to-rand/1/bin** muestran el mayor número de mejoras para los modelos con este esquema de recombinación para las funciones de prueba empleadas.

5.1.2. Mecanismo de selección de heurísticas

Con base en los estudios experimentales realizados se detectaron algunas características particulares del papel que juegan algunos parámetros, tanto de la ED como del número de variables de decisión del problema, para el comportamiento de cada uno de los modelos de ED empleados. El modelo propuesto de la hiperheurística desarrollada se basa en la idea de incorporar al mecanismo de selección de heurísticas de bajo nivel las características observadas, de tal manera que se puedan coordinar para ser aplicadas en el mejor momento de acuerdo a las características mismas del problemas como de los parámetros propios de la ED.

De tal modo, los elementos que deben constituir el esquema de selección de heurísticas de la hiperheurística son:

- De acuerdo al valor del parámetro R de la ED, se debe determinar qué tipo de modelo de ED usar, ya sea una variante con cruce exponencial o una con cruce binomial.
- Una vez determinado qué tipo de recombinación se usará, se debe elegir qué modelo en específico se empleará para generar las nuevas soluciones a partir de la población actual.
- Se debe determinar la prioridad del tipo de recombinación que se debe usar de acuerdo al número de variables de decisión del problema.

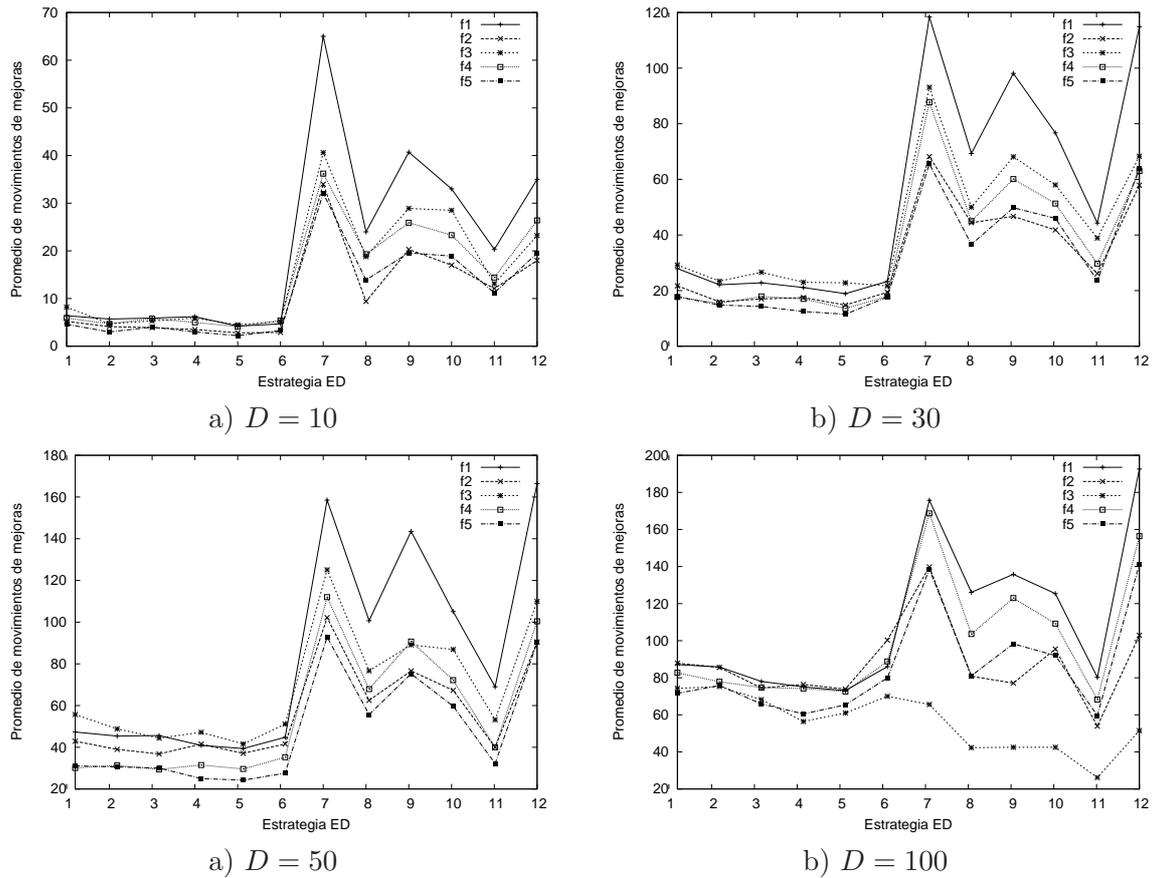


Figura 5.3: Promedio de mejoras por cada variante para: a) $D = 10$, b) $D = 30$, c) $D = 50$ y $D = 100$ variables de decisión con $R = 0.2$ para las cinco funciones de prueba, donde el eje x representa el índice de la variante de la ED de acuerdo a la tabla 5.1, de la pág. 39.

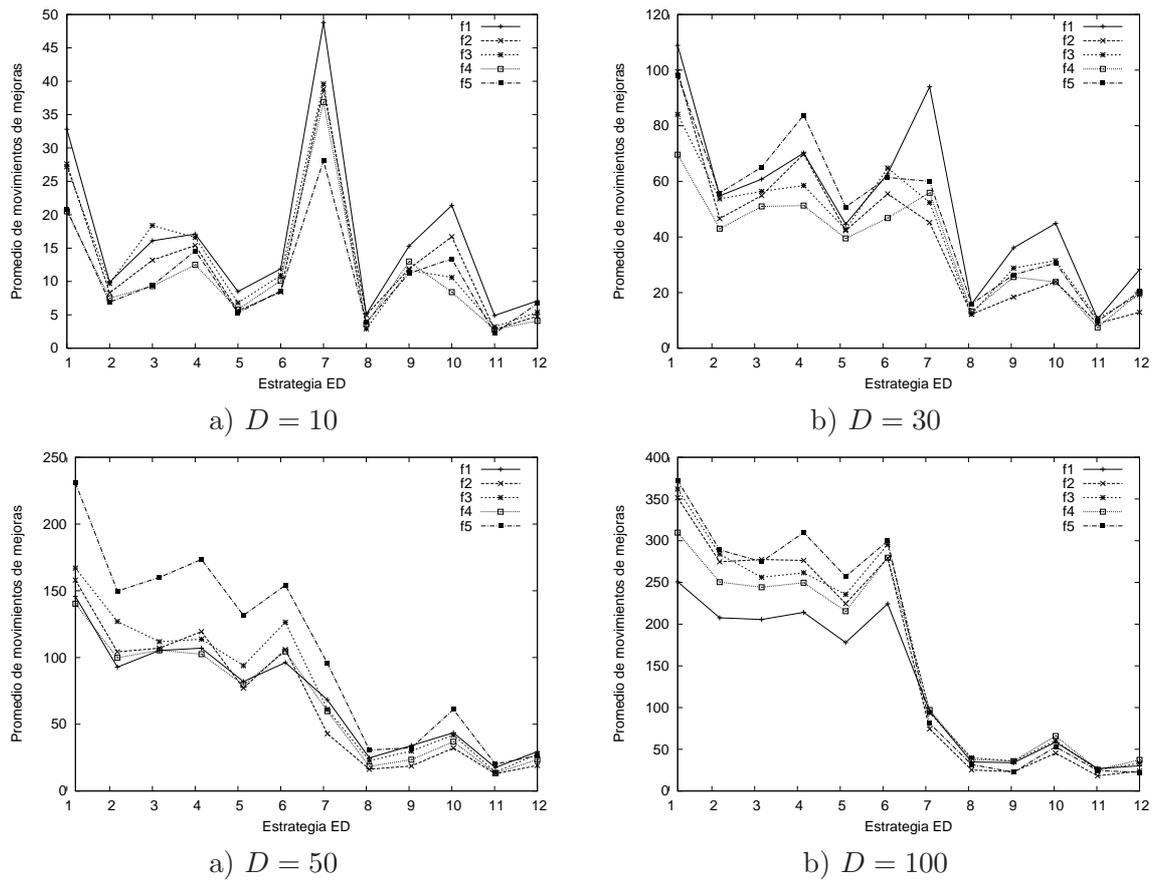


Figura 5.4: Promedio de mejoras por cada variante para: a) $D = 10$, b) $D = 30$, c) $D = 50$ y $D = 100$ variables de decisión con $R = 0.8$ para las cinco funciones de prueba, donde el eje x representa el índice de la variante de la ED de acuerdo a la tabla 5.1, de la pág. 39.

Con base en el listado anterior se diseñaron dos mecanismos de selección que constan de dos fases. En ambos, la primera fase se encarga de elegir el tipo de recombinación a usar (exponencial o binomial). Para ello se varía aleatoriamente en cada generación g el parámetro R en el rango $[0, 1]$ y a partir del valor que adopte, se selecciona el tipo de cruce a usar. Una vez determinado el tipo de recombinación a emplear, la segunda fase consiste en elegir el modelo específico que se aplicará para generar la población $P_{x,g+1}$. El primer mecanismo de selección, denominado **R aleatorio**, elige de manera aleatoria el modelo a usar mientras que en el segundo se hace uso de una ruleta para elegir dicho modelo.

La selección mediante ruleta fue propuesta por DeJong [19] como mecanismo de selección de individuos en los algoritmos genéticos y ha sido uno de los métodos más comúnmente usado debido no sólo a su simplicidad, sino al hecho de que su implementación se incluye en el libro clásico sobre AGs de David Goldberg. El algoritmo 10 muestra el pseudocódigo original de la ruleta propuesto por DeJong [19], donde:

$$T = \sum_{i=1}^N V_i \quad (5.7)$$

y

$$V_i = \frac{f_i(\vec{x})}{\bar{f}(\vec{x})} \quad i = 1, 2, \dots, N \quad (5.8)$$

donde $f_i(\vec{x})$ es la aptitud del individuo i , $\bar{f}(\vec{x})$ es la media de aptitud de la población y NP es el tamaño de la población.

```

1 Calcular la suma de valores esperados  $T$  de todos los individuos
2 for  $i \leftarrow 1, 2, \dots, N$  do
3    $r \leftarrow U(0, T)$ 
4    $sum \leftarrow 0$ 
5    $i \leftarrow 1$ 
6   /* Ciclar a través de los individuos sumando los VEs hasta exceder  $r$  */
7   repeat
8     /* Suma el valor esperado del  $i$ -ésimo individuo */
9      $sum \leftarrow sum + V_i$ 
10     $i \leftarrow i + 1$ 
11  until  $sum \geq r$ ;
12  Seleccionar al individuo que hace exceder el límite

```

Algoritmo 10: Pseudocódigo de la ruleta propuesto por DeJong

Debido a que los individuos más aptos tienen asignado un valor esperado mayor, se espera que sean seleccionados más veces que los menos aptos. Sin embargo, este algoritmo presenta el problema de que el individuo menos apto puede ser seleccionado más de una vez dependiendo de la posición que ocupe en la ruleta.

El uso de la ruleta en nuestra propuesta consiste en contabilizar el número de mejoras que cada variante ha producido durante el proceso de búsqueda. A partir de esta información,

se calculan los valores esperados para, finalmente, aplicar el algoritmo 10. Cabe mencionar que, a diferencia del ciclo que se repite N veces, sólo requerimos aplicarlo una sola vez ya que únicamente se necesita seleccionar una heurística. Además, al inicio del proceso de búsqueda no contamos con información alguna, para lo cual requerimos de una primera etapa de *entrenamiento*, la cual, consiste en la ejecución de un número máximo de generaciones en las que usamos el mecanismo de selección **aleatorio descendente** propuesto en [16] para inicializar los valores esperados (VEs) de cada una de las variantes de acuerdo a la siguiente ecuación:

$$V_i = \frac{\text{éxitos}_i}{\frac{1}{12} \sum_j^{12} \text{éxitos}_j} \quad i = 1, 2, \dots, 12 \quad (5.9)$$

Se implementaron tres tipos de ruletas. A continuación se detallan cada una de ellas:

- **Ruleta original:** corresponde al algoritmo original propuesto en [19].
- **Ruleta con inicio aleatorio:** la diferencia con el algoritmo original consiste en seleccionar la posición inicial de manera aleatoria. Si se llega a la última posición sin exceder r , la siguiente posición corresponde a la primera posición de la ruleta.
- **Ruleta con permutación:** para este enfoque se crea una permutación de las posiciones de la ruleta para posteriormente aplicar el algoritmo original.

Las dos últimas versiones se implementaron con el propósito de variar el recorrido de las posiciones de la ruleta, a fin de mitigar el problema del algoritmo original de seleccionar al peor individuo en varias ocasiones. En la figura 5.5 se muestra un ejemplo de cada uno de los tres enfoques de selección mediante ruleta, en el que cada heurística de bajo nivel h_i tiene asociado un porcentaje de probabilidad de ser seleccionada, en donde i corresponde a la posición de inicio y U es un número aleatorio en la amplitud $[1, 4]$ que es el total de posiciones de la ruleta para este ejemplo en particular.

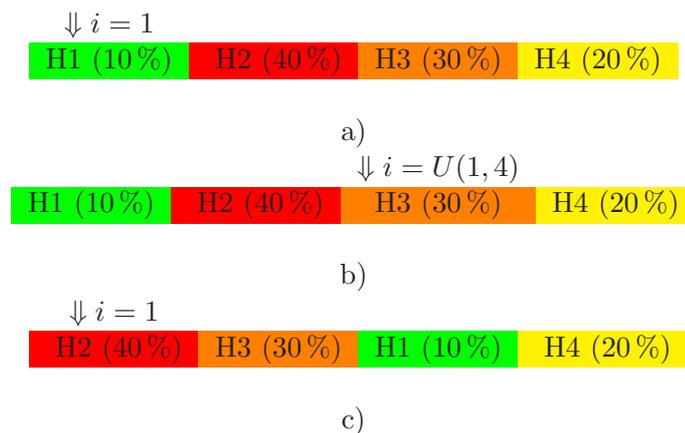


Figura 5.5: Ejemplo hipotético de los tres tipos de ruletas con cuatro heurísticas de bajo nivel: a) Ruleta original, b) Ruleta con inicio aleatorio y c) Ruleta con permutación

El algoritmo 11 muestra el pseudocódigo de la descripción de la propuesta del mecanismo de selección de la hiperheurística para optimización global sin restricciones, donde

mecanismo se refiere al tipo de selección a usar y R_{sel} es un parámetro que indica la probabilidad de seleccionar modelos con cruza binomial o exponencial. El parámetro R_{sel} es calculado con base en la ecuación (5.10), con el propósito de que el porcentaje de selección de modelos con cruza exponencial o binomial se adapte de acuerdo a las características del problema y haciendo innecesario que este parámetro sea proporcionado por el usuario.

$$R_{sel} = \frac{\text{Total de éxitos de variantes con cruza binomial}}{\text{Total de éxitos de todas las variantes}}$$

$$R_{sel} = \frac{\sum_{i=7}^{12} \text{éxitos}_i}{\sum_{i=1}^6 \text{éxitos}_i + \sum_{i=7}^{12} \text{éxitos}_i} \quad (5.10)$$

```

Input: mecanismo
Output: heurística seleccionada
1 switch mecanismo do
2   | case ALEATORIO
3   |   | heuristica_sel ← rand(h1, h12)
4   | case R_ALEATORIO
5   |   | R ← U(0, 1)
6   |   | if R > R_sel then
7   |   |   | /* Estrategias con cruza exponencial */
8   |   |   | heuristica_sel ← rand(h1, h6)
9   |   |   | else
10  |   |   |   | /* Estrategias con cruza binomial */
11  |   |   |   | heuristica_sel ← rand(h7, h12)
12  |   | case RULETA
13  |   |   | R ← U(0, 1)
14  |   |   | if R > R_sel then
15  |   |   |   | Seleccionar estrategias con cruza exponencial mediante ruleta
16  |   |   |   | else
17  |   |   |   |   | Seleccionar estrategias con cruza binomial mediante ruleta

```

Algoritmo 11: Mecanismo de selección de la hiperheurística propuesta

Finalmente, en el algoritmo 12 se muestra el pseudocódigo de la hiperheurística propuesta. Los parámetros de control del algoritmo propuesto son los siguientes:

- $G_{m\acute{a}x}$: número máximo de generaciones.
- N : número de individuos que componen la población.
- R : parámetro de control de la cruza discreta usada por la Evolución Diferencial.
- *mecanismo*: tipo de selección de las heurísticas de bajo nivel a usar.

```

Input:  $N, G_{max}, R, \text{mecanismo}$ 
1  $G \leftarrow 0$ 
2 Inicializar  $P_{x,G}$ 
3 if mecanismo es CR_ALEATORIO o RULETA then
    /* Etapa de entranamiento */
4  $G_{aux} \leftarrow 0$ 
5 while  $G_{aux} < G_{prueba}$  do
    /* Aplicar mecanismo aleatorio */
6  $\text{mecanismo\_seleccion}()$ 
7 repeat
8      $\text{aplicar\_heuristica}(\text{heuristica\_sel})$ 
9      $G \leftarrow G + 1$ 
10     $G_{aux} \leftarrow G_{aux} + 1$ 
11 until no mejora la solución previa ;
12 Inicializar el valor esperado de cada heurística de bajo nivel
13 while Criterio de terminación no satisfecho do
14  $\text{mecanismo\_seleccion}()$ 
15 if mecanismo es DESCENDENTE then
16     repeat
17          $\text{aplicar\_heuristica}(\text{heuristica\_sel})$ 
18          $G \leftarrow G + 1$ 
19     until no mejora la solución previa ;
20 else
21      $\text{aplicar\_heuristica}(\text{heuristica\_sel})$ 
22      $G \leftarrow G + 1$ 

```

Algoritmo 12: Propuesta de hiperheurística basada en variantes de la ED

Con el objetivo de comprobar cada uno de los mecanismos propuestos, en el Apéndice B se muestran los resultados experimentales en los problemas de prueba sin restricciones, haciendo una comparativa con el modelo original **ED/rand/1/bin** el cual, de acuerdo a [74], es la variante de la ED que presenta mejores rendimientos.

Entre los diversos tipos de mecanismos de selección de heurísticas de bajo nivel se compara el **aleatorio descendente** propuesto por Cowling en [16], y los dos esquemas propuestos en esta tesis: variando el parámetro R y haciendo uso de las ruletas en sus tres tipos, empleando el mismo conjunto de funciones de prueba para optimización global sin restricciones.

En dicho estudio se hace una comparativa entre los diversos mecanismos usando tanto el promedio del resultado de la función objetivo como del número de generaciones requeridas en las que cada uno de los algoritmos convergió. Para el comparativo se usaron los cuatro valores para el número de variables de decisión D . En el caso del mecanismo de selección aleatorio descendente se muestra el resultado para dos valores del parámetro de control de cruza R de la ED, 0.2 y 0.8, dejando fijo éste con $R = 0.8$ para el modelo **ED/rand/1/bin** que fue con el que se obtuvieron los mejores resultados.

De las tablas del Apéndice B, en la pág. 79, se observa que para las funciones f_2 y f_3 conforme se aumenta la dimensionalidad del problema, el modelo **ED/rand/1/bin** tiende a mostrar rendimientos muy pobres, además de que en todo el conjunto de funciones de prueba se requiere de un mayor número de generaciones para hallar una solución para cada caso de estudio en comparación con los que usan un enfoque hiperheurístico.

La función f_3 resulta ser la más difícil de resolver a partir de $D = 30$ tanto para el modelo **ED/rand/1/bin** como para los mecanismos de selección **aleatorio descendente** y variando la constante de recombinación R . Por otro lado, los mecanismos que hacen uso de las ruletas ofrecen resultados competitivos para esta función.

5.2. Manejo de restricciones

Como se ha mencionado en el capítulo 4, en la pág. 27, el método más ampliamente usado para el manejo de restricciones, incluyendo a las propuestas basadas en la ED, son las funciones de penalización.

Como se indicó, el principal problema de las técnicas para el manejo de restricciones es lograr el balance correcto entre tener prioridad en minimizar la función objetivo o en minimizar el grado de violación de las restricciones. La Jerarquización Estocástica (JE) [63] ha sido un método de manejo de restricciones muy exitoso, debido a que trata de balancear la influencia de considerar la función objetivo o el grado de violación de las restricciones mediante un ordenamiento de la población, por lo que se decidió hacer uso de este mecanismo para incorporarlo a nuestra propuesta.

En [25] se demostró que un mecanismo de JE embebido a la ED puede proveer información sobre la dirección idónea al operador de mutación acelerando la velocidad de búsqueda. En la figura 5.6 se muestran dos ejemplos hipotéticos para las direcciones de búsqueda considerando: a) el movimiento de un punto infactible a uno factible y b) el movimiento de un punto factible a uno infactible, guiando de esta manera la generación de nuevas soluciones más cerca o más lejos de la región factible \mathcal{F} , respectivamente.

En [63], al jerarquizar λ posibles soluciones, se propone el uso de una probabilidad P_f que va a determinar el balance entre tener prioridad con respecto al valor de la función objetivo o al grado de violación de las restricciones. De tal suerte, que una vez aplicada la jerarquización a la población actual la mayor cantidad de buenos individuos estarán jerarquizados en las primeras posiciones de la población.

La forma de elegir los vectores \vec{x}_{r_1} y \vec{x}_{r_2} permite guiar la búsqueda a cualquiera de ambas direcciones de búsqueda, mientras que la elección del tercer vector \vec{x}_{r_3} será con base en el modelo de la ED a usar. Como se mencionó anteriormente, la Jerarquización Estocástica impondrá un orden de los individuos. A partir de la jerarquía de cada individuo se puede guiar el proceso de búsqueda a una cierta dirección específica. En la figura 5.7 se muestra el diagrama de flujo desde la aplicación de la JE hasta la elección de los vectores para generar el vector de mutación $\vec{v}_{i,g}$, esto para el caso particular en el que se desea guiar la búsqueda

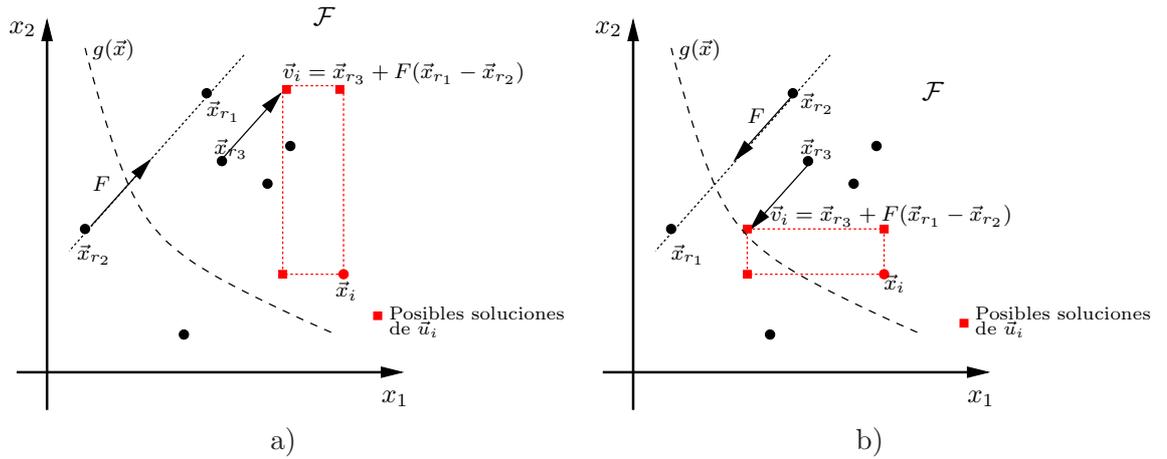


Figura 5.6: Ejemplificación de los dos tipos de dirección de búsqueda del operador de mutación de la ED: a) hacia la zona factible \mathcal{F} y b) hacia la zona infactible

hacia la región factible. En caso de buscar guiar la dirección de búsqueda hacia la región infactible, \vec{x}_{r1} deberá ser seleccionado del grupo 2 y \vec{x}_{r2} del grupo 1.

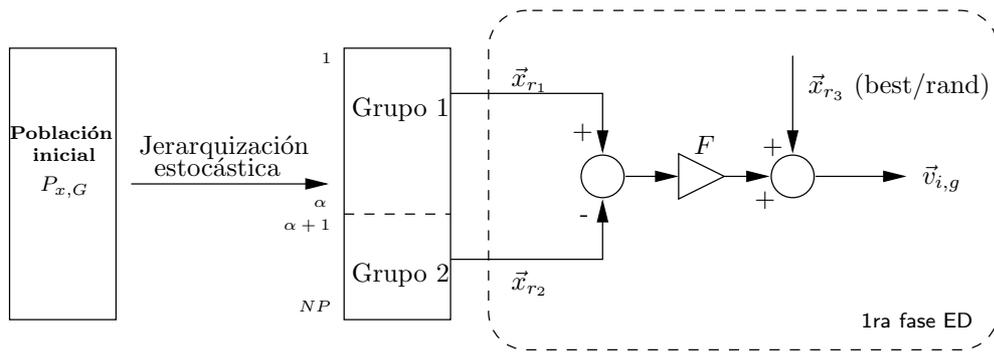


Figura 5.7: Diagrama de flujo del proceso de división de la población

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

$$\vec{x}_{i,G+1} = \begin{cases} \vec{u}_{i,G+1} & \text{si } f(\vec{u}_{i,G+1}) < f(\vec{x}_{i,G}) \\ \vec{x}_{i,G} & \text{en otro caso} \end{cases} \quad (5.11)$$

Por lo anterior, es necesario modificar el operador de selección de la ED utilizando el siguiente criterio:

- Si ambas soluciones son factibles se elige la que tenga un mejor valor en la función objetivo.
- De lo contrario, se elige a la que tenga el menor grado de violación de restricciones.

Finalmente, para poder escoger la dirección de búsqueda se hace uso de un elemento probabilístico con el fin de buscar guiar la búsqueda tanto hacia adentro de la región factible como hacia afuera de ella con el propósito de explorar los límites entre ambas regiones del espacio de búsqueda. Dicho valor se demostró experimentalmente que ofrecía buenos resultados al emplear el mismo valor P_f de la jerarquización estocástica.

El algoritmo 13 muestra el proceso que se lleva a cabo para generar, a partir de la población actual $P_{x,G}$, la nueva población $P_{x,G+1}$. La función objetivo es denotada por f y el grado de violación de las restricciones es denotada por ϕ .

```

Input: estrategia de ED a usar
Result: generación  $G + 1$ 
1 begin
2   Aplicar la jerarquización estocástica para jerarquizar la población  $P_{x,G}$ 
3   for  $i \leftarrow \{1, \dots, N\}$  do
4     /* Selección de vectores para la perturbación */
5     if  $U(0,1) < P_f$  then
6       /* Dirección hacia la región infactible */
7        $r_1 \leftarrow \text{rand}(1, \alpha)$ 
8        $r_2 \leftarrow \text{rand}(\alpha + 1, N)$ 
9     else
10      /* Dirección hacia la región factible */
11       $r_1 \leftarrow \text{rand}(\alpha + 1, N)$ 
12       $r_2 \leftarrow \text{rand}(1, \alpha)$ 
13     Seleccionar el vector  $r_3$  de acuerdo a la estrategia de ED a usar (best ó rand)
14     /* Mutación */
15      $\vec{v}_{k,G} = \vec{x}_{k,G,r3} + F(\vec{x}_{k,G,r1} - \vec{x}_{k,G,r2})$ 
16     /* Recombinación */
17     Aplicar el operador de recombinación de acuerdo a la estrategia de ED para
18     obtener  $\vec{u}_i$ 
19     /* Selección */
20     if  $\phi(\vec{u}_i) = \phi(\vec{x}_{i,G}) = 0$  then
21       if  $f(\vec{u}_i) < f(\vec{x}_{i,G})$  then
22          $\vec{x}_{i,G+1} \leftarrow \vec{u}_i$ 
23       else
24          $\vec{x}_{i,G+1} \leftarrow \vec{x}_{i,G}$ 
25     else
26       if  $\phi(\vec{u}_i) < \phi(\vec{x}_{i,G})$  then
27          $\vec{x}_{i,G+1} \leftarrow \vec{u}_i$ 
28       else
29          $\vec{x}_{i,G+1} \leftarrow \vec{x}_{i,G}$ 
30   end

```

Algoritmo 13: Pseudocódigo del algoritmo de la ED para espacios restringidos

En la primera parte de este capítulo y, durante el diseño del mecanismo de selección de la hiperheurística para problemas sin restricciones, se llevaron a cabo una serie de experimentos para validar dicha propuesta sin restricciones y que permitió identificar algunas características importantes que sirvieron de base para el diseño del esquema de selección de la hiperheurística, para finalmente incorporar un manejo de restricciones a dicha propuesta.

En el siguiente capítulo se presenta el diseño experimental para validar la hiperheurística propuesta en esta tesis para resolver problemas de optimización con restricciones. En este diseño experimental se incluye un análisis estadístico tradicional como el uso de otras herramientas estadísticas para observar el comportamiento del algoritmo en diversas funciones utilizadas en la literatura especializada y las cuales serán detalladas en el siguiente capítulo.

Resultados Experimentales

En este capítulo se detallarán los resultados obtenidos de la hiperheurística en problemas representativos de la literatura especializada para optimización global para espacios restringidos. Para validar nuestra propuesta se utilizaron trece funciones de prueba estándar tomadas de la literatura de optimización global con restricciones las cuales se listan en el Apéndice C, en la pág. 83.

En la primera parte de este capítulo se dará una breve descripción del diseño experimental llevado a cabo, presentando una introducción de las herramientas estadísticas empleadas. Finalmente, se realiza un estudio comparativo de nuestra propuesta contra el algoritmo **CDE** [46] y el enfoque hiperheurístico con mecanismo de selección aleatorio descendente propuesto en [16].

6.1. Diseño experimental

Con la finalidad de evaluar el desempeño del mecanismo de selección de la hiperheurística implementada, se llevó a cabo un análisis estadístico usando un conjunto de trece funciones de prueba estándar de optimización global con restricciones, las cuales se detallan en el Apéndice C de este trabajo.

El estudio consistió en realizar 100 ejecuciones independientes con diferente valor de semilla para la generación de números aleatorios R_{seed} para cada una de las trece funciones para optimización en espacios restringidos. Una vez realizadas las ejecuciones para una determinada función, se obtuvo la muestra original $\vec{\theta}$ de tamaño $n = 100$ que conforman cada una de las soluciones obtenidas de las 100 ejecuciones independientes.

Posteriormente, al haber generado la muestra $\vec{\theta}$ se aplica un análisis estadístico para cada una de las funciones. Dicho análisis contempla la siguiente información:

- Análisis estadístico tradicional.
- Gráficos de caja.
- Intervalos de confianza por el método de *bootstrap* [23].

A continuación se describirá brevemente cada uno de ellos.

6.1.1. Análisis estadístico

Las medidas de tendencia central sirven como puntos de referencia para interpretar un conjunto de observaciones de una prueba, cuyo propósito es mostrar en qué lugar se ubica el promedio del conjunto de muestras y permitir comparar o interpretar cualquier valor en relación con el valor central.

Sean $[x_1, x_2, \dots, x_n]^T$, los datos de una muestra $\vec{\theta}$. La **mediana** se define como el valor central de los datos observados una vez que éstos han sido ordenados. La ecuación (6.1) muestra la manera de calcular dicho valor.

$$M = \begin{cases} x_{\frac{n+1}{2}} & \text{si } n \text{ es impar} \\ \frac{1}{2} (x_{\frac{n}{2}} + x_{\frac{n}{2}+1}) & \text{si } n \text{ es par} \end{cases} \quad (6.1)$$

La **media aritmética** o **promedio** de un conjunto de datos es la cantidad total de la variable distribuida en partes iguales, y es igual a la suma de todos ellos dividida entre el número de elementos que conforman la observación. La ecuación (6.2) muestra la manera en que se calcula la media aritmética.

$$\mu = \frac{1}{n} \sum_i^n x_i \quad (6.2)$$

La **desviación estándar** es una medida de dispersión que indica cuánto tienden a alejarse los valores del promedio en una distribución. Una desviación estándar grande indica que los puntos están lejos de la media, y una desviación pequeña indica que dichos datos están agrupados cerca de la media. En la ecuación (6.3) se muestra la definición matemática para la desviación estándar de un conjunto de datos.

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2} \quad (6.3)$$

6.1.2. Gráficos de caja

Los gráficos de caja (*Boxplot*) o cajas de Tukey son una herramienta gráfica propuesta originalmente por John Tukey [73]. Constan de cinco parámetros y permiten observar de una forma clara la distribución de los datos y sus principales características así como comparar diversos conjuntos de datos simultáneamente.

Como herramienta visual se pueden utilizar para ilustrar los datos, estudiar su simetría y supuestos sobre la distribución así como para comparar diferentes poblaciones.

El gráfico consiste de un rectángulo, usualmente orientado con el sistema de coordenadas, tal que el eje vertical tiene la misma escala del conjunto de datos. La parte superior e inferior del rectángulo coinciden con el tercer (Q_3) y primer cuartil¹ (Q_1), respectiva-

¹Valor que divide a la distribución ordenada en cuatro partes iguales. Es decir, el primer cuartil es el valor por debajo del cual ocurre el 25% de las observaciones. El segundo cuartil corresponde a la mediana y el tercer cuartil es el valor por debajo del cual ocurre el 75% de las observaciones.

mente. Este rectángulo o caja se divide con una línea horizontal a nivel de la mediana (Q_2). Se define un *paso* igual a 1.5 veces el rango intercuartil ($Q_3 - Q_1$), y una línea vertical, llamada **bigote** que se extiende desde la mitad de la parte superior de la caja hasta la mayor observación de los datos si se encuentran dentro del límite del *paso*. Este proceso se hace de igual manera en la parte inferior de la caja. Las observaciones que caigan más allá de estas líneas son dibujadas individualmente, y se les considera como **puntos atípicos** (*outliers*).

En la figura 6.1 se muestra el digrama general de un gráfico de caja indicando los principales componentes que lo conforman, donde el bigote superior L_s e inferior L_i están dados por:

$$L_s = Q_3 + 1.5 \cdot (Q_3 - Q_1) \quad (6.4)$$

$$L_i = Q_1 - 1.5 \cdot (Q_3 - Q_1) \quad (6.5)$$

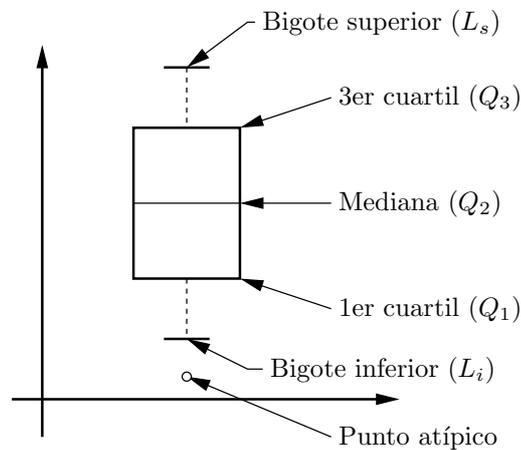


Figura 6.1: Diagrama de caja indicando sus principales componentes

La forma de la gráfica permite interpretar la distribución de los datos; mientras más larga sea la caja, más dispersa será la distribución de éstos. La mediana permite identificar la simetría; si está relativamente en el centro, la distribución es simétrica. Si por el contrario, se acerca al primer cuartil, la distribución es *asimétrica positiva*. Si se aproxima al tercer cuartil, entonces es *asimétrica negativa*; esto sucede cuando los datos tienden a concentrarse más hacia un valor en específico.

Existen muchas variaciones de este gráfico, las cuales expresan otras características de los datos que en un momento dado puedan ser de interés. Por ejemplo, a veces se suelen dibujar marcas para ubicar la media aritmética mientras que otros deforman la caja para obtener más claridad acerca de la distribución [6, 78].

6.1.3. Métodos estadísticos de cómputo intensivo

Existe una clase de métodos basados en simulación del proceso de muestreo, los cuales son llamados *métodos estadísticos de cómputo intensivo* dentro de los cuales se encuentran

las pruebas de permutación, métodos de Monte Carlo, métodos de *bootstrap*, etc. [52]. Entre las ventajas de estos métodos se listan las siguientes:

- **Menores suposiciones:** no se requiere que las muestras se ajusten a una distribución normal o que las muestras sean de gran tamaño.
- **Mayor precisión:** las pruebas de permutación y algunos métodos de *bootstrap* son más precisos en la práctica que los métodos clásicos.
- **Generalidad:** los métodos de remuestreo son muy similares para un amplio rango de estadísticas, y no requieren una fórmula para cada una de ellas.

El análisis de datos tradicional, se fundamenta en diferentes suposiciones, p.ej., los datos son independientes, los datos se ajustan a una forma de distribución específica, etc. En algunas ocasiones estas suposiciones están justificadas, pero, en los casos donde tales suposiciones no se cumplen, su relevancia es cuestionable.

Método de *bootstrap*

Los métodos de *bootstrap* son procedimientos de simulación del proceso de muestreo desarrollados por Efron [23] y han sido ampliamente usados en la determinación de estimadores estadísticos, en la construcción de intervalos de confianza, en el cómputo de probabilidades de una prueba de hipótesis, etc. El *bootstrap* es un método estadístico creado para facilitar los cálculos que no se pueden hacer con fórmulas simples por medio de las técnicas estadísticas clásicas, teniendo como herramienta importante la ayuda de una computadora.

La idea general de los métodos de *bootstrap* consiste en generar una distribución muestral para una estadística dada, empleando un remuestreo de Monte Carlo y considerando a la distribución de la muestra como la distribución de la población misma.

El método de *bootstrap* trabaja de la siguiente manera:

- Se tiene una muestra aleatoria (producida mediante reemplazo) de tamaño n , en donde $\vec{x} = [x_1, x_2, \dots, x_n]^T$ son los valores observados de dicha muestra.
- Se crea una nueva muestra del mismo tamaño muestreando aleatoriamente n veces con reemplazo a la muestra original \vec{x} , donde la probabilidad de escoger cualquier punto es uniforme.
- Luego se calcula el estadístico de interés θ para cada una de las muestras Bootstrap a partir de la remuestra obtenida, obteniéndose así Θ_b^* , donde $b = 1, 2, \dots, B$ y B es un número grande que representa la cantidad de remuestras hechas.

La magnitud de B en la práctica depende de las pruebas que se van aplicar a los datos. En general, B debería ser de entre 50 a 200 para estimar el error estándar, y al menos de 1000 para estimar los intervalos de confianza. La distribución resultante es llamada **Distribución Muestral de *Bootstrap*** (DMB) y puede ser calculada para cualquier estadística; el único requerimiento es que la muestra sea representativa de la población.

El algoritmo 14 muestra el esquema general de un procedimiento de *bootstrap* con B remuestreos, empleando una muestra θ de tamaño n , con la cual, se busca una DMB para un estimador Θ^* .

```

1 for  $i \leftarrow \{1, 2, \dots, B\}$  do
2   | Generar una remuestra  $\vec{\theta}_i$  de la muestra original  $\vec{\theta}$  con reemplazo, igual
   | probabilidad y mismo tamaño
3   | Calcular y guardar el valor  $\Theta_i^*$  de la estadística de  $\vec{\theta}_i$ 
4 Los valores  $\Theta_1^*, \Theta_2^*, \dots, \Theta_B^*$  componen a la distribución muestral de bootstrap

```

Algoritmo 14: Esquema general de un procedimiento de *bootstrap*

Intervalos de confianza

En la estimación de los parámetros de una población desconocida a partir de una muestra, es deseable poder caracterizar la precisión de las estimaciones de los parámetros. Los intervalos de confianza (IC) proveen una herramienta para esta situación.

Cuando las muestras no se ajustan a una distribución normal, el procedimiento de *bootstrap* nos permite calcular los intervalos de confianza de diversas formas. Es posible que la muestra original no se ajuste a una distribución normal, sin embargo, la distribución muestral de *bootstrap* si podría ajustarse a ella. Existen tres métodos a través de los cuales se pueden construir intervalos de confianza *bootstrap*: **método de aproximación normal**, **métodos de los percentiles** y **método de los percentiles corregidos**.

El primero de ellos utiliza la misma estructura de los procedimientos paramétricos, siempre y cuando sea posible asumir que el estadístico se distribuye según la curva normal pero el cálculo resulta analíticamente difícil o no existe fórmula para dicho cálculo. Es entonces cuando se puede emplear la distribución muestral de *bootstrap* para estimar el intervalo de confianza.

El método del percentil² hace uso de la idea básica del *bootstrap*. La idea es muy simple: un intervalo con un nivel de confianza $1 - \alpha$ incluye todos los valores de Θ^* entre los percentiles $\frac{\alpha}{2}$ y $1 - \frac{\alpha}{2}$.

El tercer método es similar al procedimiento anterior; lo único que cambia es el modo de calcular los percentiles para obtener el intervalo. Efron y Tibshirani [24] explican en detalle cómo se calculan los percentiles corregidos, siendo éste el método más adecuado ya que corrige la asimetría que pudiera presentar la distribución muestral del estadístico.

6.2. Resultados experimentales

Para estimar la dificultad de generar una solución factible para cada uno de los problemas de estudio se emplea la métrica ρ propuesta por Michalewicz y Schoenauer [50]. Esta

²Representa los valores de la variable que están por debajo de un porcentaje, el cual puede ser un valor de 1% a 100%. En otras palabras, el total de los datos es dividido en 100 partes iguales.

métrica pretende estimar la porción de la zona factible \mathcal{F} con respecto a todo el espacio de búsqueda \mathcal{S} , así $\rho = \frac{|\mathcal{F}|}{|\mathcal{S}|}$, donde $|\mathcal{F}|$ y $|\mathcal{S}|$ son aproximaciones al tamaño de la zona factible y al tamaño del espacio de búsqueda, respectivamente. Para calcular ρ se generan $|\mathcal{S}| = 1 \times 10^6$ soluciones uniformemente distribuidas en todo el espacio de búsqueda y se determina, de entre ellas, el número de soluciones factibles $|\mathcal{F}|$.

En la tabla 6.1 se listan algunas de las características de los problemas de prueba con restricciones entre las que se observan el número de variables de decisión D , el tipo de función, los valores de ρ así como el número de restricciones de desigualdad $g(\vec{x})$ e igualdad $h(\vec{x})$. De la tabla se observa cómo los problemas ofrecen una gama amplia de características.

	D	mín / máx	Tipo de función	ρ	No. $g(\vec{x})$	No. $h(\vec{x})$
g01	13	mín	cuadrática	0.0111 %	9	0
g02	20	máx	no lineal	99.9971 %	2	0
g03	10	máx	polinomial	0.0026 %	0	1
g04	5	mín	cuadrática	52.1230 %	6	0
g05	4	mín	cúbica	0.0000 %	2	3
g06	2	mín	cúbica	0.0066 %	2	0
g07	10	mín	cuadrática	0.0003 %	8	0
g08	2	mín	no lineal	0.8560 %	2	0
g09	7	mín	polinomial	0.5121 %	4	0
g10	8	mín	lineal	0.0010 %	6	0
g11	2	mín	cuadrática	0.0973 %	0	1
g12	3	máx	cuadrática	4.7713 %	1	0
g13	5	mín	no lineal	0.0000 %	0	3

Tabla 6.1: Características de las 13 funciones utilizadas para optimización en espacios restringidos

6.2.1. Parámetros de control

Como se discutió en la sección 5.1.2, nuestra propuesta de hiperheurística utiliza los siguientes parámetros: factor de amplificación F , control de cruza R , tamaño de la población N , el número de generaciones máximas $G_{\text{máx}}$ y finalmente la probabilidad P_f de comparar con respecto al valor de la función objetivo para el manejo de restricciones.

Como se explicó anteriormente, el parámetro F se genera de manera aleatoria en la amplitud $[0.3, 0.9]$ en cada generación. Con respecto al tamaño de la población, se observó un buen comportamiento del algoritmo al utilizar una población de 50 individuos. El número máximo de iteraciones se fijó de tal forma que el algoritmo realizara a lo más 300,000 evaluaciones de la función objetivo; para ello se usó $G_{\text{máx}} = 6000$ generaciones.

Finalmente, el parámetro P_f fue determinado experimentalmente realizando una búsqueda exhaustiva. En [63] los autores reportan que se logran los mejores resultados cuando $0.4 \leq P_f \leq 0.5$; en las pruebas realizadas se pudo constatar que dicho intervalo resultó ser válido. En la tabla 6.2 se muestra un resumen para los parámetros de control empleados para la hiperheurística, siendo éstos los mismos para cada uno de los mecanismos que forman parte de este estudio.

Parámetro	Valor
G_{\max}	6000
N	50
F	$U(0.3, 0.9)$ en cada generación
P_f	0.500 para g06 y g11 0.450 para las funciones restantes

Tabla 6.2: Parámetros de control empleados en las pruebas experimentales para la hiperheurística

6.2.2. Resultados obtenidos

La nueva propuesta de hiperheurística se comparó con el algoritmo **CDE** (*Constrained Differential Evolution*) propuesto por Lampinen [46], el cual representa a la mejor propuesta para optimización restringida empleando Evolución Diferencial, cuyos parámetros de control empleados se muestran en la tabla 6.3. También se hace una comparativa con la hiperheurística con mecanismo de selección **aleatorio descendente** propuesto por Cowling [16], siendo los mismos parámetros de control empleados los mostrados en la tabla 6.2.

Parámetro	Valor
G_{\max}	6000
N	50
F	0.9
R	0.8

Tabla 6.3: Parámetros de control empleados en las pruebas experimentales para el algoritmo CDE

Los resultados se agrupan de la siguiente manera:

- En las tablas 6.4, 6.5 y 6.6 se muestran los resultados obtenidos, tomando en cuenta la solución obtenida de $f(\vec{x})$ a las 100 ejecuciones independientes efectuados con los algoritmos antes mencionados; donde **R1** corresponde a la ruleta original, **R2** es la ruleta con inicio aleatorio y **R3** la ruleta con permutación. Un valor en **negritas** indica el mejor resultado obtenido por algún algoritmo.
- En las tablas 6.7, 6.8 y 6.9 se muestran los resultados de la mejor y peor solución encontrada para cada una de las funciones y algoritmos. Además se presenta el porcentaje de éxitos con respecto a las 100 ejecuciones independientes en el que los algoritmos lograron obtener la solución óptima para cada una de las funciones de prueba.

De acuerdo a los resultados presentados en las tablas 6.7, 6.8 y 6.9, se puede observar que el mecanismo de selección de la ruleta con inicio aleatorio (**R2**) obtiene buenos resultados en la mayoría de las funciones de prueba en comparación con los otros algoritmos, obteniendo un porcentaje de éxitos elevado. Con respecto al algoritmo **CDE** podemos notar que muestra resultados muy pobres en problemas como g03, g05, g10 y g13.

	Óptimo	CDE			A. Descendente		
		μ	σ	M	μ	σ	M
g01	-15.000	-13.93461	1.23×10^0	-14.99999	-11.25425	2.64×10^0	-12.000
g02	-0.803619	-0.8033878	6.83×10^{-5}	-0.8033884	-0.784641	3.27×10^{-2}	-0.7930787
g03	-1.000	0.24616	9.50×10^{-2}	-1.000	-1.000	5.13×10^{-9}	-1.000
g04	-30665.539	-30665.539	0.00×10^0	-30665.539	-30665.539	0.00×10^0	-30665.539
g05	5126.498	5315.60	3.01×10^2	5126.498	5195.899	2.20×10^2	5126.500
g06	-6961.814	-6961.814	0.00×10^0	-6961.814	-7229.388	1.41×10^3	-7818.326
g07	24.3062091	24.78596	3.12×10^{-1}	24.3062091	24.31361	6.98×10^{-2}	24.30623
g08	-0.095825	-0.09583	1.11×10^{-3}	-0.095826	-0.095825	0.00×10^0	-0.095825
g09	680.6301	680.6301	0.00×10^0	680.6301	680.6301	0.00×10^0	680.6301
g10	7049.250	7090.50762	3.99×10^2	7085.876	7094.163	8.22×10^1	7049.396
g11	0.750	0.750	0.00×10^0	0.750	0.9505711	9.51×10^{-2}	1.000
g12	-1.000	-1.000	0.00×10^0	-1.000	-1.000	0.00×10^0	-1.000
g13	0.053950	0.80852	1.87×10^{-1}	0.2476	0.3678503	1.64×10^{-1}	0.4394295

Tabla 6.4: Estadísticas con respecto a la solución obtenida $f(\vec{x})$ para CDE y mecanismo aleatorio descendente

	Óptimo	R variable			R1		
		μ	σ	M	μ	σ	M
g01	-15.000	-11.82917	2.72×10^0	-12.000	-14.81745	2.72×10^0	-15.000
g02	-0.803619	-0.8009026	6.16×10^{-3}	-0.8036145	-0.8015735	4.63×10^{-3}	-0.803613
g03	-1.000	-0.9999999	3.09×10^{-8}	-0.9999999	-0.9999997	7.55×10^{-7}	-0.9999999
g04	-30665.539	-30665.539	0.00×10^0	-30665.539	-30665.539	0.00×10^0	-30665.539
g05	5126.498	5171.077	1.42×10^2	5126.498	5170.813	1.42×10^2	5126.498
g06	-6961.814	-6961.814	0.00×10^0	-6961.814	-6961.814	0.00×10^0	-6961.814
g07	24.3062091	24.30722	7.70×10^{-3}	24.306210	24.30708	3.42×10^{-4}	24.30705
g08	-0.095825	-0.095825	0.00×10^0	-0.095825	-0.095825	0.00×10^0	-0.095825
g09	680.6301	680.6301	0.00×10^0	680.6301	680.6301	0.00×10^0	680.6301
g10	7049.250	7103.163	8.81×10^1	7049.63	7104.965	8.83×10^1	7049.783
g11	0.750	0.8992142	1.19×10^{-1}	1.000	0.901635	1.18×10^{-1}	1.000
g12	-1.000	-1.000	0.00×10^0	-1.000	-1.000	0.00×10^0	-1.000
g13	0.053950	0.3227286	2.22×10^{-1}	0.4388694	0.3071761	1.98×10^{-1}	0.4388515

Tabla 6.5: Estadísticas con respecto a la solución obtenida $f(\vec{x})$ para R variable y R1

	Óptimo	R2			R3		
		μ	σ	M	μ	σ	M
g01	-15.000	-15.000	0.00×10^0	-15.000	-14.8200	2.72×10^0	-15.000
g02	-0.803619	-0.8014436	4.67×10^{-3}	-0.8014834	-0.8014504	5.91×10^{-3}	-0.8036136
g03	-1.000	-0.9999997	7.05×10^{-7}	-0.9999999	-0.9999996	1.04×10^{-6}	-0.9999999
g04	-30665.539	-30665.539	0.00×10^0	-30665.539	-30665.539	0.00×10^0	-30665.539
g05	5126.498	5171.097	1.42×10^2	5126.498	5171.311	1.43×10^1	5170.288
g06	-6961.814	-6961.814	0.00×10^0	-6961.814	-6961.814	0.00×10^0	-6961.814
g07	24.3062091	24.30649	7.54×10^{-4}	24.30627	24.30705	3.31×10^{-3}	24.307
g08	-0.095825	-0.095825	0.00×10^0	-0.095825	-0.095825	0.00×10^0	-0.095825
g09	680.6301	680.6301	0.00×10^0	680.6301	680.6301	1.00×10^{-5}	680.6301
g10	7049.250	7103.024	8.82×10^2	7049.556	7106.478	8.92×10^1	7049.664
g11	0.750	0.8997895	1.18×10^{-1}	1.000	0.9016549	1.18×10^{-1}	1.000
g12	-1.000	-1.000	0.00×10^0	-1.000	-1.000	0.00×10^0	-1.000
g13	0.053950	0.3132858	2.13×10^{-1}	0.4388565	0.3144292	2.11×10^{-1}	0.4388585

Tabla 6.6: Estadísticas con respecto a la solución obtenida $f(\vec{x})$ para R2 y R3

	Óptimo	CDE			A. Descendente		
		f_{mejor}	f_{peor}	$\%_{\text{éxitos}}$	f_{mejor}	f_{peor}	$\%_{\text{éxitos}}$
g01	-15.000	-15.000	-9.000	10	-15.000	-6.000	16
g02	-0.803619	-0.8035179	-0.8031466	0	-0.803619	-0.688004	10
g03	-1.000	-1.000	0.000	20	-1.000	-1.000	100
g04	-30665.539	-30665.539	-30665.539	100	-30665.539	-30665.539	100
g05	5126.498	5126.49671	6091.59	18	5126.498	6112.225	64
g06	-6961.814	-6961.814	-762.715	86	-6961.814	-724.5741	42
g07	24.3062091	24.306209	24.39152	24	24.306209	25.00435	84
g08	-0.095825	-0.095826	-0.096435	12	-0.095825	-0.095825	100
g09	680.6301	680.630	680.630	100	680.6301	680.6301	100
g10	7049.250	7049.250	9291.431	10	7049.250	7251.029	46
g11	0.750	0.750	0.750	100	0.750	1.000	13
g12	-1.000	-1.000	-1.000	100	-1.000	-1.000	100
g13	0.053950	0.053942	1.000	8	0.053950	0.7728134	1

Tabla 6.7: Porcentaje de éxitos, mejor y peor solución de $f(\vec{x})$ para CDE y el mecanismo aleatorio descendente

	Óptimo	R variable			R1		
		f_{mejor}	f_{peor}	$\%_{\text{éxitos}}$	f_{mejor}	f_{peor}	$\%_{\text{éxitos}}$
g01	-15.000	-15.000	-12.000	27	-15.000	-12.000	31
g02	-0.803619	-0.8036183	-0.7618661	65	-0.8036184	-0.7768445	66
g03	-1.000	-1.000	-1.000	100	-1.000	-1.000	100
g04	-30665.539	-30665.539	-30665.539	100	-30665.539	-30665.539	100
g05	5126.498	5126.498	6030.481	72	5126.498	6029.49	71
g06	-6961.814	-6961.814	-6961.814	100	-6961.814	-6961.814	100
g07	24.3062091	24.306209	24.38309	82	24.306209	24.33157	76
g08	-0.095825	-0.095825	-0.095825	100	-0.095825	-0.095825	100
g09	680.6301	680.6301	680.6301	100	680.6301	680.6301	100
g10	7049.250	7049.250	7250.98	33	7049.250	7251.021	33
g11	0.750	0.750	1.000	29	0.750	1.000	29
g12	-1.000	-1.000	-1.000	100	-1.000	-1.000	100
g13	0.053950	0.053950	0.9992879	5	0.053950	0.9405437	12

Tabla 6.8: Porcentaje de éxitos, mejor y peor solución de $f(\vec{x})$ para R variable y R1

	Óptimo	R2			R3		
		f_{mejor}	f_{peor}	$\%_{\text{éxitos}}$	f_{mejor}	f_{peor}	$\%_{\text{éxitos}}$
g01	-15.000	-15.000	-15.000	100	-15.000	-12.000	27
g02	-0.803619	-0.8036186	-0.7768424	68	-0.8036188	-0.7618704	67
g03	-1.000	-1.000	-0.9999951	98	-1.000	-0.9999932	95
g04	-30665.539	-30665.539	-30665.539	100	-30665.539	-30665.539	100
g05	5126.498	5126.498	6030.067	71	5126.498	6029.822	71
g06	-6961.814	-6961.814	-6961.814	100	-6961.814	-6961.814	100
g07	24.3062091	24.306209	24.31268	79	24.306209	24.30841	72
g08	-0.095825	-0.095825	-0.095825	100	-0.095825	-0.095825	100
g09	680.6301	680.6301	680.6301	100	680.6301	680.6301	100
g10	7049.250	7049.250	7250.97	34	7049.250	7250.997	35
g11	0.750	0.750	1.000	29	0.750	1.000	29
g12	-1.000	-1.000	-1.000	100	-1.000	-1.000	100
g13	0.053950	0.053950	0.9998965	15	0.053950	0.9968109	10

Tabla 6.9: Porcentaje de éxitos, mejor y peor solución de $f(\vec{x})$ para R2 y R3

También podemos observar que para la función $g11$ los enfoques hiperheurísticos no fueron capaces de resolver de manera eficaz esta función ya que el porcentaje de éxitos es muy bajo en comparación con el algoritmo CDE.

En la figura 6.2 se muestra el histograma de frecuencias de soluciones para el problema $g03$ en particular, tanto de la muestra original (100 ejecuciones independientes) como aplicando el método de *bootstrap* con $B = 1000$ remuestreos. De la figura se observa que las soluciones de las 100 ejecuciones independientes no se ajustan a una distribución normal, siendo este comportamiento no propio del ejemplo en particular, sino que también está presente en las demás funciones para cada uno de los algoritmos como se puede apreciar en el Apéndice D.

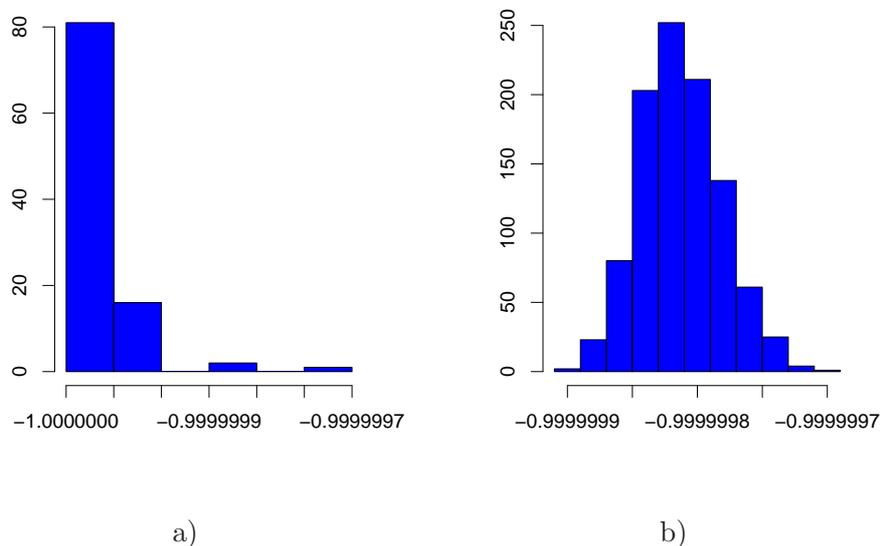


Figura 6.2: Histograma de frecuencias de las soluciones para el problema $g03$ para: a) la muestra original y b) aplicando el método de *bootstrap* a la muestra original

La explicación de esto se debe al hecho de que al estar trabajando con problemas de optimización los algoritmos tienen como objetivo encontrar la solución óptima ($\min f(\vec{x})$), siendo inexistente cualquier solución por debajo de ella y concentrándose la mayor cantidad de las observaciones en el valor óptimo o en valores muy próximos a él, impidiendo de esta manera ajustar la muestra a una forma normal, tal como se puede apreciar en la figura 6.2 a.

Por tal motivo, es necesario emplear un procedimiento de *bootstrap*, tratando de ajustar la muestra a una forma normal y de esta manera llevar a cabo un análisis estadístico más representativo. Para ello se extrajo un gran conjunto de muestras ($B = 1000$) cada una con un tamaño de $n = 100$ observaciones obtenidas a partir de la muestra original (100 corridas independientes) aleatoriamente con reemplazo. De esta manera, cada remuestra tendrá el mismo número de elementos que la muestra original.

Otro modo habitual, y muy útil, de resumir una variable de tipo numérico es utilizando los diagramas de caja como se mencionó anteriormente. En la figura 6.3 se puede apreciar el gráfico de caja para la función $g03$ tanto de la muestra original como al aplicar un procedimiento de Bootstrap. De igual manera que los histogramas, en el Apéndice D se pueden encontrar los gráficos de caja para cada una de las funciones de prueba para espacios con restricciones para el algoritmo **CDE** y la hiperheurística con los mecanismos de selección aleatorio descendente y la ruleta con inicio aleatorio (**R2**), el cual resultó ser el mejor mecanismo de nuestra propuesta.

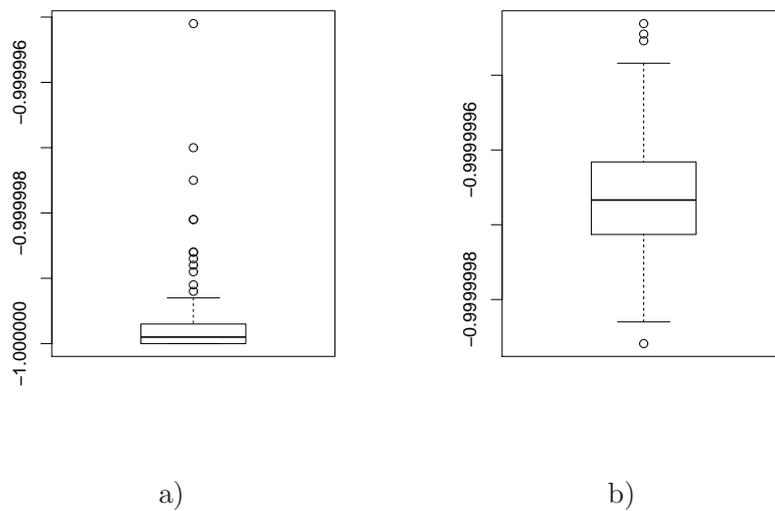


Figura 6.3: Gráficas de caja de las soluciones para el problema $g03$ para: a) la muestra original y b) aplicando el método de *bootstrap* a la muestra original

Para el gráfico de caja de la figura 6.3 que representa a la muestra original podemos observar que se trata de un conjunto de datos el cual tiende a concentrarse hacia el valor óptimo de la función. Así mismo, podemos observar que la caja no es tan grande, por lo que los datos no son muy dispersos. También se puede notar que el gráfico carece de bigote inferior, esto porque el límite inferior corresponde con el primer cuartil (valor óptimo de la función). También se encuentran presentes valores atípicos, los cuales corresponden a valores por encima de -0.999999 con lo que podemos observar que el algoritmo presenta un rendimiento bastante bueno para esta función, ya que una gran cantidad de las muestras está concentrada muy cerca del valor óptimo (-1).

Por otro lado se puede notar que el gráfico de caja que corresponde al método de *bootstrap* aparece más simétrico ya que el valor de la mediana se encuentra ubicado justo en el centro de la caja, debido a que la muestra ya ha sido ajustada a una forma normal.

Del total de gráficos de caja que se muestran en el Apéndice D se pueden encontrar diversos tipos. Para aquellos algoritmos en los que las 100 corridas independientes llegaron

con éxito al valor óptimo de la función, el gráfico de caja corresponde a una línea horizontal ya que todos los datos se concentran en un único valor, siendo éste el valor óptimo de la función. También es posible observar que para las funciones g10 y g11, los enfoques hiper-heurísticos ofrecen resultados muy dispersos y asimétricos, ya que su mediana tiende a estar muy cerca del límite inferior para el caso de g10 y del límite superior para g11.

En las tablas 6.10 y 6.11 se presentan los intervalos de confianza para la media de aptitud de cada función para cada uno de los algoritmos obtenidos a partir de un método de *bootstrap*. Entre más cercano esté el intervalo de confianza al óptimo y entre más cerrado sea dicho rango, más eficiente será el algoritmo.

	Óptimo	CDE	A. Descendente	R variable
g01	-15.000	(-13.765,-14.032)	(-11.77645, -10.71484)	(-12.37828, -11.28462)
g02	-0.803619	(-0.8034006, -0.8033739)	(-0.7894265, -0.7801403)	(-0.8022311, -0.7998746)
g03	-1.000	(-0.271, -0.219)	(-1.000, -1.000)	(-1, -0.9999998)
g04	-30665.539	(-30665.539, -30665.539)	(-30665.539, -30665.539)	(-30665.539, -30665.539)
g05	5126.498	(5277.757, 5345.213)	(5151.12, 5236.357)	(5141.473, 5197.755)
g06	-6961.814	(-7475.834, -6981.317)	(-7534.578, -6984.902)	(-6961.814, -6961.814)
g07	24.3062091	(24.765, 24.812)	(24.29931, 24.32085)	(24.30561, 24.30808)
g08	-0.095825	(0.0959654, 0.0960001)	(-0.095825, -0.095825)	(-0.095825, -0.095825)
g09	680.6301	(680.630, 680.630)	(680.6301, 680.6301)	(680.6301, 680.6301)
g10	7049.250	(7088.781, 7092.395)	(7077.457, 7109.133)	(7085.346, 7119.655)
g11	0.750	(0.750, 0.750)	(0.9333049, 0.9701665)	(0.8763732, 0.9227055)
g12	-1.000	(-1.000, -1.000)	(-1.000, -1.000)	(-1.000, -1.000)
g13	0.053950	(0.738,0.887)	(0.3365415, 0.4007169)	(0.2790923, 0.3686616)

Tabla 6.10: Intervalos de confianza de la solución obtenida $f(\vec{x})$ para CDE, aleatorio descendente y R variable

	Óptimo	R1	R2	R3
g01	-15.000	(-14.89586, -14.63738)	(-15.000, -15.000)	(-14.80757, -14.616555)
g02	-0.803619	(-0.80251, -0.8007213)	(-0.8023635, -0.8005562)	(-0.8026165, -0.8003657)
g03	-1.000	(-0.9999998, -0.9999995)	(-0.9999998, -0.9999995)	(-0.9999998, -0.9999994)
g04	-30665.539	(-30665.539, -30665.539)	(-30665.539, -30665.539)	(-30665.539, -30665.539)
g05	5126.498	(5143.086, 5197.932)	(5142.169, 5196.344)	(5139.635, 5196.589)
g06	-6961.814	(-6961.814, -6961.814)	(-6961.814, -6961.814)	(-6961.814, -6961.814)
g07	24.3062091	(24.30632, 24.30768)	(24.30632, 24.30661)	(24.30636, 24.30757)
g08	-0.095825	(-0.095825, -0.095825)	(-0.095825, -0.095825)	(-0.095825, -0.095825)
g09	680.6301	(680.6301, 680.6301)	(680.6301, 680.6301)	(680.6301, 680.6301)
g10	7049.250	(7086.997, 7121.512)	(7085.018, 7119.162)	(7090.052, 7123.73)
g11	0.750	(0.8779281, 0.9246052)	(0.8763215, 0.9220821)	(0.8790173, 0.9246877)
g12	-1.000	(-1.000, -1.000)	(-1.000, -1.000)	(-1.000, -1.000)
g13	0.053950	(0.2673649, 0.3449967)	(0.2735117, 0.3545732)	(0.2719395, 0.3526922)

Tabla 6.11: Intervalos de confianza de la solución obtenida $f(\vec{x})$ para los tres tipos de ruleta

De las tablas 6.10 y 6.11 se puede observar que el mecanismo de selección de heurísticas de bajo nivel que hace uso de la ruleta con un inicio aleatorio ofrece los mejores resultados para las funciones de prueba. Esto concuerda con los resultados presentados en las tablas 6.6 y 6.9 los cuales fueron discutidos anteriormente. Esto muestra que el mecanismo de selección de la ruleta con inicio aleatorio **R2** muestra los mejores resultados. Además, se aprecia que el algoritmo **CDE** obtiene muy malos resultados en ocho de las trece funciones de prueba, en donde los intervalos de confianza no sólo son muy amplios en varias funciones,

sino que además se encuentran muy lejos del valor óptimo.

En las tablas 6.12, 6.13 y 6.14 se muestran las estadísticas para el valor de aptitud aplicando el método de Bootstrap para cada uno de los algoritmos en todos los problemas de prueba. Un valor en **negritas** indica que es igual al valor óptimo. De las tablas se observa que el mecanismo de selección que usa la ruleta con un inicio aleatorio ofrece los mejores resultados.

	Óptimo	CDE			A. Descendente		
		μ	σ	M	μ	σ	M
g01	-15.000	-13.8832	2.00×10^0	-13.654	-11.23838	2.60×10^{-1}	-11.24078
g02	-0.803619	0.41574	1.45×10^{-1}	0.41658	-0.784473	2.32×10^{-3}	-0.7844999
g03	-1.000	0.24216	4.32×10^{-2}	0.2657	-1.000	5.10×10^{-9}	-1.000
g04	-30665.539	-30665.539	0.00×10^0	-30665.539	-30665.539	0.00×10^0	-30665.539
g05	5126.498	5218.5342	3.17×10^1	5218.4276	5197.062	2.16×10^1	5196.43
g06	-6961.814	-7171.634	0.00×10^0	-6961.814	-7229.388	1.41×10^3	-7237.664
g07	24.3062091	24.865	7.65×10^{-3}	23.954	24.31356	6.82×10^{-3}	24.31351
g08	-0.095825	-0.09685	5.54×10^{-4}	-0.09601	-0.095825	0.00×10^0	-0.095825
g09	680.6301	680.630	0.00×10^0	680.630	680.6301	0.00×10^0	680.6301
g10	7049.250	7090.53256	9.32×10^0	7088.324	7094.163	8.09×10^0	7094.467
g11	0.750	0.750	0.00×10^0	0.750	0.9505673	9.51×10^{-2}	0.950733
g12	-1.000	1.000	0.00×10^0	1.000	-1.000	0.00×10^0	-1.000
g13	0.053950	0.8086534	9.98×10^{-1}	0.799849	0.3673276	1.68×10^{-2}	0.3668168

Tabla 6.12: Estadísticas a través de un procedimiento de *bootstrap* para CDE y mecanismo aleatorio descendente

	Óptimo	R variable			R1		
		μ	σ	M	μ	σ	M
g01	-15.000	-11.83435	2.83×10^{-1}	-11.83734	-11.81047	2.77×10^{-1}	-11.81570
g02	-0.803619	-0.8008857	6.20×10^{-4}	-0.8009084	-0.8015715	4.65×10^{-4}	-0.8016082
g03	-1.000	0.9999999	3.23×10^{-8}	0.9999999	-0.9999997	6.97×10^{-8}	-0.9999997
g04	-30665.539	-30665.539	0.00×10^0	-30665.539	-30665.539	0.00×10^0	-30665.539
g05	5126.498	5170.743	1.38×10^1	5169.873	5171.351	1.40×10^1	5170.653
g06	-6961.814	-7042.782	1.47×10^2	-7042.852	-7062.532	1.42×10^2	-7070.8
g07	24.3062091	24.30723	7.40×10^{-4}	24.30719	24.30708	3.51×10^{-4}	24.30705
g08	-0.095825	-0.095825	0.00×10^0	-0.095825	-0.095825	0.00×10^0	-0.095825
g09	680.6301	680.6301	0.00×10^0	680.6301	680.6301	0.00×10^0	680.6301
g10	7049.250	7103.077	8.61×10^0	7103.065	7102.866	9.07×10^0	7102.755
g11	0.750	0.8995564	1.22×10^{-2}	0.900082	0.9015405	1.19×10^{-2}	0.9015461
g12	-1.000	-1.000	0.00×10^0	-1.000	-1.000	0.00×10^0	-1.000
g13	0.053950	0.3237253	2.20×10^{-2}	0.3234907	0.3063165	1.91×10^{-2}	0.3057378

Tabla 6.13: Estadísticas a través de un procedimiento de *bootstrap* para R variable y R1

Es importante recalcar que todo el análisis estadístico se llevó a cabo usando el lenguaje de programación R³ en su versión 2.10.1 bajo ambiente Linux, el cual ofrece un conjunto de herramientas para el análisis estadístico. A partir de esta herramienta, se hicieron los

³Desarrollado por Robert Gentleman y Ross Ihaka del Departamento de Estadística de la Universidad de Auckland en 1993

	Óptimo	R2			R3		
		μ	σ	M	μ	σ	M
g01	-15.000	-15.000	0.00×10^0	-15.000	-11.83173	2.72×10^{-1}	-11.8238
g02	-0.803619	-0.8014525	4.50×10^{-4}	-0.8014796	-0.801441	5.96×10^{-4}	-0.8015337
g03	-1.000	-0.9999997	7.07×10^{-8}	-0.9999997	-0.9999996	1.07×10^{-7}	-0.9999996
g04	-30665.539	-30665.539	0.00×10^0	-30665.539	-30665.539	0.00×10^0	-30665.539
g05	5126.498	5171.684	1.47×10^1	5170.246	5170.789	1.38×10^1	5169.739
g06	-6961.814	-6961.814	0.00×10^0	-6961.814	-7048.516	1.49×10^2	-7060.968
g07	24.3062091	24.30649	7.63×10^{-5}	24.30648	24.30706	3.35×10^{-4}	24.30702
g08	-0.095825	-0.095825	0.00×10^0	-0.095825	-0.095825	0.00×10^0	-0.095825
g09	680.6301	680.6301	0.00×10^0	680.6301	680.6301	9.97×10^{-9}	680.6301
g10	7049.250	7104.585	8.56×10^0	7104.796	7106.743	8.95×10^0	7106.483
g11	0.750	0.9000194	1.15×10^{-2}	0.9003778	0.9021907	1.20×10^{-2}	0.9024953
g12	-1.000	-1.000	0.00×10^0	-1.000	-1.000	0.00×10^0	-1.000
g13	0.053950	0.3069116	1.96×10^{-2}	0.3063764	0.3138352	2.01×10^{-2}	0.3139811

Tabla 6.14: Estadísticas a través de un procedimiento de *bootstrap* para R2 y R3

cálculos estadísticos presentados así como la generación de los histogramas y gráficos de caja que se presentan en esta tesis.

6.2.3. Comportamiento de las heurísticas de bajo nivel

Con la finalidad de apreciar el comportamiento de cada una de las variantes de Evolución Diferencial que conforman las heurísticas de bajo nivel de la hiperheurística, se generó a partir de las 100 ejecuciones independientes para cada uno de los algoritmos y funciones de prueba un conjunto de gráficas que indican el porcentaje de *buenos movimientos* que aportó cada modelo durante el proceso de búsqueda.

En la figura 6.4 se muestran las gráficas de comportamiento de las variantes de Evolución Diferencial para las funciones g01 y g02. En el Apéndice D se pueden observar el total de gráficas para cada uno de los problemas y mecanismos de selección de la hiperheurística. Se observa de la figura que el comportamiento de cada una de las variantes es muy diferente para cada uno de los problemas, reafirmando que para un problema particular en el que una heurística trabaje muy bien para otro con características diferentes quizás no ofrezca el mismo rendimiento.

Para el ejemplo específico de la figura 6.4 se puede observar que los modelos **ED/rand/1/bin** (h_7) y **ED/current-to-best/1/bin** (h_9) aportaron los mejores movimientos durante el proceso de búsqueda para g01. Por otro lado, **ED/best/1/exp** (h_1) y **ED/current-to-best/1/exp** (h_3) fueron los dos modelos que mejor comportamiento presentaron para la función g02. También se puede observar que para cada uno de los problemas las variantes con cruza *binomial* son mejores para g01 mientras que las estrategias con recombinación *exponencial* lo son para g02.

El comportamiento observado en la figura 6.4 no es único ya que existen problemas como g04 y g06 en los que tanto los modelos con cruza *binomial* como los de cruza *exponencial* ofrecen buenos movimientos durante el proceso de búsqueda.

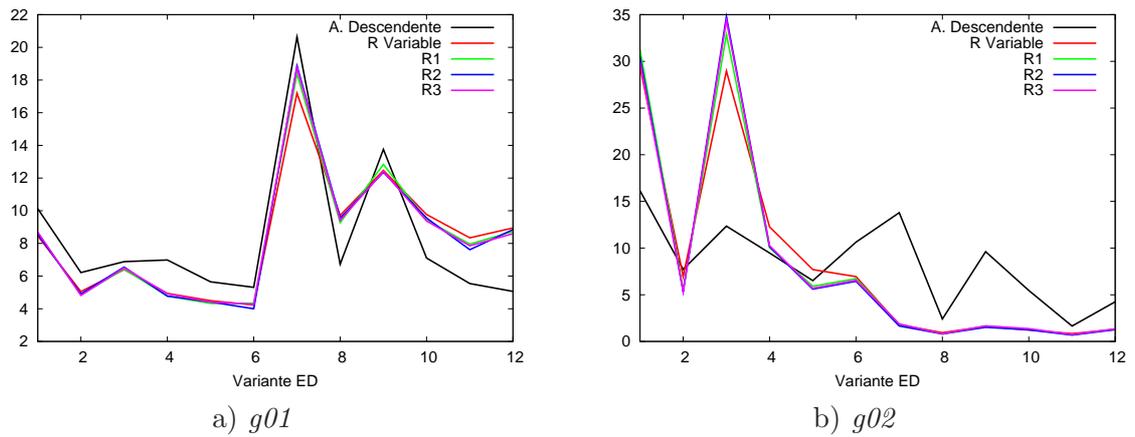


Figura 6.4: Porcentaje de mejoras por cada variante para cada uno de los modelos de selección de heurísticas para las funciones $g01$ y $g02$

Es interesante notar que el mecanismo de selección aleatorio descendente para las funciones $g03$, $g05$, $g07$, $g09$, $g10$, $g11$ y $g13$ ofrece un resultado muy diferente en cuanto al comportamiento de las heurísticas de bajo nivel en comparación con los otros mecanismos, en los que el comportamiento de las heurísticas de bajo nivel es muy similar para los diversos mecanismos de selección.

Como criterio de paro hemos considerado que a lo más el número de evaluaciones de la función objetivo sea igual a 300,000. Sin embargo, no todas las funciones requieren del total de evaluaciones para encontrar la solución óptima. Es por ello que en la tabla 6.15 se presentan el promedio de generaciones en el que los algoritmos convergieron, así como el valor de la generación mínima en el que el algoritmo obtuvo el valor óptimo de la función. Un número en **negritas** indica el valor más pequeño tanto para la media como para el mínimo de generaciones requeridas para cada algoritmo.

	CDE		A. Desc.		CR variable		R1		R2		R3	
	μ	mín	μ	mín	μ	mín	μ	mín	μ	mín	μ	mín
g01	614.88	543	725.93	714	727.84	563	722.37	763	549.32	541	720.14	701
g02	5843.77	5954	5963.33	5934	5960.73	5979	5939.29	5954	5950.26	5907	5975.90	5995
g03	4134.24	3152	5719.55	4217	5693.44	4218	5650.55	4174	5682.42	3128	5598.37	3505
g04	675.33	476	872.34	758	887.19	738	867.98	749	510.26	449	868.86	719
g05	3456.54	1652	4395.93	1786	4293.48	3059	4448.92	1411	3855.19	1346	4316.96	1613
g06	1162.60	154	1100.83	435	931.24	423	1092.50	445	1070.48	291	965.39	496
g07	5945.12	4130	5973.51	5726	5762.79	5834	5766.33	5730	5903.23	4976	6783.88	5757
g08	165.75	98	214.59	100	163.16	97	161.65	97	161.61	78	162.18	102
g09	2621.86	1068	3878.70	1481	3520.84	1581	3518.51	1374	2393.24	1028	3417.16	1384
g10	5954.63	5921	5920.25	5975	5921.15	5985	5943.33	4895	5875.74	4987	5907.12	4989
g11	536.14	367	457.28	302	401.55	302	417.25	302	166.14	230	391.43	466
g12	201.72	163	158.95	134	163.48	119	161.76	129	159.32	119	172.54	132
g13	4119.88	4003	4577.57	3876	5184.46	2529	5007.46	3906	5208.23	2258	5136.14	3111

Tabla 6.15: Promedio y mínimo de generaciones requeridas para el algoritmo CDE y los diversos mecanismos de selección

De la tabla 6.15 se observa que el mecanismo de selección de la ruleta con inicio aleatorio

(**R2**) en diez de las funciones de prueba logra resolver las funciones en una menor cantidad de iteraciones en comparación con los otros mecanismos y del algoritmo CDE. Por otro lado, el promedio de generaciones de las 100 corridas independientes también resulta ser la menor en el mecanismo **R2** para siete de las funciones de prueba, quedando una vez más recalcado que este mecanismo resultó ser el mejor de las cuatro propuestas presentadas en esta tesis.

En la tabla 6.16 se muestra una comparativa entre el mecanismo de selección de la ruleta con inicio aleatorio R2 y la Jerarquización Estocástica. En los resultados de la JE se realizan 350,000 evaluaciones de la función objetivo para encontrar los óptimos de las trece funciones de prueba. A partir de los resultados podemos observar que el mecanismo R2 es competitivo con respecto a esta técnica, ya que logra encontrar las soluciones óptimas de las funciones con un número menor de evaluaciones (300,000).

	Óptimo	R2			JE		
		f_{mejor}	μ	σ	f_{mejor}	μ	σ
g01	-15.000	-15.000	-15.000	0.00×10^0	-15.000	-15.000	0.00×10^0
g02	-0.8036186	-0.803618	-0.8014436	4.67×10^{-3}	-0.803619	-0.781975	2.00×10^{-2}
g03	-1.000	-1.000	-0.9999997	7.05×10^{-7}	-1.000	-1.000	0.00×10^0
g04	-30665.539	-30665.539	-30665.539	0.00×10^0	-30665.539	-30665.539	0.00×10^0
g05	5126.498	5126.498	5171.097	1.42×10^2	5126.497	5128.881	3.50×10^0
g06	-6961.814	-6961.814	-6961.814	0.00×10^0	-6961.814	-6875.940	1.60×10^2
g07	24.3062091	24.306209	24.30649	7.54×10^{-4}	24.307	24.374	6.60×10^{-2}
g08	-0.095825	-0.095825	-0.095825	0.00×10^0	-0.095825	-0.095825	0.00×10^0
g09	680.6301	680.6301	680.6301	0.00×10^0	680.630	680.656	3.40×10^{-2}
g10	7049.250	7049.250	7103.024	8.82×10^2	7049.248	7559.192	5.30×10^2
g11	0.750	0.750	0.8997895	1.18×10^{-1}	0.750	0.750	0.00×10^0
g12	-1.000	-1.000	-1.000	0.00×10^0	-1.000	-1.000	0.00×10^0
g13	0.053950	0.053950	0.3132858	2.13×10^{-1}	0.053950	0.067543	3.10×10^{-2}

Tabla 6.16: Comparativa entre el mecanismo de selección R2 y la Jerarquización Estocástica

Finalmente, analizamos a cada una de las funciones para tratar de encontrar alguna relación que guardaran entre ellas, como en su caso se hizo para los problemas sin restricciones, usando los siguientes aspectos:

- Tipo de función (lineal, cuadrática, etc.)
- Número de variables de decisión de cada problema.
- Tamaño de la región factible.
- Diversidad en la población.

Por último, en el apéndice D, en la pág. 91 se muestran las gráficas de convergencia para las trece funciones de prueba y para los mecanismos de selección propuestos y se les comparó con el mecanismo de selección aleatorio descendente. Cada gráfica corresponde a la solución mediana para las 100 ejecuciones independientes de cada experimento. De estas gráficas se puede observar que el mecanismo de selección **R2** logra converger de manera más rápida en diez de las trece funciones de prueba, siendo esta convergencia más acentuada en funciones como g04, g05, g09 y g10.

Conclusiones y Trabajo Futuro

El problema general de optimización aún permanece como no resuelto; es por ello que aunque se han desarrollado gran cantidad de algoritmos competitivos para dar solución a problemas difíciles, aún se siguen explorando nuevas alternativas. Por esta razón, áreas como la Computación Evolutiva siguen muy activas en investigación en la actualidad.

Sin embargo, estos algoritmos en su forma canónica no cuentan con un mecanismo explícito para el manejo de restricciones, siendo éste otro problema a resolver ya que el manejo de restricciones es una tarea tan difícil como la optimización misma. Para las técnicas de programación matemática dichas limitaciones pueden ser, por ejemplo, que la función objetivo y/o restricciones no son diferenciables. Por otro lado, el problema general de las técnicas de manejo de restricciones en los Algoritmos Evolutivos es lograr el balance correcto entre tener prioridad en minimizar la función objetivo o el grado de violación de las restricciones.

Otro inconveniente que surge es la imposibilidad de diseñar un algoritmo heurístico que sea mejor que los demás en todas las clases de problemas. En esta tesis se estudió el concepto de *hiperheurística*, que es un mecanismo que se encarga de escoger heurísticas para aplicarlas en un momento determinado del proceso de búsqueda para dar solución a problemas de optimización. En este trabajo dichas heurísticas están conformadas por doce modelos del algoritmo de la Evolución Diferencial (ED), el cual ha demostrado ser un algoritmo *robusto* de búsqueda en problemas continuos.

De tal manera, en esta tesis se presentan cuatro mecanismos de selección para un enfoque hiperheurístico basado en los modelos originales de ED. Uno de éstos enfoques de selección se basa en una selección aleatoria y tres enfoques usando tres versiones de la ruleta, uno de los mecanismos de selección más ampliamente usado en el área de la computación evolutiva. Aunado a esto, se incorporó un manejo de restricciones basado en la Jerarquización Estocástica (JE), que a diferencia de otras propuestas, no se requiere ajustar ningún coeficiente de penalización y que ha sido la mejor técnica de manejo de restricciones propuesta a la fecha.

La propuesta de hiperheurística en espacios restringidos se basa en agregar información sobre la dirección idónea de búsqueda al operador de mutación de la ED proporcionada por la jerarquización de los individuos mediante el procedimiento de ordenamiento de la JE, con el objetivo de guiar el proceso de búsqueda mejorando de ésta manera la capacidad de exploración del algoritmo en la frontera entre la región factible e infactible del espacio de

búsqueda.

Doce modelos de Evolución Diferencial se estudiaron con el propósito de identificar algunas propiedades que permitieron el diseño del mecanismo de selección de la hiperheurística. Para esta primera etapa se usaron únicamente problemas de optimización sin restricciones escalables en su número de variables. De estos resultados se concluyó que el parámetro de la constante de recombinación R de la ED juega un papel muy importante en el tipo de recombinación a usar. También se pudo observar que entre más variables de decisión presentaba el problema ($D > 30$), los modelos con recombinación exponencial resultaban mejores.

A partir de las observaciones realizadas se diseñó el primer mecanismo de selección que consistió en variar en cada generación el valor del parámetro R y, a partir del valor que éste tomara seleccionar aleatoriamente modelos con cruza exponencial o binomial. Este mecanismo fue denominado R variable. Posteriormente, se diseñó un segundo mecanismo de selección que hace uso del algoritmo de la ruleta para elegir de una manera más “adecuada” el modelo específico de ED. Para este mecanismo se varía igualmente el parámetro R para elegir el tipo de recombinación a usar, siendo diferente la elección del modelo, ya que en este caso no se elige aleatoriamente, sino que la elección tenderá a elegir a aquellas variantes que han presentado los mejores rendimientos durante el proceso de búsqueda haciendo uso del algoritmo de la ruleta. Se implementaron tres variantes: la ruleta original (denominado R1), variando el punto de inicio de la ruleta (denominado R2) y una última variante que permuta las posiciones de la ruleta (denominado R3).

Al igual que los algoritmos genéticos, la Evolución Diferencial es una heurística de búsqueda global que no tiene empujado ningún mecanismo para manejar restricciones en su versión original. En la literatura de la ED se han utilizado ampliamente las funciones de penalización como técnica para manejar restricciones. Sin embargo, como se indicó en el Capítulo 4, éstas presentan la gran desventaja de requerir que los coeficientes de penalización sean ajustados para cada tipo de problema.

A partir de la idea de la JE de ordenar a la población, tanto considerando la función objetivo como la violación de las restricciones, se diseñó un mecanismo para elegir los vectores de perturbación de la Evolución Diferencial para guiar el proceso de búsqueda tanto hacia adentro de la región factible como en dirección hacia la región infactible con el objetivo de explorar la frontera entre ambas regiones.

Se efectuaron una serie de experimentos con el fin de validar el rendimiento de la propuesta. Los experimentos consistieron en llevar a cabo 100 ejecuciones independientes para los cuatro mecanismos de selección propuestos, así como para el algoritmo de Evolución Diferencial Restringida [46] y el mecanismo de selección aleatorio descendente [16]. A partir de las 100 ejecuciones independientes se realizó un estudio estadístico que incluyó un análisis tradicional con las medidas de tendencia central. También se usaron los gráficos de caja para observar de una manera más clara la distribución de los datos y procedimientos estadísticos de cómputo intensivo que nos permitieron obtener aproximaciones a distribuciones muestrales de las diferentes muestras.

Finalmente, también se muestran una serie de gráficas que exponen el comportamiento que presentó cada uno de los modelos de Evolución Diferencial para los distintos métodos hiperheurísticos que se trabajaron en esta tesis; esto con el propósito de verificar cómo para cada problema, las variantes se comportan de manera muy diferente.

De acuerdo a los resultados presentados en el Capítulo 6 podemos decir que el mecanismo de selección de la ruleta con inicio aleatorio resulta ser competitivo, ya que logra obtener mejores resultados que el algoritmo CDE ofreciendo las mejores tasas de éxito. Por otro lado, resulta interesante notar que los enfoques hiperheurísticos se comportaron de mejor manera que el algoritmo CDE, con lo que se confirma que estas nuevas propuestas de combinar varias heurísticas resulta ser un tema de investigación muy importante en el diseño de nuevos algoritmos hiperheurísticos.

Para el trabajo futuro, existen varios temas por investigar. Se puede mejorar el mecanismo de selección haciendo un estudio más profundo del comportamiento de los modelos de ED a fin de entender más claramente cómo operan en las diversas funciones de prueba con y sin restricciones, y así poder proponer una alternativa para mejorar el comportamiento del mecanismo de selección. También se podría intentar acoplar otro tipo de heurísticas como el recocido simulado, cúmulo de partículas, algoritmos genéticos, etc., con el fin de que éstos pudieran emplearse cuando ninguno de los modelos de la Evolución Diferencial sean capaces de producir mejores resultados.

Se piensa que el dirigir la dirección de la búsqueda tanto hacia la región factible como hacia la infactible, puede ayudar a resolver los problemas de manera más eficiente, tratando de explorar más adecuadamente la frontera entre la región factible e infactible al tener mayor diversidad en la población. Para responder a esta cuestión se requiere realizar un análisis profundo de la forma de escoger a los vectores para generar la perturbación al vector base tanto hacia dentro como hacia fuera de la región factible. Sin embargo, esto último ya no se comprobó en esta tesis, quedando como otro tema de investigación a ser trabajado.

Funciones de Prueba sin Restricciones

1. Función de Ackley

$$f_1 = -20 \exp \left(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D x_i^2} \right) - \exp \left(\frac{1}{D} \sum_{i=1}^D \cos(2\pi x_i) \right) + 20 + e \quad (\text{A.1})$$

donde:

$$x_j \in [-32, 32] \quad \forall j = 1, 2, \dots, D$$

$$\text{mín } f_1 = f(0, 0, \dots, 0) = 0$$

2. Hiper-Elipsoide

$$f_2 = \sum_{j=1}^D j^2 x_j^2 \quad (\text{A.2})$$

donde:

$$x_j \in [-1, 1] \quad \forall j = 1, 2, \dots, D$$

$$\text{mín } f_2 = f(0, 0, \dots, 0) = 0$$

3. Función de Rastrigin

$$f_3 = 10D + \sum_{j=1}^D (x_j^2 - 10 \cdot \cos(2\pi * x_j)) \quad (\text{A.3})$$

donde:

$$x_j \in [-600, 600] \quad \forall j = 1, 2, \dots, D$$

$$\text{mín } f_3 = f(0, 0, \dots, 0) = 0$$

4. Función de Griewank

$$f_4 = \sum_{j=1}^D \frac{x_j^2}{4000} - \prod_{j=1}^D \cos\left(\frac{x_j}{\sqrt{j}}\right) + 1 \quad (\text{A.4})$$

donde:

$$x_j \in [-600, 600] \quad \forall j = 1, 2, \dots, D$$

$$\text{mín } f_4 = f(0, 0, \dots, 0) = 0$$

5. Función de penalización de Shubert

$$f_5 = g_2(x) + u(x, 10, 100, 2) \quad (\text{A.5})$$

donde:

$$g_2(x) = \frac{\pi}{D} \left\{ 10 \sin^2 \left[\pi + \frac{\pi}{4} (x_1 - 1) \right] + \sum_{i=1}^{D-1} 0.125 (x_i - 1)^2 \left[1 + 10 \sin^2 \left(\pi + \frac{\pi}{4} (x_{i+1} - 1) \right) \right] \right. \\ \left. + 0.125 (x_D - 1)^2 \right\}$$

y

$$u(z, a, k, m) = \begin{cases} k (z - a)^m & z > a \\ 0 & -a \leq z \leq a \\ k (-z - a)^m & z < -a \end{cases}$$

para:

$$x_j \in [-50, 50] \quad \forall j = 1, 2, \dots, D$$

$$\text{mín } f_5 = f(1, 1, \dots, 1) = 0$$

6. Problema de Schwefel 1.2

$$f_6 = \sum_{i=1}^D \left(\sum_{j=1}^i x_j \right)^2 \quad (\text{A.6})$$

donde:

$$x_i \in [-100, 100] \quad \forall i = 1, 2, \dots, D$$

$$\text{mín } f_6 = f(0, 0, \dots, 0) = 0$$

7. Problema de Schwefel 2.22

$$f_7 = \sum_{i=1}^D |x_i| + \prod_{i=1}^D |x_i| \quad (\text{A.7})$$

donde:

$$x_i \in [-10, 10] \quad \forall i = 1, 2, \dots, D$$

$$\text{mín } f_7 = f(0, 0, \dots, 0) = 0$$

8. Problema generalizado de Schwefel 2.26

$$f_8 = \sum_{i=1}^D \left(x_i \sin \left(\sqrt{|x_i|} \right) \right) \quad (\text{A.8})$$

donde:

$$x_i \in [-500, 500] \quad \forall i = 1, 2, \dots, D$$

$$\text{mín } f_8 = f(-420.9687, -420.9687, \dots, -420.9687)$$

9. Función de Levy

$$f_9 = 0.1 \left\{ \sin^2(3\pi x_1) + \sum_{i=1}^{D-1} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})] + (x_D - 1) [1 + \sin^2(2\pi x_D)] \right\} + \sum_{i=1}^D u(x_i, 5, 100, 4) \quad (\text{A.9})$$

donde:

$$x_i \in [-50, 50] \quad \forall i = 1, 2, \dots, D$$

$$\text{mín } f_9 = f(1, 1, \dots, 1) = 0$$

Resultados Experimentales para Optimización sin Restricciones

En este apéndice se muestran los resultados experimentales para los problemas de optimización numérica sin restricciones.

En las tablas B.1 a B.5 se muestran el promedio de $f(\vec{x})$ y el número de generaciones requeridas para cuatro distintos números de variables de decisión D (10, 30, 50 y 100) para cada una de las funciones de estudio.

Los resultados corresponden a una comparativa entre el modelo **ED/rand/1/bin**, al mecanismo de selección aleatorio descendente [16] ($R = 0.2$ y $R = 0.8$), variando el parámetro R y usando los tres tipos de ruletas, donde **R1** corresponde al algoritmo original de la ruleta, **R2** es la ruleta con inicio aleatorio y, finalmente, **R3** es la ruleta con una permutación de las posiciones.

Finalmente, en la tabla B.6 se muestran los resultados correspondientes a las funciones restantes para el mecanismo de selección usando la ruleta con inicio aleatorio, que en términos generales produjo los mejores resultados en las funciones f_1 a f_5 .

	ED/rand/ 1/bin	Hiperheurística					
		$R = 0.2$	$R = 0.8$	R variable	R1	R2	R3
D = 10							
$f(\vec{x})$	5.37×10^{-12}	3.68×10^{-12}	5.11×10^{-12}	4.07×10^{-12}	4.10×10^{-12}	4.17×10^{-12}	3.78×10^{-12}
G_{prom}	663.20	827.90	449.30	603.60	536.40	537.00	538.50
D = 30							
$f(\vec{x})$	1.71×10^{-11}	1.20×10^{-11}	2.11×10^{-11}	1.39×10^{-11}	1.49×10^{-11}	1.38×10^{-11}	1.55×10^{-11}
G_{prom}	2103.10	2860.30	1938.50	2146.70	2001.80	1878.40	1942.30
D = 50							
$f(\vec{x})$	2.48×10^{-11}	2.14×10^{-11}	1.33×10^0	2.06×10^{-11}	2.18×10^{-11}	2.55×10^{-11}	2.02×10^{-11}
G_{prom}	3758.10	5444.60	2977.50	3975.90	3608.80	3648.40	3644.80
D = 100							
$f(\vec{x})$	4.07×10^{-11}	3.98×10^{-8}	2.40×10^0	3.63×10^{-11}	3.76×10^{-11}	3.65×10^{-11}	3.78×10^{-11}
G_{prom}	9207.60	10000.00	6042.20	8861.30	8414.40	8403.30	8380.80

Tabla B.1: Resultados experimentales para f_1

	ED/rand/ 1/bin	Hiperheurística					
		$R = 0.2$	$R = 0.8$	R variable	R1	R2	R3
D = 10							
$f(\vec{x})$	6.79×10^{-13}	5.83×10^{-13}	7.93×10^{-13}	9.27×10^{-13}	1.09×10^{-12}	7.28×10^{-13}	1.05×10^{-12}
G_{prom}	413.30	481.10	359.10	368.70	356.40	321.00	327.40
D = 30							
$f(\vec{x})$	5.62×10^{-12}	4.62×10^{-12}	2.00×10^{-11}	4.51×10^{-12}	5.30×10^{-12}	5.84×10^{-12}	4.15×10^{-12}
G_{prom}	1591.80	2221.90	1656.30	1519.70	1447.70	1419.60	1389.10
D = 50							
$f(\vec{x})$	1.30×10^{-10}	7.71×10^{-12}	9.95×10^{-12}	6.97×10^{-12}	1.06×10^{-11}	8.52×10^{-12}	6.70×10^{-12}
G_{prom}	4891.00	4812.70	3295.90	3032.90	2809.50	2841.50	2845.90
D = 100							
$f(\vec{x})$	3.27×10^{-2}	3.78×10^{-5}	1.52×10^{-11}	1.19×10^{-11}	1.73×10^{-11}	1.29×10^{-11}	2.44×10^{-11}
G_{prom}	10000.00	10000.00	8644.50	7336.10	7053.20	7057.90	7105.90

Tabla B.2: Resultados experimentales para f_2

	ED/rand/ 1/bin	Hiperheurística					
		$R = 0.2$	$R = 0.8$	R variable	R1	R2	R3
D = 10							
$f(\vec{x})$	1.03×10^{-12}	8.14×10^{-13}	2.69×10^0	8.37×10^{-13}	6.80×10^{-13}	8.26×10^{-13}	1.06×10^{-12}
G_{prom}	609.80	823.60	522.80	598.60	569.10	595.70	567.40
D = 30							
$f(\vec{x})$	5.45×10^{-12}	4.37×10^{-12}	1.21×10^1	4.64×10^{-12}	4.80×10^{-12}	4.58×10^{-12}	4.34×10^{-12}
G_{prom}	2502.60	2196.40	2397.40	2397.40	2427.10	2323.20	2352.00
D = 50							
$f(\vec{x})$	3.98×10^{-1}	1.49×10^0	2.24×10^1	8.71×10^{-1}	6.49×10^{-12}	6.21×10^{-12}	7.01×10^{-12}
G_{prom}	5026.20	6960.40	4502.90	4320.20	5067.89	4983.13	4994.40
D = 100							
$f(\vec{x})$	1.13×10^1	3.08×10^1	5.48×10^1	2.69×10^0	4.16×10^{-8}	5.14×10^{-8}	1.50×10^{-7}
G_{prom}	10000.00	10000.00	9895.50	10000.00	10000.00	10000.00	10000.00

Tabla B.3: Resultados experimentales para f_3

	ED/rand/ 1/bin	Hiperheurística					
		$R = 0.2$	$R = 0.8$	R variable	R1	R2	R3
D = 10							
$f(\vec{x})$	1.44×10^{-12}	5.37×10^{-13}	4.70×10^{-2}	7.14×10^{-13}	1.53×10^{-2}	2.00×10^{-2}	1.95×10^{-2}
G_{prom}	770.30	1132.10	432.50	708.80	666.50	586.60	574.00
D = 30							
$f(\vec{x})$	5.43×10^{-12}	7.40×10^{-4}	5.42×10^{-3}	3.03×10^{-12}	4.72×10^{-12}	3.66×10^{-12}	3.74×10^{-12}
G_{prom}	1316.30	2105.20	1473.30	1518.90	1415.30	1376.90	1394.80
D = 50							
$f(\vec{x})$	8.62×10^{-12}	9.86×10^{-4}	7.38×10^{-3}	6.28×10^{-12}	8.03×10^{-12}	7.49×10^{-12}	7.43×10^{-12}
G_{prom}	2802.00	3931.20	2891.80	2758.50	2602.10	2544.90	2422.00
D = 100							
$f(\vec{x})$	1.63×10^{-11}	1.39×10^{-11}	2.89×10^{-11}	3.13×10^{-11}	1.54×10^{-11}	2.12×10^{-11}	1.44×10^{-11}
G_{prom}	6924.00	9616.00	7276.70	6255.10	5888.40	5778.40	5759.20

Tabla B.4: Resultados experimentales para f_4

	ED/rand/ 1/bin	Hiperheurística					
		$R = 0.2$	$R = 0.8$	R variable	R1	R2	R3
D = 10							
$f(\vec{x})$	1.12×10^{-12}	6.18×10^{-13}	1.36×10^{-12}	6.05×10^{-13}	5.58×10^{-13}	6.29×10^{-13}	8.84×10^{-13}
G_{prom}	398.80	500.10	341.30	389.70	355.70	352.70	355.30
D = 30							
$f(\vec{x})$	5.43×10^{-12}	4.44×10^{-12}	6.12×10^{-12}	4.58×10^{-12}	4.92×10^{-12}	5.39×10^{-12}	4.27×10^{-12}
G_{prom}	1316.30	1840.00	1949.70	1393.70	1263.10	1263.30	1281.10
D = 50							
$f(\vec{x})$	8.34×10^{-12}	8.02×10^{-12}	1.47×10^{-11}	8.17×10^{-12}	9.62×10^{-12}	9.35×10^{-12}	7.07×10^{-12}
G_{prom}	2509.90	3458.20	4765.10	2469.80	2303.40	2294.60	2304.50
D = 100							
$f(\vec{x})$	1.72×10^{-11}	1.58×10^{-11}	3.26×10^0	1.38×10^{-11}	1.54×10^{-11}	1.31×10^{-11}	1.44×10^{-11}
G_{prom}	6388.50	8572.80	9232.10	5707.20	5439.80	5397.80	5388.60

Tabla B.5: Resultados experimentales para f_5

D	Parámetro	f_6	f_7	f_8	f_9
10	$f(\vec{x})$	3.57×10^{-13}	3.78×10^{-12}	-4.18×10^3	3.79×10^{-13}
	g_{en}	2652.80	504.40	383.80	348.40
	$\bar{f}(\vec{x})$	5.57×10^{-11}	1.14×10^{-11}	-1.26×10^4	4.00×10^{-12}
30	g_{en}	10000.00	1868.10	1451.60	1288.30
	$\bar{f}(\vec{x})$	6.65×10^{-10}	2.13×10^{-11}	-2.09×10^4	7.89×10^{-12}
50	g_{en}	10000.00	3546.80	2640.90	2347.70
	$\bar{f}(\vec{x})$	4.50×10^{-10}	3.10×10^{-11}	-4.19×10^4	1.32×10^{-11}
100	g_{en}	10000.00	8140.00	6136.00	5546.70

Tabla B.6: Resultados experimentales para las funciones f_6 a f_9 usando el mecanismo de selección de la ruleta con inicio aleatorio.

Funciones de Prueba con Restricciones

1. **g01**

Minimizar:

$$f(x) = 5 \sum_{i=1}^4 x_i - 5 \sum_{i=1}^4 x_i^2 - \sum_{i=5}^{13} x_i \quad (\text{C.1})$$

sujeto a:

$$g_1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0$$

$$g_2(x) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0$$

$$g_3(x) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0$$

$$g_4(x) = -8x_1 + x_{10} \leq 0$$

$$g_5(x) = -8x_2 + x_{11} \leq 0$$

$$g_6(x) = -8x_3 + x_{12} \leq 0$$

$$g_7(x) = -2x_4 - x_5 + x_{10} \leq 0$$

$$g_8(x) = -2x_6 - x_7 + x_{11} \leq 0$$

$$g_9(x) = -2x_8 - x_9 + x_{12} \leq 0$$

donde:

$$0 \leq x_i \leq 1 \quad (i = 1, 2, \dots, 9), \quad 0 \leq x_i \leq 100 \quad (i = 10, 11, 12) \quad \text{y} \quad 0 \leq x_{13} \leq 1$$

$$x^* = [1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1]^T$$

$$\text{mín } f(x^*) = -15$$

Las restricciones g_1, g_2, g_3, g_7, g_8 y g_9 son activas.

2. **g02**

Maximizar:

$$f(x) = \left| \frac{\sum_{i=1}^n \cos^4(x_i) - 2 \prod_{i=1}^n \cos^2(x_i)}{\sqrt{\sum_{i=1}^n i x_i^2}} \right| \quad (\text{C.2})$$

sujeto a:

$$g_1(x) = 0.75 - \prod_{i=1}^n x_i \leq 0$$

$$g_2(x) = \sum_{i=1}^n x_i - 7.5n \leq 0$$

donde:

$$n = 20$$

$$0 \leq x_i \leq 10 \quad (i = 1, 2, \dots, n)$$

El óptimo es desconocido. La mejor aproximación conocida es:

$$\text{máx } f(x^*) = 0.803619$$

3. **g03**

Maximizar:

$$f(x) = (\sqrt{n})^n \prod_{i=1}^n x_i \quad (\text{C.3})$$

sujeto a:

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

donde:

$$n = 10$$

$$0 \leq x_i \leq 1 \quad (i = 1, 2, \dots, n)$$

$$x^* = \frac{1}{\sqrt{n}} [1, 1, \dots, 1]^T$$

$$\text{máx } f(x^*) = 1$$

4. **g04**

Minimizar:

$$f(x) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792 - 141 \quad (\text{C.4})$$

sujeto a:

$$g_1(x) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0$$

$$g_2(x) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leq 0$$

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

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

$$g_5(x) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0$$

$$g_6(x) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0$$

donde:

$$78 \leq x_1 \leq 102, 33 \leq x_2 \leq 45 \text{ y } 27 \leq x_i \leq 45 \text{ (} i = 3, 4, 5\text{)}$$

$$x^* = [78, 33, 29.995256025682, 45, 36.775812905788]^T$$

$$\text{mín } f(x^*) = -30665.539$$

Las restricciones g_1 y g_6 son activas.5. **g05**

Minimizar:

$$f(x) = 3x_1 + 0.000001x_1^3 + 2x_2 + \frac{0.000002}{3}x_2^3 \quad (\text{C.5})$$

sujeto a:

$$g_1(x) = -x_4 + x_3 - 0.55 \leq 0$$

$$g_2(x) = -x_3 + x_4 - 0.55 \leq 0$$

$$h_1(x) = 1000 \sin(-x_3 - 0.25) + 1000 \sin(-x_4 - 0.25) + 894.8 - x_1 = 0$$

$$h_2(x) = 1000 \sin(x_3 - 0.25) + 1000 \sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$$

$$h_3(x) = 1000 \sin(x_4 - 0.25) + 1000 \sin(x_4 - x_3 - 0.25) + 1294.8 = 0$$

donde:

$$0 \leq x_i \leq 1200 \text{ (} i = 1, 2\text{) y } -0.55 \leq x_i \leq 0.55 \text{ (} i = 3, 4\text{)}$$

$$x^* = [679.9453, 1026.067, 0.1188764, -0.3962336]^T$$

$$\text{mín } f(x^*) = 5126.4981$$

6. g06

Minimizar:

$$f(x) = (x_1 - 10)^3 + (x_2 - 20)^3 \quad (\text{C.6})$$

sujeto a:

$$g_1(x) = -(x_1 - 5)^3 + (x_2 - 5)^3 + 100 \leq 0$$

$$g_2(x) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0$$

donde:

$$13 \leq x_1 \leq 100 \text{ y } 0 \leq x_2 \leq 100$$

$$x^* = [14.095, 0.84296]^T$$

$$\text{mín } f(x^*) = -6961.81388$$

Ambas restricciones son activas.

7. g07

Minimizar:

$$f(x) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5) + (x_5 - 3)^2$$

$$+ 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45 \quad (\text{C.7})$$

sujeto a:

$$g_1(x) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0$$

$$g_2(x) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0$$

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

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

$$g_5(x) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 50 \leq 0$$

$$g_6(x) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leq 0$$

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

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

donde:

$$-10 \leq x_i \leq 10 \quad (i = 1, 2, \dots, 10)$$

$$x^* = [2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927]^T$$

$$\text{mín } f(x^*) = 24.3062091$$

Las restricciones g_1, g_2, g_3, g_4, g_5 y g_6 son activas.

8. g08

Maximizar:

$$f(x) = \frac{\sin^3(2\pi x_1) \sin(2\pi x_2)}{x_1^3(x_1 + x_2)} \quad (\text{C.8})$$

sujeto a:

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

donde:

$$0 \leq x_1 \leq 10 \text{ y } 0 \leq x_2 \leq 10$$

$$x^* = [1.2279713, 4.2453733]^T$$

$$\text{máx } f(x^*) = 0.095825$$

9. g09

Minimizar:

$$\begin{aligned} f(x) = & (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 \\ & + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7 \end{aligned} \quad (\text{C.9})$$

sujeto a:

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

donde:

$$-10 \leq x_i \leq 10 \quad (i = 1, 2, \dots, 7)$$

$$x^* = [2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227]^T$$

$$\text{mín } f(x^*) = 680.6300573$$

Las restricciones g_1 y g_4 son activas.

10. **g10**

Minimizar:

$$f(x) = x_1 + x_2 + x_3 \quad (\text{C.10})$$

sujeto a:

$$g_1(x) = -1 + 0.0025(x_4 + x_6) \leq 0$$

$$g_2(x) = -1 + 0.0025(x_5 + x_7 - x_4) \leq 0$$

$$g_3(x) = -1 + 0.01(x_8 - x_5) \leq 0$$

$$g_4(x) = -x_1x_6 + 833.33252x_4 + 100x_1 - 83333.333 \leq 0$$

$$g_5(x) = -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \leq 0$$

$$g_6(x) = -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \leq 0$$

donde:

$$100 \leq x_1 \leq 10000, 1000 \leq x_i \leq 10000 \quad (i = 2, 3) \text{ y } 10 \leq x_i \leq 1000 \quad (i = 4, 5, \dots, 8)$$

$$x^* = [579.19, 1360.13, 5109.92, 182.01, 295.60, 217.99, 286.40, 395.60]^T$$

$$\text{mín } f(x^*) = 7049.25$$

Las restricciones g_1, g_2 y g_3 son activas.

11. **g11**

Minimizar:

$$f(x) = x_1^2 + (x_2 - 1)^2 \quad (\text{C.11})$$

sujeto a:

$$h_1(x) = x_2 - x_1^2 = 0$$

donde:

$$-1 \leq x_1 \leq 1 \text{ y } -1 \leq x_2 \leq 1$$

$$x^* = \left[\pm \frac{1}{\sqrt{2}}, \frac{1}{2} \right]^T$$

$$\text{mín } f(x^*) = 0.75$$

12. g12

Maximizar:

$$f(x) = \frac{100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2}{100} \quad (\text{C.12})$$

sujeto a:

$$g_1(x) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0$$

donde:

$$0 \leq x_i \leq 10 \text{ (} i = 1, 2, 3 \text{) y } p, q, r = 1, 2, \dots, 9$$

Un punto $[x_1, x_2, x_3]^T$ es factible si y sólo si existen p, q, r tal que la desigualdad se cumple.

$$x^* = [5, 5, 5]^T$$

$$\text{máx } f(x^*) = -1$$

13. g13

Minimizar:

$$f(x) = e^{x_1 x_2 x_3 x_4 x_5} \quad (\text{C.13})$$

sujeto a:

$$h_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0$$

$$h_2(x) = x_2 x_3 - 5 x_4 x_5 = 0$$

$$h_3(x) = x_1^3 + x_2^3 + 1 = 0$$

donde:

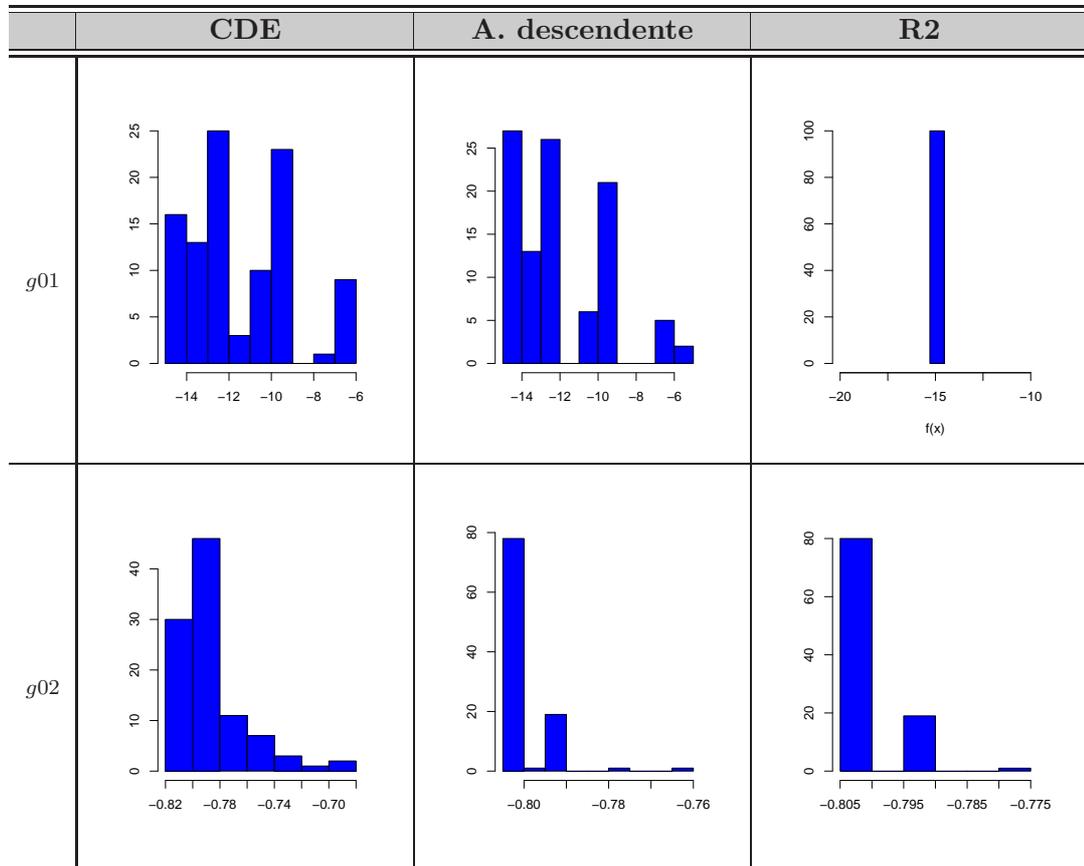
$$-2.3 \leq x_i \leq 2.3 \text{ (} i = 1, 2 \text{) y } -3.2 \leq x_i \leq 3.2 \text{ (} i = 3, 4, 5 \text{)}$$

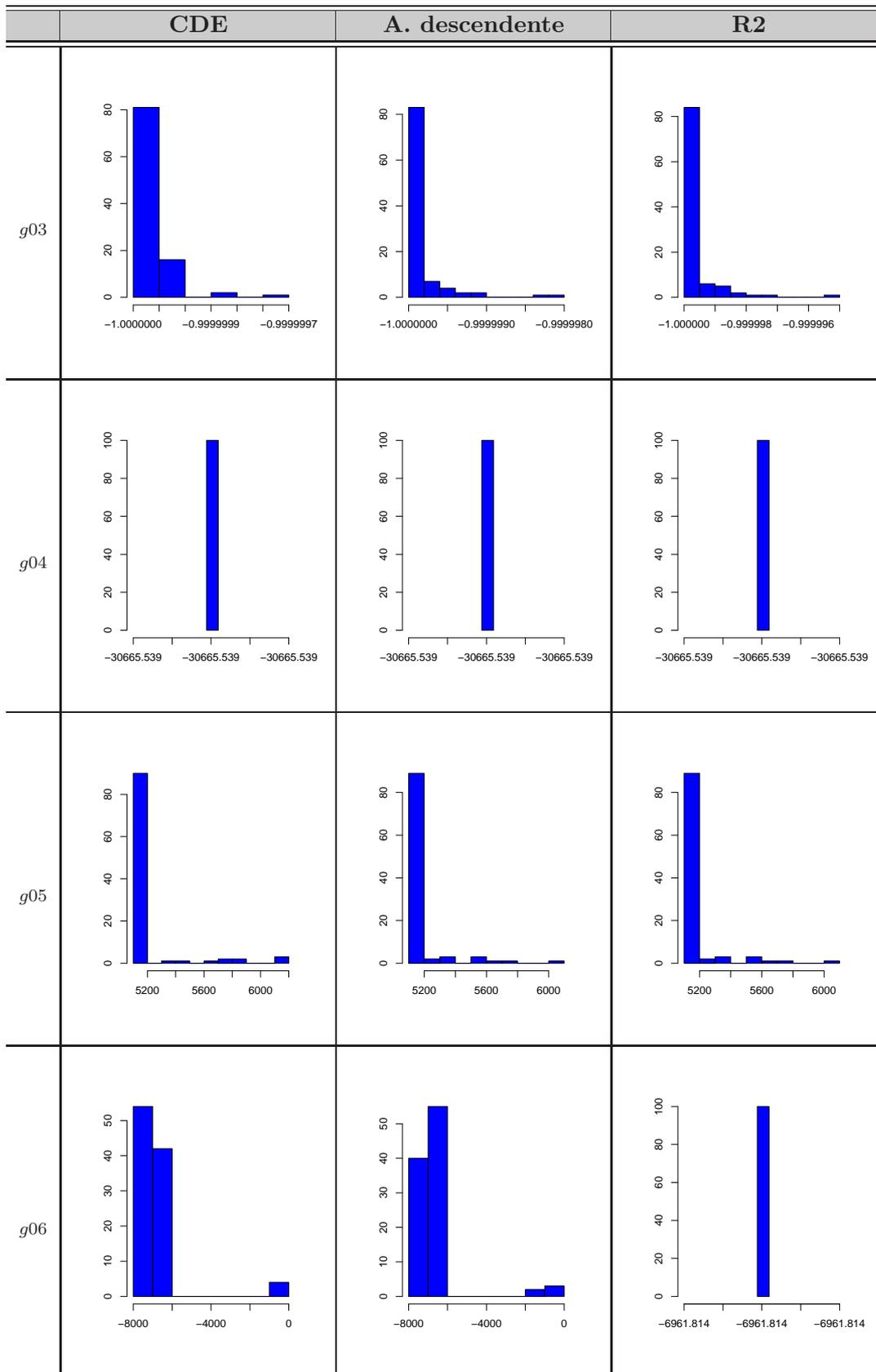
$$x^* = [-1.717143, 1.595709, 1.827247, -0.7636413, -0.763645]^T$$

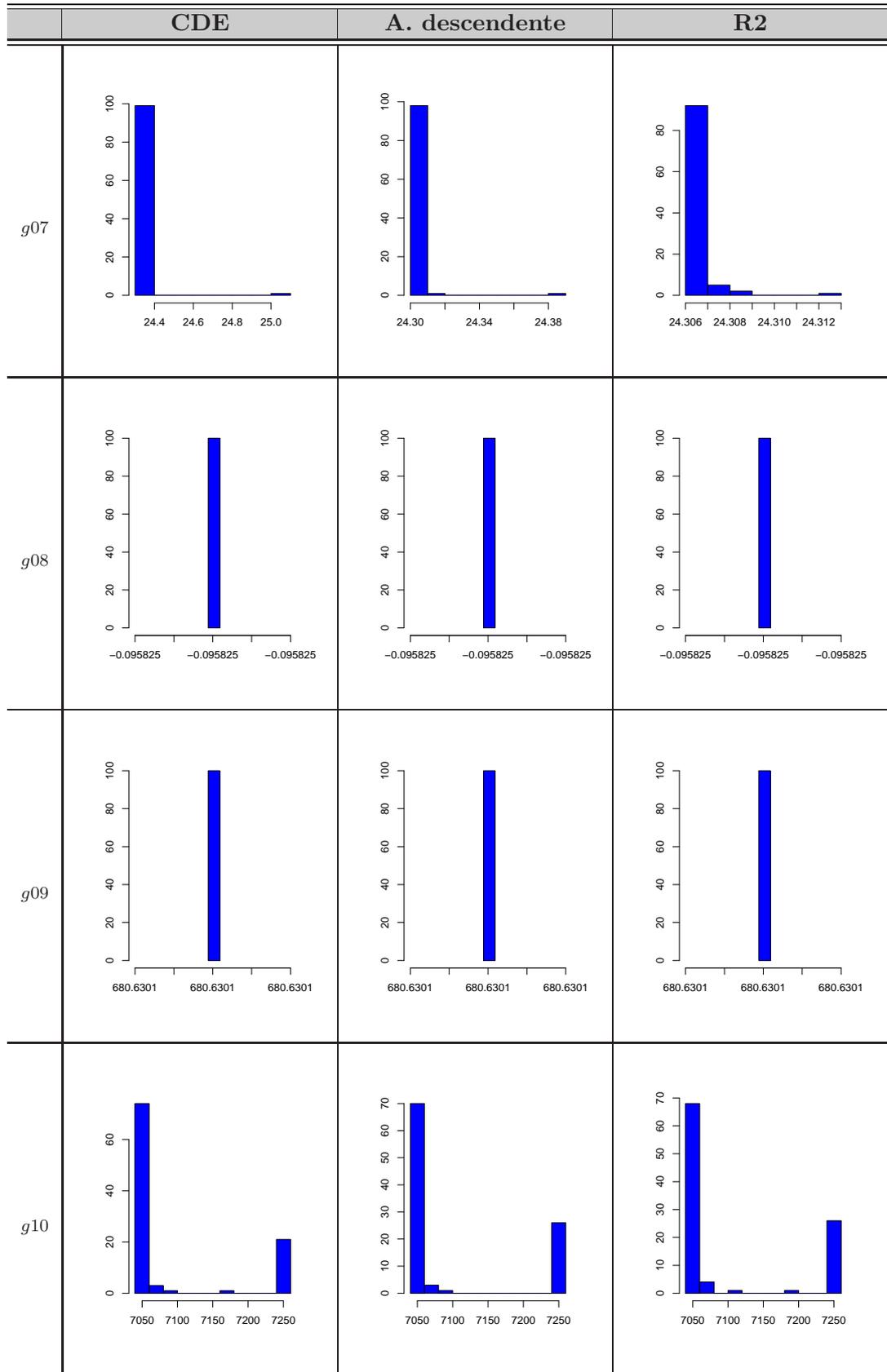
$$\text{mín } f(x^*) = 0.0539498$$

Gráficas de Resultados

D.1. Histogramas de resultados







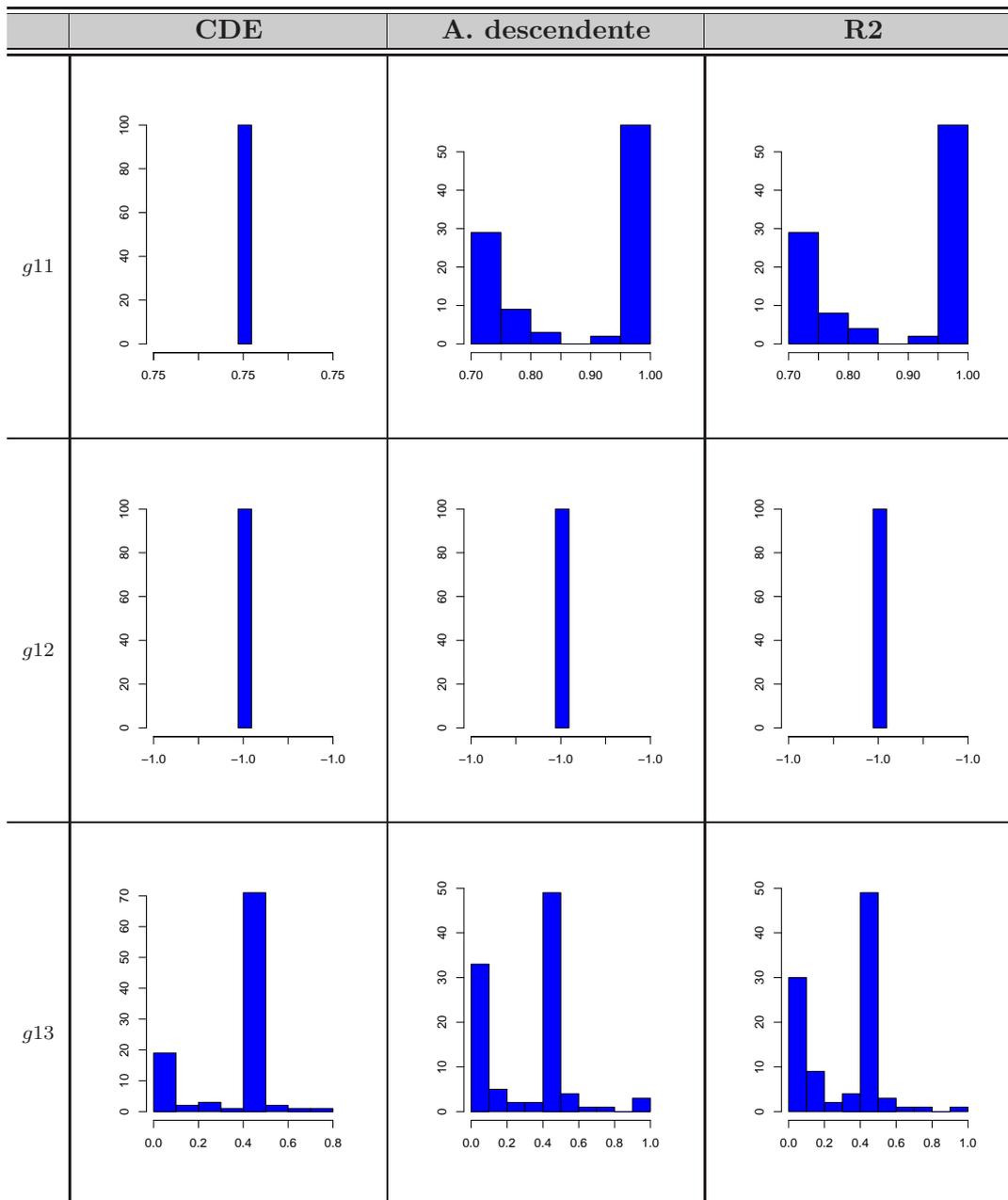
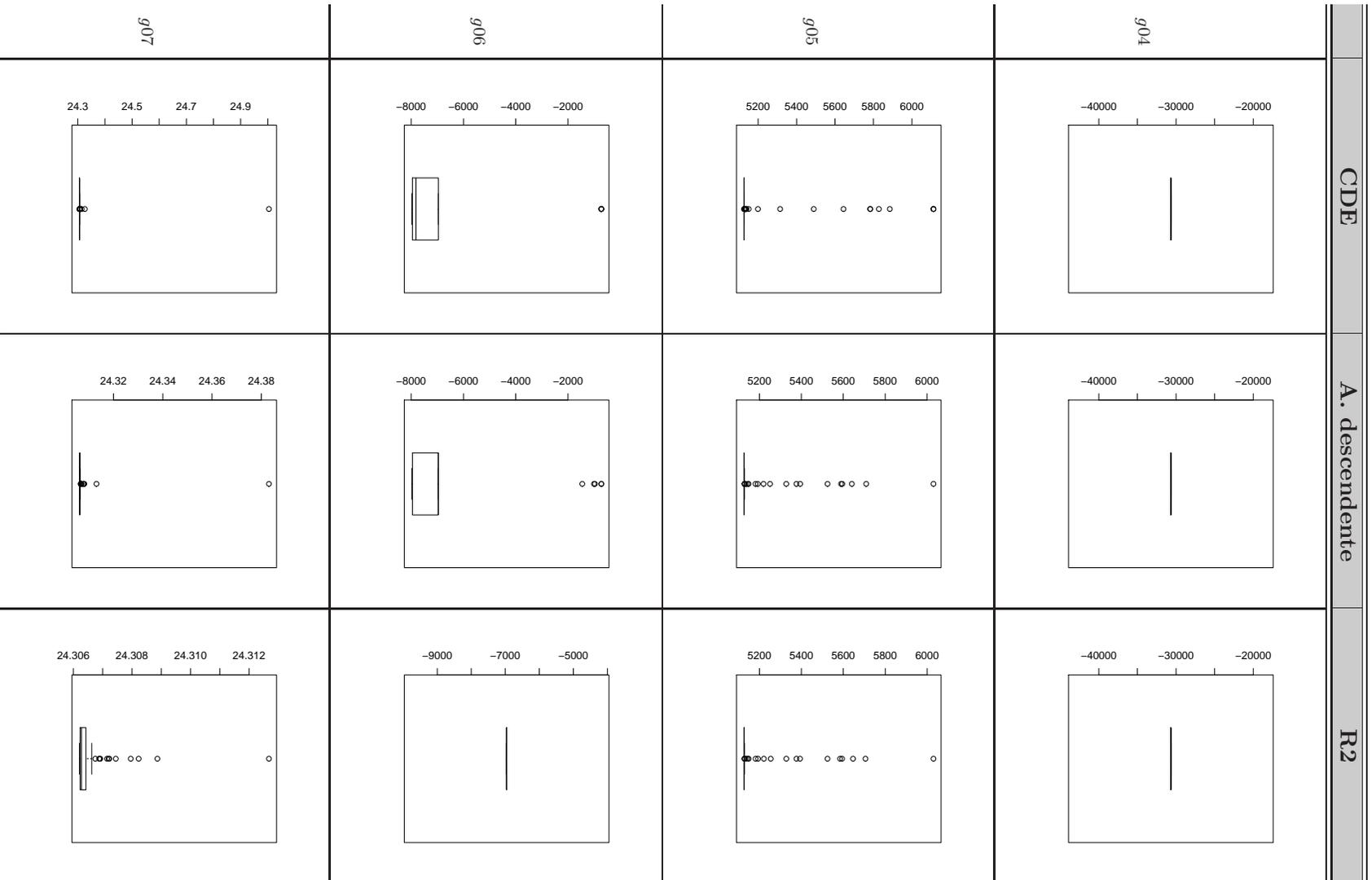
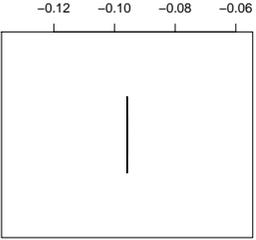
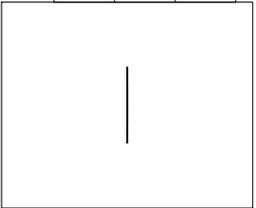
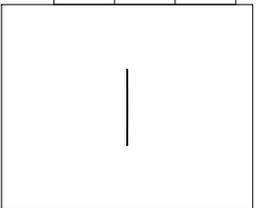
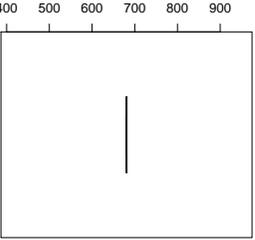
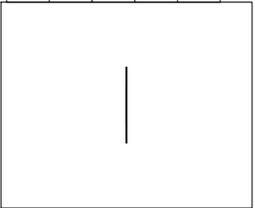
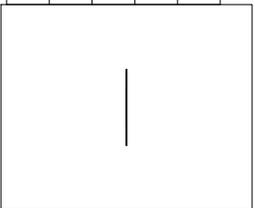
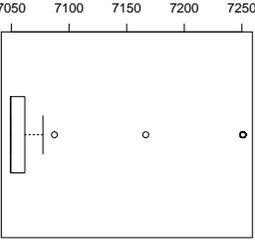
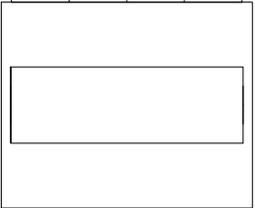
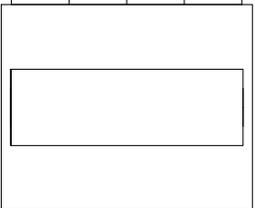
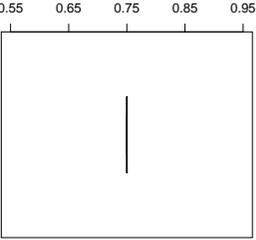
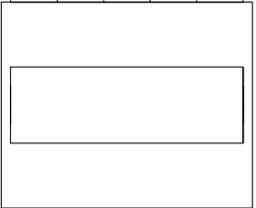
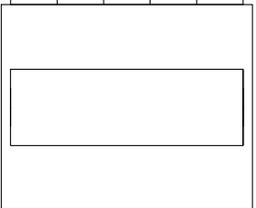


Tabla D.1: Histogramas de frecuencias de las soluciones de las 100 ejecuciones independientes para los algoritmos CDE, el mecanismo de selección aleatoria descendente y nuestra propuesta R2 como mecanismo de selección, la cual fue la que mejor desempeño tuvo.

D.2. Gráficos de caja

	CDE	A. descendente	R2
g01			
g02			
g03			



	CDE	A. descendente	R2
g_{08}			
g_{09}			
g_{10}			
g_{11}			

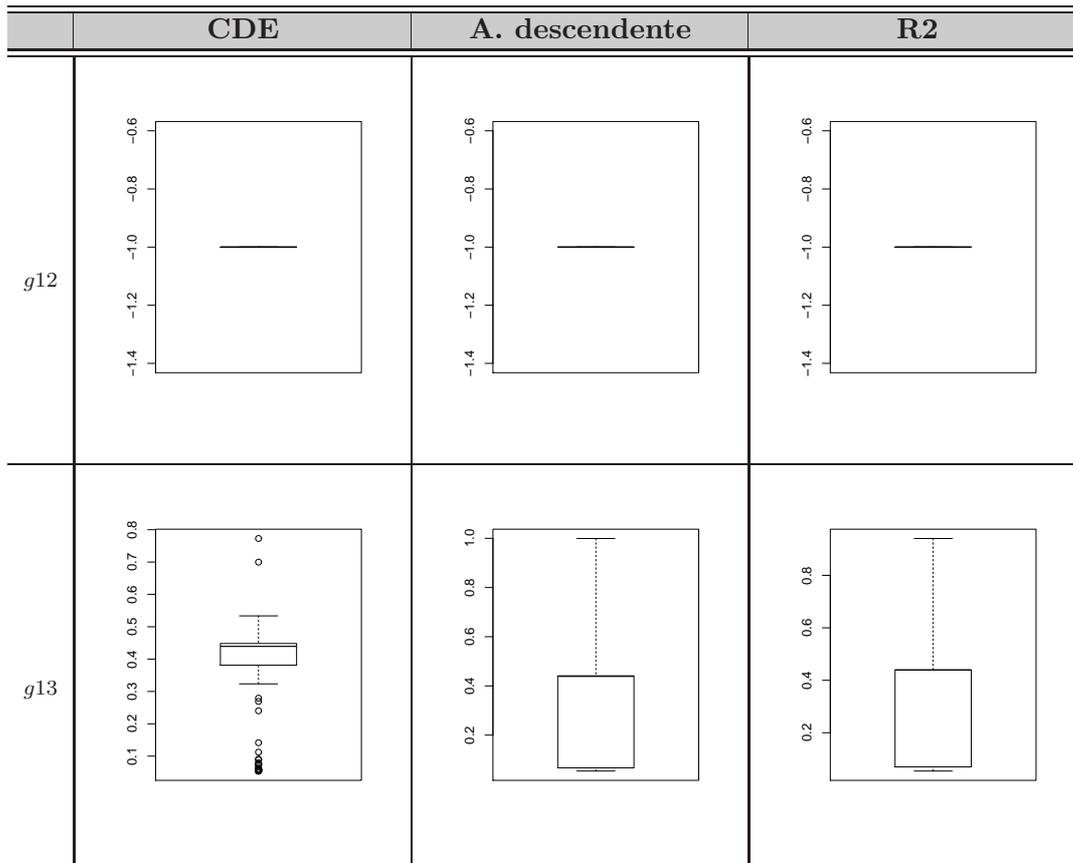


Tabla D.2: Gráficas de caja de las soluciones de las 100 ejecuciones independientes para los algoritmos CDE, el mecanismo de selección aleatoria descendente y nuestra propuesta R2 como mecanismo de selección, la cual fue la que mejor desempeño tuvo.

D.3. Comportamiento de heurísticas de bajo nivel

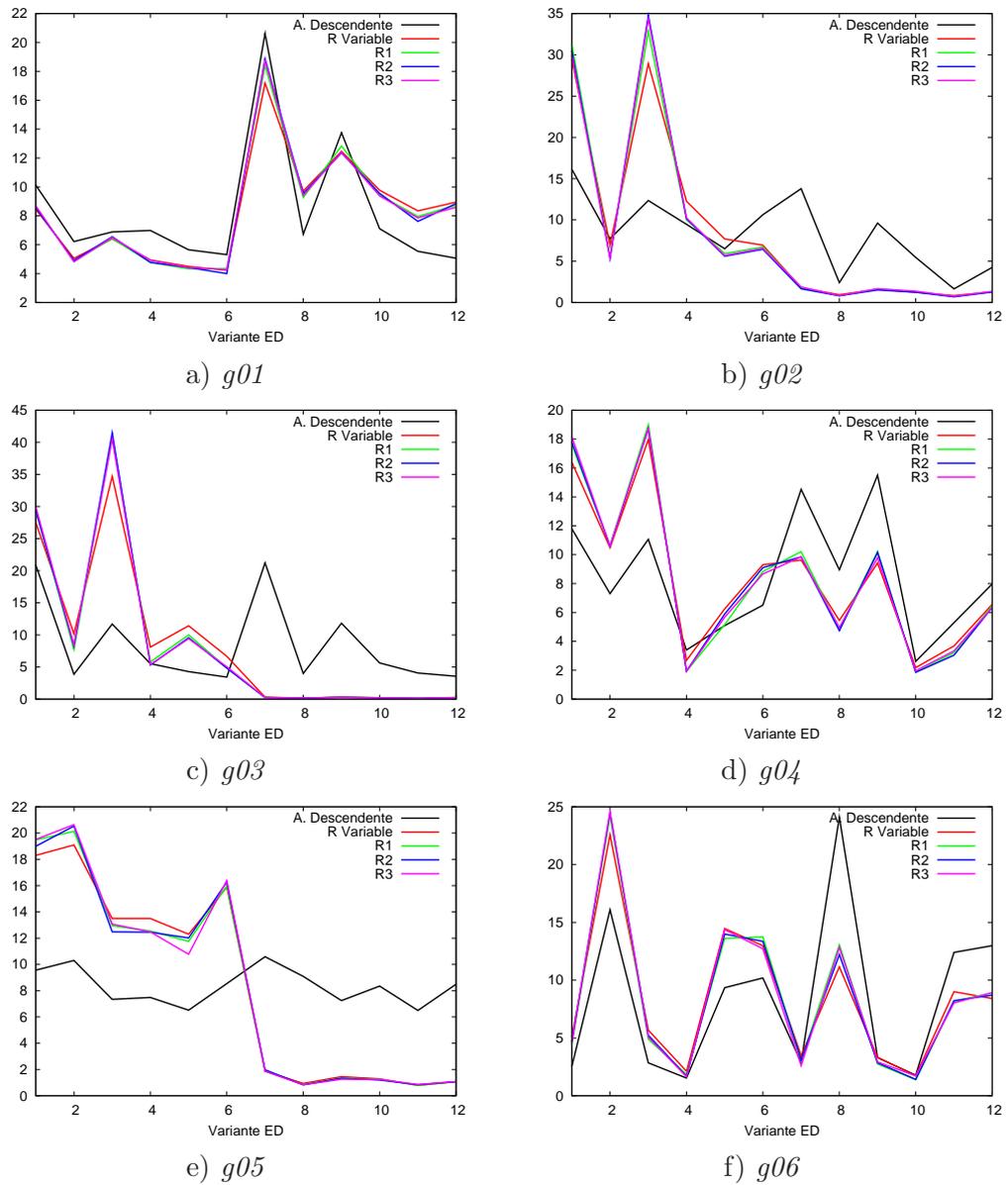


Figura D.1: Porcentaje de mejoras por cada variante para cada uno de los modelos de selección de heurísticas para las funciones $g01$ - $g06$

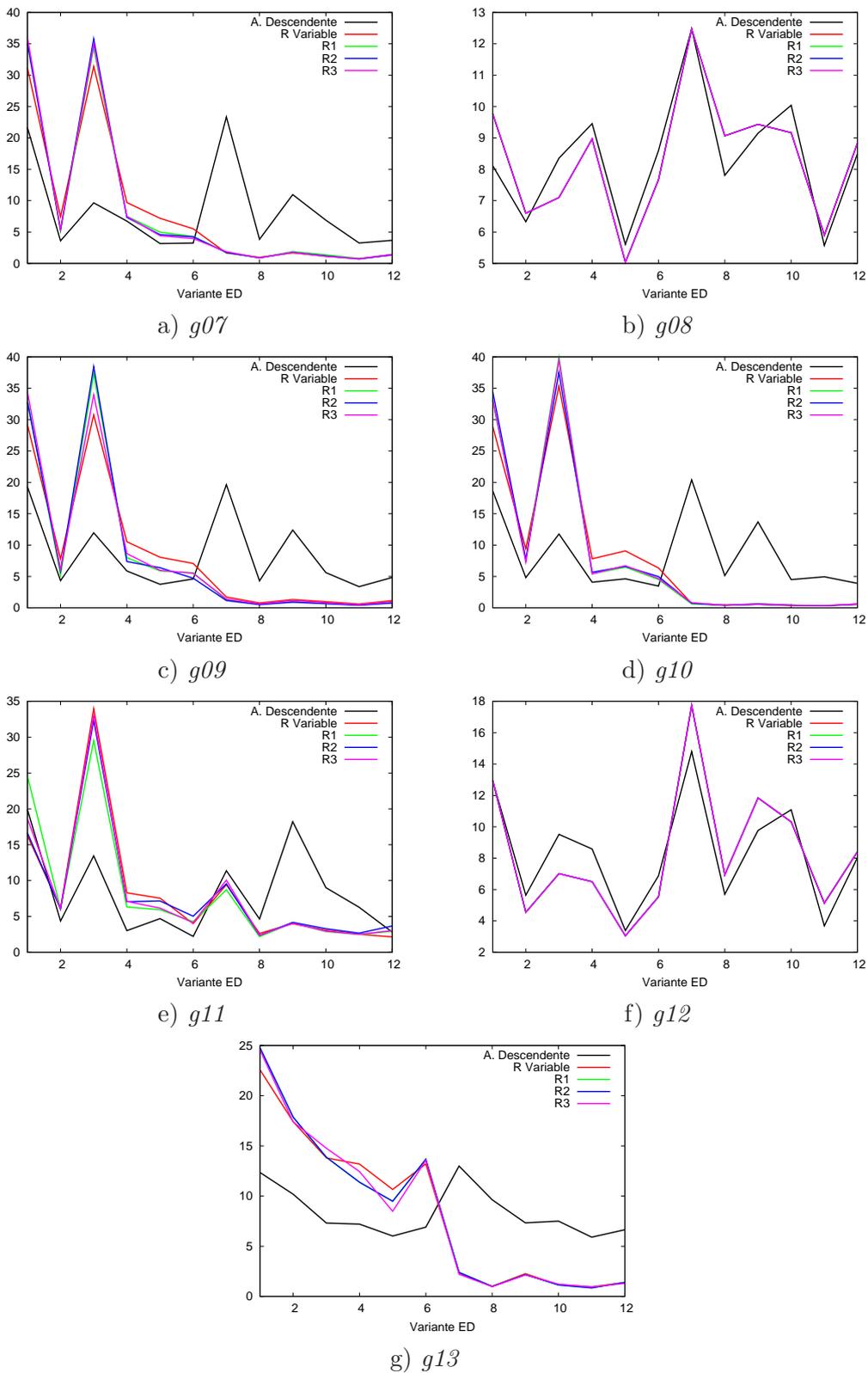
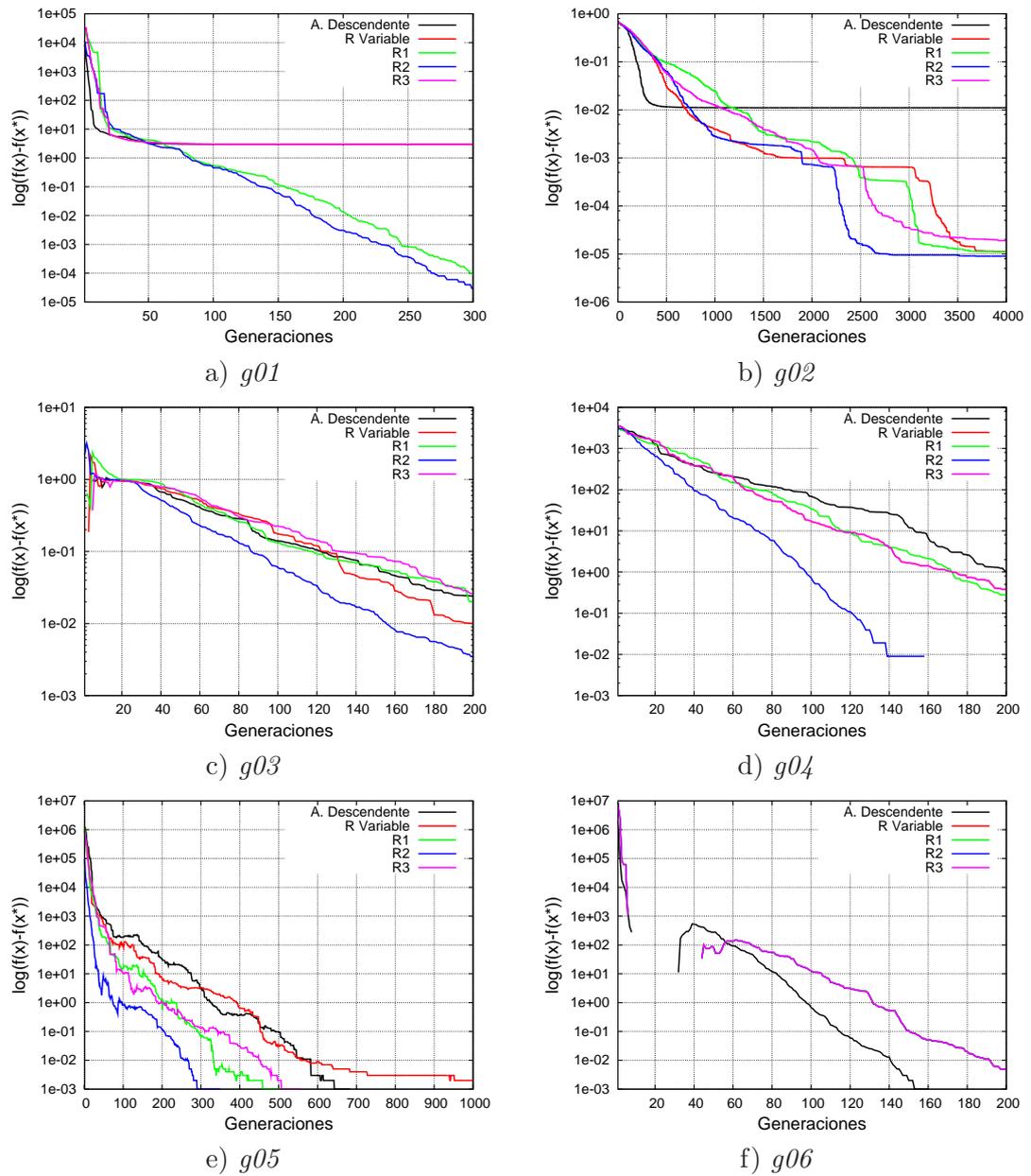


Figura D.2: Porcentaje de mejoras por cada variante para cada uno de los modelos de selección de heurísticas para las funciones $g07 - g13$

D.4. Gráficas de convergencia

Figura D.3: Gráficas de convergencia para las funciones $g_{01} - g_{06}$

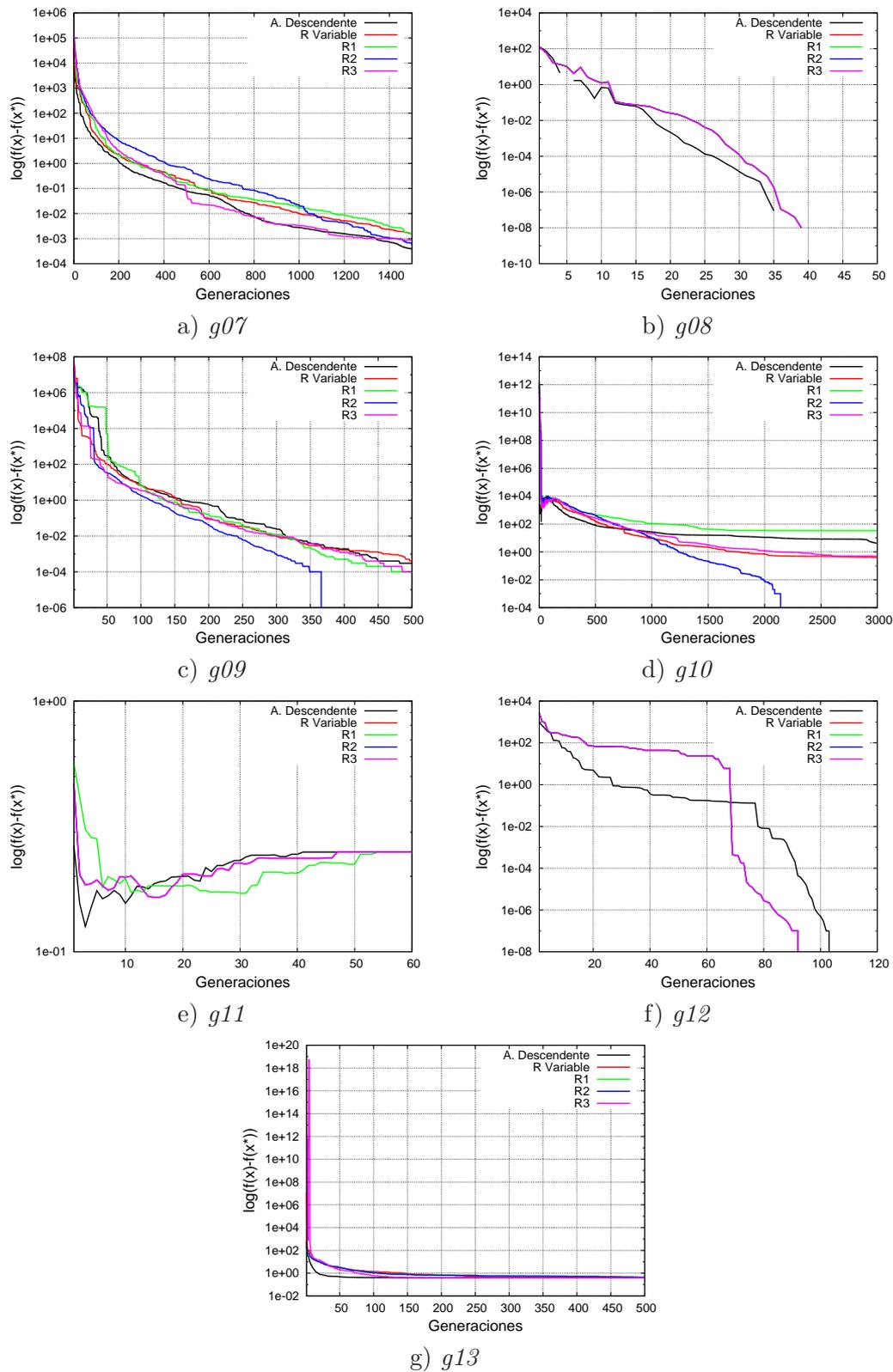


Figura D.4: Gráficas de convergencia para las funciones $g07 - g13$

Bibliografía

- [1] AARTS, E., KORST, J., AND VAN LAARHOVEN, P. Simulated annealing. In *Local Search in Combinatorial Optimisation*, E. Aarts and J. Lenstra, Eds. John Wiley & Sons, Chichester, 1997, pp. 91–120.
- [2] ADELI, H., AND CHENG, N. Augmented lagrangian genetic algorithm for structural optimization. *Journal of Aerospace Engineering* 7 7, 1 (Jan. 1994), 104–118.
- [3] AYOB, K., AND KENDALL, G. A monte carlo hyper-heuristic to optimise component placement sequencing for multi head placement machine. In *Proceedings of the International Conference on Intelligent Technologies* (Dec. 2003), pp. 132–141.
- [4] BAI, R., AND KENDALL, G. An investigation of automated planograms using a simulated annealing based hyper-heuristic. In *Proceedings of the 5th Metaheuristics International Conference (MIC2003)* (Kyoto, Japan, Aug. 2003).
- [5] BEAN, J. C., AND HADJ-ALOUANE, A. B. A dual genetic algorithm for bounded integer programs. Tech. Rep. TR 92-53, Department of Industrial and Operations Engineering, The University of Michigan, 1992.
- [6] BENJAMINI, Y. Opening the box of a boxplot. *The American Statistician* 42, 4 (1988), 257–262.
- [7] BIAZZINI, M., BÁNHÉLYI, B., MONTRESOR, A., AND JELASITY, M. Distributed hyper-heuristics for real parameter optimization. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation* (Montréal Québec, Canada, July 2009), ACM, pp. 1339–1346.
- [8] BREST, J., GREINER, S., BOSKOVIC, B., MERNIK, M., AND ZUMER, V. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation* 10, 6 (2006), 646–657.
- [9] BURKE, E., KENDALL, G., AND SOUBEIGA, E. A tabu-search hyper-heuristic for timetabling and rostering. *Journal of Heuristics* 9, 6 (Nov. 2004), 451–470.
- [10] BURKE, E., MCCOLLUM, B., MEISELS, A., AND PETROVIC, S. A graph-based hyper-heuristic for educational timetabling problems. *European Journal of Operational Research* 176, 1 (Jan. 2007), 177–192.

-
- [11] CHAKHLEVITCH, K. *A hyperheuristic methodology for real-world scheduling*. PhD thesis, University of Bradford, UK, 2006.
- [12] CHAKHLEVITCH, K., AND COWLING, P. *Hyperheuristics: Recent Developments*, vol. 136 of *Studies in Computational Intelligence*. Springer Berlin, June 2008, pp. 3–29.
- [13] COELLO COELLO, C. Theoretical and numerical constraint handling techniques used with evolutionary algorithms: A survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering* 191, 11 (Jan. 2002), 1245–1287.
- [14] COWLING, P., AND CHAKHLEVITCH, K. Hyperheuristics for managing a large collection of low level heuristics to schedule personnel. In *Proceedings of the 2003 IEEE Congress on Evolutionary Computation (CEC 2003)* (Los Alamitos, 2003), IEEE Press, pp. 1214–1221.
- [15] COWLING, P., AND CHAKHLEVITCH, K. Using a large set of low level heuristics in a hyperheuristic approach to personnel scheduling. In *Evolutionary Scheduling* (Heidelberg, 2007), K. Dahal, K. Tan, and P. Cowling, Eds., Springer.
- [16] COWLING, P., KENDALL, G., AND SOUBEIGA, E. A hyperheuristic approach to scheduling a sales summit. In *Proceedings of the Third Metaheuristic International Conference (MIC 2001)* (Porto, Portugal, 2000), vol. 2079, Springer-Verlag, pp. 127–131.
- [17] COWLING, P., KENDALL, G., AND SOUBEIGA, E. A parameter-free hyperheuristic for scheduling a sales summit. In *Proceedings of the 3rd Metaheuristic International Conference (MIC 2001)* (Porto, Portugal, 2001), pp. 127–131.
- [18] DASGUPTA, D., Ed. *Artificial Immune Systems and Their Applications*. Springer-Verlag, Berlin, 1999.
- [19] DE JONG, K. A. *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, 1975.
- [20] DEB, K. Efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering* 186 2, 4 (2000), 311–338.
- [21] DORIGO, M. *Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Italy, 1992.
- [22] DUECK, G. New optimisation heuristics: the great deluge algorithm and the record-to-record travel. *Journal of Computational Physics* 104 (1993), 86–92.
- [23] EFRON, B. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics* 7, 1 (1979), 1–26.
- [24] EFRON, B., AND TIBSHIRANI, R. J. *An Introduction to the Bootstrap*. Chapman & Hall, New York, 1993.
- [25] FAN, Z., LIU, J., SORENSEN, T., AND WANG, P. Improved differential evolution based on stochastic ranking for robust layout synthesis of mems components. *IEEE Transactions On Industrial Electronics* 56, 4 (2008), 937–948.
-

- [26] FANG, H.-L., ROSS, P., AND CORNE, D. A promising hybrid GA heuristic approach for open-shop scheduling problems. In *Proceedings of 11th European Conference on Artificial Intelligence (ECAI 1994)* (1994), A. Cohn, Ed., pp. 590–594.
- [27] FOGEL, L., OWENS, A., AND WALSH, M. *Artificial Intelligence through Simulated Evolution*. Wiley, Chichester, UK, 1966.
- [28] GLOVER, F., AND LAGUNA, M. *Tabu search*. Kluwer Academic, Norwell, 1997.
- [29] GOLDBERG, D. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison - Wesley Longman Publishing Co., Boston, MA, USA, 1989.
- [30] HADJ-ALOUANE, A. B., AND BEAN, J. C. A genetic algorithm for the multiple-choice integer program. *Operations Research* 45 (1997), 92–101.
- [31] HALTON, J. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik* 2 (Dec. 1960), 84–90.
- [32] HAMMERSLEY, J. M. Monte-carlo methods for solving multivariable problems. *Annals of the New York Academy of Science* 86 (1960), 844–874.
- [33] HINTERDING, R., AND MICHALEWICZ, Z. Your brains and my beauty: Parent matching for constrained optimisation. In *Proceedings of the 5th International Conference on Evolutionary Computation* (Anchorage, Alaska, May 1998), pp. 810–815.
- [34] HOLLAND, J. H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications in Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, Massachusetts, USA, 1992.
- [35] HOMAIFAR, A., LAI, S. H. Y., AND QI, X. Constrained optimization via genetic algorithms. *Simulation* 62 4 (1994), 242–254.
- [36] JIMÉNEZ, F., AND VERDEGAY, J. L. Evolutionary techniques for constrained optimization problems. In *7th European Congress on Intelligent Techniques and Soft Computing (EUFIT'99)* (Aachen, Germany, 1999), H.-J. Zimmermann, Ed., Verlag Mainz.
- [37] JOINES, J., AND HOUCK, C. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with ga's. In *Proceedings of the first IEEE Conference on Evolutionary Computation* (Orlando, Florida, 1994), D. Fogel, Ed., IEEE Press, pp. 579–584.
- [38] KAEHLING, L., LITTMAN, M., AND MOORE, A. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research* 4 (1996), 237–285.
- [39] KAZARLIS, S., AND PETRIDIS, V. Varying fitness functions in genetic algorithms studying in the rate increase of the dynamic penalty terms. In *Proceedings of the 5th Parallel Problem Solving from Nature (PPSN V)* (Heidelberg, Germany, Sept. 1998), A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, Eds., vol. 1498 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 211–220.

- [40] KENDALL, G., AND MOHAMAD, M. Channel assignment in cellular communication using a great deluge hyper-heuristic. In *Proceedings of the 12th IEEE International Conference on Networks* (2004), vol. 2, pp. 769–773.
- [41] KENDALL, G., AND MOHAMAD, M. Channel assignment optimisation using a hyper-heuristic. In *Proceedings of the 2004 IEEE Conference on Cybernetics and Intelligent Systems (CIS 2004)* (Singapore, Dec. 2004).
- [42] KENDALL, G., AND MOHD HUSSIN, N. An investigation of a tabu search based hyperheuristic for examination timetabling. In *Multidisciplinary Scheduling: Theory and Applications, Selected papers from the 1st Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2003)* (Heidelberg, 2005), G. Kendall, E. Burke, S. Petrovic, and M. Gendreau, Eds., Springer, pp. 309–328.
- [43] KENDALL, G., AND MOHD HUSSIN, N. Tabu search hyper-heuristic approach to the examination timetabling problem at university of technology mara. *PATAT 2004 3616* (2005), 199–217.
- [44] KOWALCZYK, R. Constraint consistet genetic algorithms. In *Proceedings of the 1997 IEEE Congress on Evolutionary Computation* (Indianapolis, USA, Apr. 1997), IEEE Press, pp. 343–348.
- [45] KURI-MORALES, A., AND QUEZADA, C. V. A universal electric genetic algorithm for constrained optimization. In *Proceedings 6th European Congress on Intelligent Techniques & Soft Computing, EUFIT'98* (Aachen, Germany, Sept. 1998), Verlag Mainz, pp. 518–522.
- [46] LAMPINEN, J. Constraint handling approach for the differential evolution algorithm. In *Proceedings of the Congress on Evolutionary Computation 2002 (CEC'2002)* (Piscataway, New Jersey, May 2002), vol. 2 of *IEEE Service Center*, pp. 1468–1473.
- [47] LIU, J., AND LAMPINEN, J. A fuzzy adaptive differential evolution algorithm. *Soft Computing - A Fusion of Foundations, Methodologies and Applications* 9, 6 (June 2005), 448.462.
- [48] MICHALEWICZ, Z. A survey of constraint handling techniques in evolutionary computation methods. In *Proceedings of the 4th Annual Conference on Evolutionary Programming* (Cambridge, Massachusetts, 1995), J. McDonell, R. Reynolds, and D. Fogel, Eds., The MIT Press, pp. 135–155.
- [49] MICHALEWICZ, Z., AND JANIKOW, C. Z. Handling constraints in genetic algorithms. In *Proceedings of the Fourth International Conference on Genetic Algorithm (ICGA-91)* (University of California, San Diego, 1991), R. K. Belew and L. B. Booker, Eds., Morgan Kaufmann Publishers, pp. 151–157.
- [50] MICHALEWICZ, Z., AND SCHOENAUER, M. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation* 4, 1 (1996), 1–32.
- [51] MONTES, E., COELLO COELLO, C., AND TUN-MORALES, E. Simple feasibility rules and differential evolution for constrained optimization. In *Proceedings of the 3rd Mexican International Conference on Artificial Intelligence (MICAI'2003)* (Apr. 2004),

- R. Monroy, G. Arroyo-Figueroa, L. Sucar, and H. Sossa, Eds., no. 2972, Springer-Verlag, pp. 707–716.
- [52] MOORE, M., AND MCCABE, G. *Introduction to the Practice of Statistics*. Freeman, New York, 2004.
- [53] NAREYEK, A. Choosing search heuristics by non-stationary reinforcement learning. In *Metaheuristics: Computer decision-making* (Dordrecht, 2003), Kluwer Academic Publishers, pp. 523–544.
- [54] NORENKOV, I. Scheduling and allocation for simulation and synthesis of cad system hardware. In *Proceedings of East-West International Conference (EWITD 1994)* (Moscow, 1994), pp. 20–24.
- [55] NUNES DE CASTRO, L., AND TIMMIS, J. *An Introduction to Artificial Immune Systems: A New Computational Intelligence Paradigm*. Springer-Verlag, 2002.
- [56] OSY CZKA, A. Multicriteria optimization for engineering design. In *Design Optimization*, J. S. Gero, Ed. Academic Press, 1985, pp. 193–227.
- [57] PAREDIS, J. Co-evolutionary constraint satisfaction. In *Proceedings of the 3rd Conference on Parallel Problem Solving from Nature* (New York, 1994), Springer-Verlag, pp. 46–55.
- [58] PRICE, K. An introduction to differential evolution. In *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds. McGraw-Hill, 1999, pp. 79–106.
- [59] RAO, S. S. *Engineering Optimization. Theory and Practice*, third ed. John Wiley & Sons, Inc., 1996.
- [60] REEVES, C., Ed. *Modern heuristic techniques for combinatorial problems*. John Wiley & Sons, Inc., New York, USA, 1993.
- [61] REEVES, C. Modern heuristic techniques. In *Modern Heuristic Search Methods* (Dec. 1996), V. Rayward-Smith, I. Osman, C. Reeves, and G. Smith, Eds., Wiley, p. 314.
- [62] RICHARDSON, J. T., PALMER, M. R., LIEPINS, G., AND HILLIARD, M. Some guidelines for genetic algorithms with penalty functions. In *Proceedings of the Third International Conference on Genetic Algorithms (ICGA-89)* (San Mateo, California, June 1989), Morgan Kaufmann Publishers, pp. 191–197.
- [63] RUNARSSON, T. P., AND YAO, X. Stochastic ranking for constrained evolutionary optimization. *Transactions On Evolutionary Computation* 4, 3 (Sept. 2000), 284–294.
- [64] RUNARSSON, T. P., AND YAO, X. Search biases in constrained evolutionary optimization. *IEEE Transactions on Systems* 35, 2 (May 2005), 233–243.
- [65] SCHWEFEL, H.-P. *Numerical Optimization of Computer Models*. John Wiley & Sons, Wiley, Chichester, UK, 1981.

- [66] SMITH, A. E., AND TATE, D. M. Genetic optimization using a penalty function. In *Proceedings of the Fifth International Conference on Genetic Algorithms* (University of Illinois at Urbana-Champaign, July 1993), S. Forrest, Ed., Morgan Kaufmann Publishers.
- [67] SOBOL', I. M. Uniformly distributed sequences with an additional uniform property. *USSR Computational Mathematics and Mathematical Physics* 16 (1976), 236–242.
- [68] STORER, R., WU, S., AND VACCARI, R. Problem and heuristic search space strategies for job shop scheduling. *ORSA Journal on Computing* 7 (1995), 453–467.
- [69] STORN, R., AND PRICE, K. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. Tech. Rep. TR-95-012, International Computer Science Institute, Mar. 1995.
- [70] STORN, R., AND PRICE, K. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization* 11 (Mar. 1997), 341–359.
- [71] SYSWERDA, G. Uniform crossover in genetic algorithms. In *Proceedings of the 9th International Conference on Genetic Algorithms* (Virginia USA, June 1989), Morgan Kaufmann Publishers, pp. 2–9.
- [72] TEO, J. Exploring dynamic self-adaptive populations in differential evolution. *Soft Computing - A Fusion of Foundations, Methodologies and Applications* 10, 8 (June 2006), 673–686.
- [73] TUKEY, J. *Exploratory Data Analysis*. Adison - Wesley Publishing Company, Reading, Massachusetts, 1977.
- [74] VELÁZQUES REYES, J. Propuesta de evolución diferencial para optimización de espacios restringidos. Master's thesis, CINVESTAV-IPN, México, D.F., Jan. 2006.
- [75] WOLPERT, D., AND MACREADY, W. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* 1, 1 (1997), 67–82.
- [76] WONG, T.-T., LUK, W.-S., AND HENG, P.-A. Sampling with hammersley and halton points. *Journal of Graphics Tools* 2, 2 (1997), 9–24.
- [77] YAO, X., LIU, Y., LIANG, K.-H., AND LIN, G. Fast evolutionary algorithms. In *Advances in evolutionary computing: theory and applications*, T. Bäck and A. E. Eiben, Eds. Springer-Verlag, New York, USA, 2003, pp. 45–94.
- [78] ZANI, G., RIANI, M., AND CORBELLINI, A. Robust bivariate boxplots and multiple outlier detection. *Computational Statistics & Data Analysis* 28 (1998), 257–270.