Universidad Veracruzana
CENTRO DE INVESTIGACIÓN EN INTELIGENCIA ARTIFICIAL

Esquema Adaptativo para el Manejo de Restricciones de Límite en Problemas de Optimización Numérica Restringida

Tesis presentada por:
Efrén Juárez Castillo

para obtener el grado de:
Doctor en en Inteligencia Artificial

Director de Tesis: ***Dr. Héctor Gabriel Acosta Mesa***
Codirector: ***Dr. Efrén Mezura Montes***

Xalapa, Veracruz *Enero de 2019*

Resumen

Los algoritmos de optimización inspirados en la naturaleza son heurísticas que basan su funcionamiento en fenómenos "inteligentes" encontrados en la naturaleza para resolver problemas de optimización. Muchos problemas de optimización están restringidos y tienen un espacio de búsqueda delimitado del cual salen algunos vectores solución cuando se aplican los operadores de variación, esto hace necesaria la aplicación de métodos para el manejo de restricciones de límite, los cuales permiten reparar los vectores no válidos.

En esta tesis se presenta un esquema adaptativo para el manejo de restricciones de límite en problemas de optimización numérica restringida. El esquema adaptativo propuesto opera en dos etapas, al inicio, cuando aún no existen soluciones factibles, se emplea un método que beneficia la exploración del espacio de búsqueda, en la segunda etapa se selecciona uno de varios métodos, de acuerdo con sus probabilidades asociadas.

Las probabilidades asociadas a los métodos se actualizan cada cierto período de aprendizaje, de modo que los métodos que generan las mejores soluciones reparadas tendrán una mayor posibilidad de ser aplicados.

El esquema propuesto ha sido probado dentro de dos de los principales algoritmos de optimización inspirados en la naturaleza: Optimización por Cúmulos de Partículas y Evolución Diferencial, empleando en ambos casos tanto su versión canónica, así como una versión del estado del arte, especializada en optimización restringida.

En las pruebas realizadas se comparó el rendimiento del esquema adaptativo contra seis de los principales métodos para el manejo de restricciones de límite, resolviendo un amplio conjunto de problemas de optimización numérica restringida.

Los resultados muestran que este esquema adaptativo tiene un gran impacto en el rendimiento del algoritmo, y es capaz de promover mejores resultados finales principalmente dentro de problemas de alta dimensionalidad.

Agradecimientos

A todo el personal Docente y Administrativo del Centro de Investigación en Inteligencia Artificial de la Universidad Veracruzana, gracias por recibirme en el seno de tan agradable familia.
A mis compañeros de la generación 2013-2017 del Doctorado en Inteligencia Artificial, gracias por enriquecer mis pensamientos con sus muy acertados puntos de vista.
Un agradecimiento especial a mis asesores: Dr. Héctor Gabriel Acosta Mesa y Dr. Efrén Mezura Montes, les estoy profundamente agradecido por todo su apoyo, sin ustedes, esto no habría sido posible.
A mi familia, muchas gracias por todo su apoyo.

A mi querida esposa Lulú con todo mi cariño, gracias por tu apoyo.

Índice general

Capítulo 1

Introducción

En nuestra vida diaria realizamos actividades para las cuales es común que tratemos de buscar una mejor manera de ejecutarlas, un ejemplo puede ser nuestra rutina matutina, misma que realizamos todos los días antes de ir al trabajo y que si cambiáramos el orden de alguna u otra tarea podríamos ver algún beneficio como reducir el tiempo de nuestra rutina o realizar cierta tarea de una mejor forma. Otra actividad que es común que busquemos mejorar es la ruta a nuestro trabajo, ya sea para disminuir el tiempo, el combustible o la distancia del recorrido.

Cuando buscamos realizar una actividad o un proceso de una mejor manera lo que estamos haciendo es optimizar el proceso. Se dice que se ha optimizado algo (una actividad, un método, un proceso, un sistema, etc.) cuando se han efectuado modificaciones en la forma usual de proceder y se han obtenido resultados que están por encima de lo regular o lo esperado.

En este sentido, optimizar es un proceso para encontrar un resultado que maximice o minimice un objetivo para resolver un problema. Comúnmente, el problema a optimizar se representa como una función objetivo para la cual se busca un vector solución que maximice o minimice dicha función. Las componentes del vector solución, conocidas como variables de diseño, toman valores entre un límite inferior y otro superior, a estos límites se les conoce como restricciones de límite y definen el espacio en el cual se debe buscar una solución válida, es decir, el espacio de búsqueda.

Adicionalmente, en la mayoría de los problemas o aplicaciones de optimización de la vida real se requiere encontrar un vector solución que optimice la función objetivo y además pueda satisfacer un conjunto de restricciones de igualdad o desigualdad, mismas que también se representan como funciones, a este tipo de restricciones se les conoce como restricciones funcionales y definen la región factible dentro del espacio de búsqueda. A los problemas de optimización con restricciones funcionales se les conoce como problemas de optimización restringida.

De manera intuitiva, para encontrar una solución óptima a un problema restringido se podrían probar todas las posibles soluciones, es decir, aquellas que se encuentren dentro del espacio de

búsqueda y quedarnos con aquella que nos proporcione los mejores resultados y a la vez se encuentre dentro de la región factible. Sin embargo, esto no siempre es posible, pues existen problemas para los cuales, si se tuvieran que evaluar todas las posibles soluciones, no se podría garantizar encontrar una solución óptima en un tiempo razonable, pues la cantidad de posibles soluciones es enorme [54].

Existen varios métodos de programación matemática; como el multiplicador de lagrange, la programación cuadrática y el método de proyección de gradiente [54]; que pueden emplearse en la resolución de problemas de optimización numérica restringida, sin embargo, estos métodos no garantizan llegar a soluciones optimas (o cuasi-óptimas) en todo tipo de problemas de optimización restringida en un tiempo razonable [28].

Otro enfoque que ha mostrado buenos resultados en la resolución de problemas de optimización ha sido el empleo de algoritmos de optimización inspirados en la naturaleza, los cuales permiten buscar soluciones óptimas empleando estrategias de búsqueda, como podría ser evaluar sólo una parte de las soluciones y no todo el conjunto de ellas [12]. Aún cuando no siempre se garantice encontrar la mejor opción, este tipo de algoritmos son capaces de encontrar soluciones aproximadas, con un tiempo y consumo de recursos aceptables [28].

Un algoritmo es un conjunto ordenado de operaciones inequívocas y efectivamente computables que cuando se ejecutan producen un resultado y se detiene en una cantidad de tiempo finita [19]. En el caso de los algoritmos inspirados en la naturaleza, estas operaciones imitan los procesos naturales para resolver problemas complejos de optimización (ya sean problemas combinatorios o de optimización continua).

La forma de operación de un algoritmo inspirado en la naturaleza es realizando una búsqueda continua, mediante operadores de variación, por el espacio de búsqueda. En esta búsqueda se trata de mantener el equilibrio entre diversificación e intensificación. Lo primero se refiere a la exploración de nuevas regiones del espacio de búsqueda, mientras que lo segundo a la explotación de alguna región concreta [15] . La existencia de este balance, identifica rápidamente las regiones prometedoras del espacio de búsqueda y evita el consumo de tiempo en las regiones que ya han sido exploradas o que no contienen soluciones de alta calidad.

Los algoritmos de optimización inspirados en la naturaleza se diseñaron originalmente para tratar espacios de búsqueda no restringidos, por lo tanto, para resolver un problema restringido, se debe emplear alguna técnica apropiada que permita tratar con las restricciones funcionales mediante la incorporación de información de factibilidad en el sesgo de búsqueda [28].

La literatura especializada en este tipo de algoritmos ha mostrado avances significativos en técnicas para el manejo de restricciones funcionales[9], en contraste, un aspecto que ha sido menos explorado es la forma de lidiar con las restricciones de límite [22], es decir, los métodos que permiten

mantener al algoritmo buscando dentro de los rangos válidos definidos por cada variable de diseño en un espacio continuo.
Sin embargo, diversas investigaciones [21, 64] han reportado que el método para el manejo de restricciones de límite empleado en los algoritmos de optimización como la Evolución Diferencial (DE) y Optimización por Cúmulos de Partículas (PSO) tiene un efecto en el rendimiento general del algoritmo ya que sus operadores de variación no garantizan que el nuevo valor se encuentre dentro de los límites definidos para cada variable de diseño.
Adicionalmente, los métodos para el manejo de restricciones de límite han sido diseñados únicamente para reposicionar las soluciones que salen del espacio de búsqueda en posiciones cercanas al lugar donde se produce la violación del límite, en ningún momento se intenta incorporar la información de factibilidad a fin de reposicionar de una mejor manera las soluciones que salen del espacio de búsqueda, o emplear algún mecanismo que permita obtener mejores resultados aprendiendo del problema que se intenta resolver.
Estas limitaciones nos han motivado a proponer un esquema adaptativo para el manejo de restricciones de límite. Este esquema trabaja en dos etapas, en las primeras generaciones, cuando no existen soluciones factibles, mantiene un enfoque de diversificación beneficiando la exploración de nuevas regiones del espacio de búsqueda, en la segunda etapa, emplea un esquema de aprendizaje que le permite seleccionar de entre varios métodos, aquel que permita obtener mejores resultados al momento de reparar las soluciones que salen del espacio de búsqueda.
De manera adicional, en el presente trabajo también presentamos un nuevo método basado en centroide y una modificación a un método basado en remuestreo. El método basado en centroide incorpora información de factibilidad en el manejo de restricciones de límite y el método basado en remuestreo presenta una mejora a un método existente que beneficia la exploración del espacio de búsqueda; ambos métodos pueden ser empleados dentro del propio esquema adaptativo propuesto.
A continuación, en el presente capítulo se plantea la problemática, la justificación, la hipótesis que ha guiado la presente investigación, el objetivo general, los objetivos específicos, la lista de publicaciones científicas que se generaron como producto de ésta investigación y la organización capitular del presente trabajo.

1.1. Planteamiento del problema

Los Problemas de Optimización Numérica Restringida se presentan en una amplia gama de problemas científicos y de ingeniería cuyo propósito es determinar los valores para un conjunto de variables de decisión, que componen un vector solución, optimizando una función objetivo

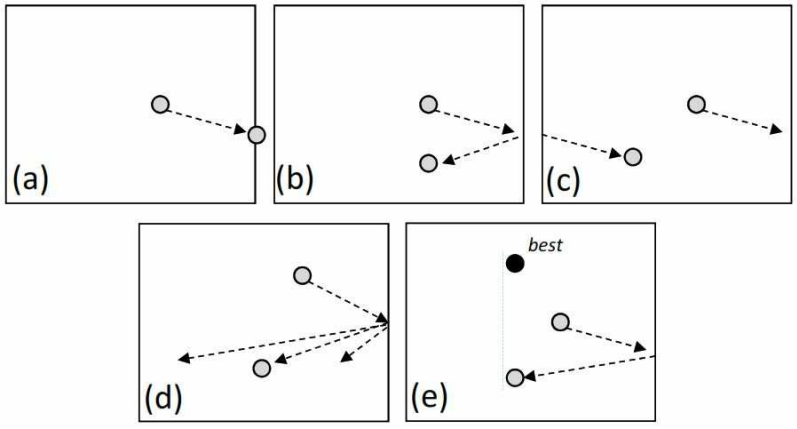


Figura 1.1: Mecanismos utilizados comúnmente por los métodos para el manejo de restricciones de límite.

mientras se satisfacen restricciones funcionales y de límite. Este tipo de problemas pueden tratarse mediante algoritmos inspirados en la naturaleza, tal como los algoritmos evolutivos [53] o los algoritmos de inteligencia colectiva [28].

Las *Restricciones de Límite* restringen los valores que pueden tomar las variables de decisión. Cada variable de decisión (componentes de todo vector solución) está restringida mediante un límite inferior y otro superior, los cuales definen en su conjunto el espacio de búsqueda [37].

Los algoritmos de optimización inspirados en la naturaleza comúnmente emplean operadores de variación, como la cruza o mutación, para explorar el espacio de búsqueda con el fin de encontrar los valores óptimos para la función objetivo. En ocasiones, estos operadores de variación pueden hacer que algunas de las variables de decisión tomen valores que están fuera de sus límites superior e inferior, generando vectores solución no válidos, es decir, soluciones ubicadas fuera del espacio de búsqueda.

Cuando algún vector solución abandona el espacio de búsqueda, es necesario aplicar algún método para el manejo de restricciones de límite (*Boundary Constraint-Handling Methods, BCHM*) con la finalidad de corregir el vector inválido [21].

La función principal de los BCHMs consiste en aplicar mecanismos para controlar que la búsqueda se realice dentro del espacio de búsqueda, ya sea evitando que las soluciones salgan del espacio de búsqueda, reubicando las que salen o asegurándose que vuelvan a reintroducirse en futuras generaciones.

En la literatura especializada para el manejo de restricciones de límite se observa que en los principales BCHMs (ver Figura 1.1), las variables de decisión que violan los límites se restablecen al

límite violado (a) [4], en otros casos, los valores son reflejados desde el límite violado (b) [46] o desde el lado opuesto al límite violado (c) [64] por la cantidad de violaciones. En otros métodos, la variable es reflejada una cantidad aleatoria (d) entre los límites inferior y superior [43] o entre el límite violado y algún límite establecido por el mejor vector solución (e) de la población actual [18].

Esta variedad de mecanismos empleados en el manejo de restricciones de límite nos presenta de manera natural dos preguntas: ¿El BCHM empleado tiene alguna influencia en la calidad de las soluciones encontradas por el algoritmo de optimización?, en cuyo caso sería necesario seleccionar el BCHM más adecuado al problema a tratar, lo cual nos lleva naturalmente a la segunda pregunta: dado un conjunto de problemas de optimización en específico, ¿existe algún BCHM que muestre un rendimiento promedio mejor que el de sus competidores?, y de ser así ¿cuál es? o ¿qué características debería tener tal BCHM?.

1.2. Justificación

El manejo de las restricciones de límite adquiere una mayor relevancia al observar la cantidad de vectores solución que salen del espacio de búsqueda, en este sentido, datos reportados en un estudio previo en donde se empleó como algoritmo de optimización DE, junto con 8 diferentes BCHMs, en 36 problemas de optimización restringida en 10 y 30 dimensiones, indican que en promedio un 44 % de los vectores solución, salieron del espacio de búsqueda [26].

En otro estudio, donde se compararon también 6 diferentes BCHM en DE con problemas no restringidos en 10 y 30 dimensiones se observó que el 41 % de los vectores solución fueron reparados [2]. Estos números son de importancia, pues prácticamente en 4 de cada 10 vectores solución influye más el BCHM que el algoritmo de optimización.

De este tipo de estudios, que indican la notable cantidad de soluciones reparadas, tambien se ha podido observar que la calidad de las soluciones encontradas por el algoritmo de optimización está influenciada[1] significativamente por el BCHM empleado [2, 18, 20, 24, 61], dando pie a considerar la elección del BCHM como una posibilidad para mejorar el rendimiento del algoritmo de optimización, siendo esta una función secundaria de los BCHMs.

En cuanto a las particularidades de los BCHMs, se ha observado que algunos de ellos muestran un mejor rendimiento en cierto tipo de problemas, por ejemplo, se obtiene un mejor rendimiento si se

[1] La influencia se ha observado en la calidad de las soluciones reparadas, se pueden obtener soluciones reparadas de mejor calidad con algún método que con otro. Por ejemplo, si el óptimo se encuentra cerca de un límite, será común que las soluciones abandonen el espacio de búsqueda por ese límite en su intento de dirigirse hacia esa región; en este caso, si se repara con un método que ubique las soluciones reparadas en el límite se obtendrán soluciones reparadas de buena calidad, y si se repararan con un método que refleje las soluciones por el lado opuesto seguramente las soluciones no serán de buena calidad [26].

emplea algún BCHM que corrija las variables de decisión reestableciéndolas al límite violado en problemas donde el óptimo se encuentra cerca de los límites, que si se empleara algún BCHM que corrija las variables de decisión de forma aleatoria entre sus límites inferior y superior [33].
Sin embargo, es imposible conocer a priori la ubicación de las soluciones óptimas, pues precisamente ésta es la tarea del algoritmo de optimización, por tanto, no se dispone de esta información a priori para determinar cuál BCHM es el más apropiado a cierto problema.
Por lo tanto, para elegir el BCHM más adecuado para un problema en particular se requerirían numerosas ejecuciones de prueba y error, este enfoque tampoco es aceptable pues requeriría requisitos computacionales poco realistas, en particular si la función objetivo es computacionalmente costosa o si se requieren soluciones en tiempo real. Ante este escenario, comúnmente se elige alguno de los BCHMs que ofrecen un rendimiento promedio, evitando con ello y en consecuencia, que se emplee el método más adecuando al problema a resolver.
Según el teorema de *No Free Lunch* (NFL), para cualesquiera dos algoritmos A y B, si A se desempeña mejor que B para algunos problemas, debe haber algunos problemas en los que B se desempeñará mejor que A. Es decir, si se mide sobre todos los posibles problemas, el promedio del rendimiento de ambos algoritmos es esencialmente equivalente[58].
En otras palabras, todos los algoritmos de optimización darán el mismo rendimiento promedio cuando se promedien todas las funciones posibles, por tanto, no existe un mejor método universal para todos los problemas de optimización, lo que también implica que cualquier algoritmo es tan bueno (o malo) como una búsqueda aleatoria.
Esto podría indicar que no hay necesidad de diseñar nuevos algoritmos ya que a final de cuentas todos ellos tendrán el mismo rendimiento, pero no necesariamente es así, en principio, esto no significa que todos los algoritmos tendrán el mismo rendimiento promedio sobre un conjunto de problemas en específico, por otro lado, los investigadores originales del NFL también han demostrado que existen almuerzos gratuitos para algoritmos coevolutivos, en donde su enfoque consiste en un conjunto de jugadores que compiten para producir un campeón, mismo que después será enfrentado a un contrincante [59].
Por lo anteriormente expuesto, si ningún método para el manejo de restricciones de límite, por sí sólo, podría superar el rendimiento de todos los restantes en todo tipo de problemas, al menos sí podría esperarse que un enfoque coevolutivo pudiera mostrar un mejor rendimiento, no sobre todos pero si sobre un conjunto determinado de problemas de optimización.
Por último, ninguno de los BCHMs comúnmente empleados fue diseñado para aprovechar las características propias de los problemas de optimización numérica restringida, en donde existe una región factible y las soluciones buscadas se encuentran precisamente dentro de esta región.
Lo anteriormente expuesto justifica el desarrollo de algún BCHM que se adapte a diversos tipos de

problemas de optimización numérica continua con restricciones funcionales. Este esquema adaptativo puede plantearse desde la teoría del aprendizaje por refuerzo, en donde existe un agente que determina que acción debe escoger en un entorno dado con el fin de maximizar alguna opción de recompensa, en este sentido, el agente sería el esquema adaptativo y las posibles acciones serían los distintos BCHM que podría emplear para corregir un vector inválido, posteriormente, con base en la evaluación del vector corregido, se podría saber si éste es de buena calidad, en cuyo caso el esquema adaptativo recibiría una recompensa asociada al BCHM empleado (la acción).
Este esquema adaptativo seguramente apoyaría al algoritmo de optimización a obtener soluciones de mejor calidad, empleando también información de factibilidad [2], además de cumplir su función principal que consiste en controlar que la búsqueda se realice dentro del espacio de búsqueda.

1.3. Hipótesis

Como se ha mencionado en el apartado anterior, debido a la importancia que reviste el manejo de restricciones de límite, y siendo que un enfoque coevolutivo podría reportar mejores resultados, en lugar de emplear un sólo BCHM, se propone hacer uso de un conjunto de ellos, planteando la siguiente hipótesis:
Si se cuenta con un conjunto de métodos para el manejo de restricciones de límite (BCHM) y algún mecanismo adaptable que permita aplicar aquel BCHM que genere las mejores soluciones, dependiendo del problema y de la información de factibilidad, se mejorará tanto la cantidad como la calidad de las soluciones factibles encontradas por el algoritmo de optimización.

1.4. Objetivo general

El objetivo general de la presente investigación consiste en desarrollar un esquema adaptativo para el manejo de restricciones de límite en problemas de optimización numérica restringida.
Dicho esquema hará uso de información de factibilidad y de un conjunto de BCHMs de los cuales aprenderá, durante el proceso evolutivo, cuál de ellos es el más adecuando al problema a resolver, con la finalidad de que dicho esquema permita que el algoritmo de optimización genere soluciones factibles de mejor calidad a las obtenidas mediante los métodos comúnmente empleados en el manejo de restricciones de límite.

[2] Información relacionada con la región factible, en este caso puede ser de importancia emplear la posición de las primeras soluciones factibles.

1.4.1. Objetivos específicos

- Realizar una investigación sobre los principales métodos para el manejo de restricciones de límite empleados en algoritmos de optimización numérica inspirados en la naturaleza.

- Determinar, mediante experimentación, el conjunto de métodos para el manejo de restricciones de límite que serán empleados dentro del esquema adaptativo, considerando la cantidad y calidad de las soluciones factibles alcanzadas.

- Determinar la forma en que será empleada la información de factibilidad dentro del esquema adaptativo. Para esto, será necesario considerar las primeras soluciones factibles, pues estas indicarán la región factible.

- Diseñar el mecanismo que permitirá al esquema adaptativo aprender durante el proceso de optimización y elegir al BCHM más adecuado al problema a resolver.

- Seleccionar y codificar dos de los principales algoritmos de optimización numérica inspirados en la naturaleza, uno evolutivo y otro de inteligencia colectiva, ambos en su versión canónica y una versión del estado del arte especializada en problemas de optimización numérica restringida, con base en los cuales se desarrollará la etapa de experimentación [3].

- Identificar y codificar los principales métodos para el manejo de restricciones de límite, empleados en los algoritmos de optimización numérica seleccionados en el punto anterior, contra los cuales se comparará el rendimiento del esquema adaptativo propuesto en el presente trabajo.

- Identificar un conjunto de problemas de optimización numérica restringida que contenga un variado tipo de funciones de diferentes dimensionalidades y con restricciones igualdad y desigualdad, mismo que será empleado en la etapa de experimentación.

- Realizar un estudio comparativo entre el esquema adaptativo propuesto en el presente trabajo y los principales métodos empleados en el manejo de restricciones de límite.

[3] Debido a que los algoritmos de optimización inspirados en la naturaleza se clasifican principalmente en algoritmos evolutivos y de inteligencia colectiva, se considera que es suficiente con realizar la etapa experimental con el principal algoritmo de cada categoría. Por otro lado, es importante realizar pruebas con la versión canónica de cada algoritmo para ver el impacto que tendrá el esquema adaptativo en el algoritmo original; el empleo de una versión del estado del arte, especializada en problemas de optimización numérica restringida, nos permitirá observar si este impacto sigue manifestándose aún con los mecanismos para el manejo de restricciones funcionales, que implementan estos algoritmos.

1.5. Publicaciones

En el transcurso del doctorado y como resultado de la presente investigación, he publicado cuatro artículos científicos, los cuales se mencionan a continuación y se incluyen en el Anexo C, en su versión extensa. Los dos primeros artículos fueron publicados en revistas científicas con factor de impacto y los dos últimos se publicaron en congresos internacionales y están indizados en Scopus. El primer artículo contiene la aportación principal del presente trabajo de investigación, los siguientes tres contienen aportaciones secundarias, producidas durante el mismo proceso de investigación doctoral.

1. Juárez-Castillo, E., Acosta-Mesa, H. G., & Mezura-Montes, E. (2018). Adaptive boundary constraint-handling scheme for constrained optimization. Soft Computing, 1-34.

2. Juárez-Castillo, E., Pérez-Castro, N., & Mezura-Montes, E. (2017). An Improved Centroid-Based Boundary Constraint-Handling Method in Differential Evolution for Constrained Optimization. International Journal of Pattern Recognition and Artificial Intelligence, 31(11), 1759023.

3. Juárez-Castillo, E., Acosta-Mesa, H. G., & Mezura-Montes, E. (2017, June). Empirical study of bound constraint-handling methods in Particle Swarm Optimization for constrained search spaces. In Evolutionary Computation (CEC), 2017 IEEE Congress on (pp. 604-611). IEEE.

4. Juárez-Castillo, E., Pérez-Castro, N., & Mezura-Montes, E. (2015, May). A novel boundary constraint-handling technique for constrained numerical optimization problems. In Evolutionary Computation (CEC), 2015 IEEE Congress on (pp. 2034-2041). IEEE.

1.6. Organización del documento

El presente trabajo está organizado en ocho capítulos y tres apéndices. En el Capítulo 1 se presenta la introducción junto con el planteamiento del problema, la hipótesis, los objetivos y las publicaciones científicas generadas a partir del presente trabajo de tesis. En el Capítulo 2 se presentan los conceptos básicos de optimización, los métodos empleados en la solución de tales problemas, se describen los algoritmos de optimización inspirados en la naturaleza y los métodos para el manejo de restricciones funcionales. En el Capítulo 3 se detalla la investigación sobre los métodos para el manejo de restricciones de límite, así como el funcionamiento de los métodos empleados en el manejo de posiciones y velocidades de los vectores solución que salen del espacio de búsqueda. En el Capítulo 4 se describen las contribuciones presentadas en la presente tesis,

la primera de ellas, y la principal, describe el funcionamiento del *Esquema Adaptativo para el Manejo de Restricciones de Límite*, posteriormente se describen dos contribuciones secundarias, una basada en centroide y otra basada en remuestreo. En el Capítulo 5 se presentan los materiales y métodos, en el Capítulo 6 se presentan resultados de experimentos preliminares, en el Capítulo 7 se presentan los resultados del experimento principal, en el cual se compara el rendimiento del esquema adaptativo contra el de los métodos comúnmente empleados en el manejo de restricciones de límite. Finalmente, en el Capítulo 8 se presentan las conclusiones del presente trabajo de tesis y los posibles trabajos a futuro.

Capítulo 2

Conceptos básicos

En este capítulo se aborda el concepto de optimización numérica restringida, así como dos de los principales algoritmos de optimización inspirados en la naturaleza: Optimización por Cúmulos de Partículas y Evolución Diferencial. Se presentan también, al final del capítulo, algunos métodos para el manejo de restricciones funcionales.

2.1. Optimización numérica restringida

Los Problemas de Optimización Numérica Restringida usualmente se plantean como un problema de programación no lineal [10], definido de la siguiente manera:

$$\text{Minimizar: } f(\vec{X}) \tag{2.1}$$

Sujeta a:

$$l_i \leq X_i \leq u_i \tag{2.2}$$

$$g_j(\vec{X}) \leq 0,\ j = 1,...,J \tag{2.3}$$

$$h_k(\vec{X}) = 0,\ k = 1,...,K \tag{2.4}$$

En donde $\vec{X} = [X_1, X_2, ..., X_n]^T \in \mathbb{R}^n$ representa un vector solución y $f(\vec{X})$ representa una función objetivo.

La Ecuación 2.2 representa las *Restricciones de Límite*, las cuales restringen los valores que pueden tomar la variables de decisión que componen el vector solución, en donde cada variable X_i en el vector $\vec{X}$ está restringida mediante un límite inferior y otro superior, los cuales definen en su conjunto el espacio de búsqueda $\mathscr{S}$ [37].

El objetivo del presente trabajo se centra precisamente en éstas *Restricciones de Límite*, por tal

razón el Capítulo 3 (siguiente capítulo) está dedicado exclusivamente a las restricciones de limite.
Existe también otro tipo de restricciones llamadas *Restricciones Funcionales* estas restricciones están representadas por las Ecuaciones 2.3 y 2.4, en donde $g_j(\vec{X})$ representa a la j-ésima restricción de desigualdad y $h_k(\vec{X})$ representa la k-ésima restricción de igualdad [30]. Las restricciones de igualdad comúnmente son transformadas en restricciones de desigualdad de la forma: $|h_k(\vec{X})| - \varepsilon \leq 0$, en donde ε es un pequeño valor de tolerancia permitido [48].
La región factible es denotada como $\mathscr{F}$ y representa al conjunto de todas las soluciones que satisfacen las restricciones de límite y las restricciones funcionales dentro de $\mathscr{S}$.
Existen varios métodos matemáticos tradicionales para resolver problemas de optimización numérica restringida como el Multiplicador de Lagrange para resolver problemas con restricciones de igualdad, el multiplicador de Lagrange aumentado para restricciones de desigualdad, la programación cuadrática y el método de proyección de gradiente entre otros [54].
Los métodos tradicionales siempre deben considerarse como la primera opción para resolver un problema de optimización ya que en ciertos casos pueden garantizar la convergencia al óptimo global[1].
Sin embargo, las características de los problemas del mundo real suelen ser tan particulares que resulta difícil y en ocasiones hasta imposible aplicar métodos matemáticos tradicionales en su resolución. Cuando la aplicación del método tradicional es compleja, o cuando el costo computacional es alto y aunque los resultados sean buenos no sean los esperados, entonces puede considerarse el uso de un método no tradicional, una heurística.
Dentro de los métodos heurísticos se tiene una categoría que basa su funcionamiento en fenómenos inteligentes encontrados en la naturaleza, a los cuales se les conoce como algoritmos bio-inspirados o inspirados en la naturaleza[15].

2.2. Algoritmos de optimización inspirados en la naturaleza

Los algoritmos de optimización inspirados en la naturaleza basan su funcionamiento en fenómenos inteligentes encontrados en la naturaleza. De manera general, este tipo de algoritmos generan un conjunto de soluciones al problema (población inicial), evalúan cada solución en la función objetivo a optimizar, seleccionan las mejores soluciones de la población con base en su valor en la función objetivo, generan nuevas soluciones a partir de las mejores soluciones utilizando operadores de variación y evalúan las nuevas soluciones para escoger las soluciones que serán parte de la siguiente iteración o generación [38].
Este tipo de algoritmos se dividen a su vez en dos subgrupos de acuerdo con el tipo de fenómeno

[1]Para una amplia revisión de los métodos matemáticos para resolver problemas de optimización se puede consultar el libro: *Practical Mathematical Optimization* de Jan A. Snyman [54].

natural en el que se basan: Algoritmos de inteligencia colectiva y Algoritmos Evolutivos [38].
Los algoritmos de inteligencia colectiva se inspiran en el comportamiento de seres vivos que interactúan de manera local con su ambiente, de esa interacción surgen comportamientos sociales que les permiten resolver problemáticas complejas de manera conjunta [15], por otra parte, los algoritmos evolutivos basan su funcionamiento en la teoría de la evolución de las especies, la supervivencia del más apto y la transmisión de características de padres a hijos [12].
En esta sección se detalla el funcionamiento de dos de los principales algoritmos de optimización inspirados en la naturaleza: el algoritmo de Optimización por Cúmulos de Partículas, uno de los principales algoritmos de inteligencia colectiva, y Evolución Diferencial, uno de los principales algoritmos evolutivos.

2.2.1. Optimización por cúmulos de partículas

La Optimización por Cúmulos de Partículas, PSO (*Particle Swarm Optimization*) por sus siglas en inglés, es un algoritmo de búsqueda estocástico para resolver problemas de optimización sobre espacios continuos, propuesto por Kennedy y Eberhart [11]. PSO simula el comportamiento de interacción social de un enjambre de individuos (llamadas partículas) que se mueven en el espacio de búsqueda, donde cada partícula representa una solución candidata.
El algoritmo PSO se inicializa con una población de NP partículas colocadas aleatoriamente dentro del espacio de búsqueda. La posición de cada partícula está representada por un vector: $\vec{X}_i = (X_{i,1}, X_{i,2}, ..., X_{i,D}), i = 1, ..., NP$, donde $X_{i,j}$ es la $j-$esima componente de la partícula i, mientras que D es la dimensionalidad del espacio de búsqueda.
Cada una de las partículas representa una solución candidata a una función objetivo (Ecuación 2.1) en un problema de optimización numérica. Esta función objetivo, en el contexto de los algoritmos de optimización inspirados en la naturaleza, recibe el nombre de *función de aptitud*. Cada una de las soluciones candidatas son evaluadas en la función de aptitud y como resultado de ésta evaluación se obtiene un *valor de aptitud* que indica que tan apta o que tan buena es la solución candidata; ante un problema de minimización, entre menor sea el valor de aptitud, mejor es la solución.
Las partículas se mueven en el espacio de búsqueda a fin de encontrar la solución óptima, por lo tanto, cada partícula tiene una velocidad, que se representa como: $\vec{V}_i = (V_{i,1}, V_{i,2}, ..., V_{i,D})$. La mejor posición previa de la partícula[2] es registrada por un vector: $\vec{pbest}_i = (pbest_{i,1}, pbest_{i,2}, ..., pbest_{i,D})$, y la mejor posición obtenida por la población se denomina: $\vec{gbest} = (gbest_1, gbest_2, ..., gbest_D)$. En el momento de la inicialización, $\vec{pbest}_i$ se establece a $\vec{X}_i$.
En cada generación, las partículas actualizan su velocidad tomando la combinación lineal de la velocidad anterior con la experiencia personal y global del enjambre mediante la Ecuación 2.5 de

[2] Aquella posición previa que obtuvo el mejor valor de aptitud.

actualización de velocidad:

$$\vec{V}_i^g = w\vec{V}_i^{g-1} + c_1 r_1(\vec{pbest}_i^{g-1} - \vec{X}_i^{g-1}) + c_2 r_2(\vec{gbest}^{g-1} - \vec{X}_i^{g-1}) \tag{2.5}$$

En donde $\vec{V}_i^g$ es la velocidad de la partícula i en la generación g; w es el factor de inercia[3], que determina cuánto se conserva la velocidad anterior [51]; c_1 y c_2 son los coeficientes de aceleración[4], que controlan la proporción relativa de la cognición y la interacción social en el enjambre; r_1 y r_2 son números aleatorios[5] generados de manera uniforme dentro de $[0,1]$; $\vec{pbest}_i^{g-1}$ es la mejor posición previamente ocupada por la partícula i mientras que $\vec{gbest}^{g-1}$ es la mejor posición previamente ocupada por cualquier partícula del enjambre.

La posición de las partículas se actualiza cada generación usando la ecuación 2.6:

$$\vec{X}_i^g = \vec{X}_i^{g-1} + \vec{V}_i^g \tag{2.6}$$

La Ecuación 2.6 puede generar posiciones fuera de los límites de cada variable. Cuando esto sucede, se hace necesaria la aplicación de un método para el manejo de restricciones de límites, de modo que permita mantener la búsqueda dentro del espacio definido por los límites inferior y superior de cada variable.

Después de actualizar la velocidad y las posiciones, el mejor vector personal $\vec{pbest}_i^g$ de las partículas también se actualiza como:

$$\vec{pbest}_i^g = \begin{cases} \vec{X}_i^g & \text{si } f(\vec{X}_i^g) <= f(\vec{pbest}_i^{g-1}) \\ \vec{pbest}_i^{g-1} & \text{en cualquier otro caso} \end{cases} \tag{2.7}$$

La mejor posición personal para la partícula i en la generación (g) se actualiza de acuerdo con el valor de la función de aptitud de su anterior $\vec{pbest}_i^{g-1}$ y la posición $\vec{X}_i^g$ actual.

En el Algoritmo 1 se muestra el pseudocódigo del algoritmo clásico de optimización por cúmulos

[3]Cuando se asigna un valor pequeño a $w(< 0.8)$, PSO se comporta como un algoritmo de búsqueda local, mientras que cuando se asigna a w un valor grande (> 1.2) se comporta como un método de búsqueda global, por tales razones, lo recomendable es que se asigne a w un valor entre este rango $(0.8 < w < 1.2)$ [51]. Varios autores [14, 27, 51] han coincidido en recomendar el uso de $w = 0.95$.

[4]Los coeficientes c_1 y c_2, de aceleración cognitiva y social, determinan en qué medida influyen sobre el movimiento de la partícula su propia memoria y la cooperación entre individuos respectivamente. Si el valor de c_1 es mayor, las partículas tienden a ser más autónomas, mientras que si el valor c_1 es mayor, las partículas tienden más a seguir el comportamiento social. Este valor es una constante que se establece antes de ejecutar el algoritmo; algunos autores [13, 14, 50], con base en estudios experimentales, han sugerido emplear los siguientes valores: $c_1 = c_1 = 1.49445$, con la finalidad de evitar la convergencia prematura del algoritmo; atendiendo a estas recomendaciones, en el presente trabajo se emplearán estos mismos valores.

[5]El objetivo de los valores aleatorios r_1 y r_2 es emular el comportamiento estocástico y un tanto impredecible que exhibe la población del enjambre [27].

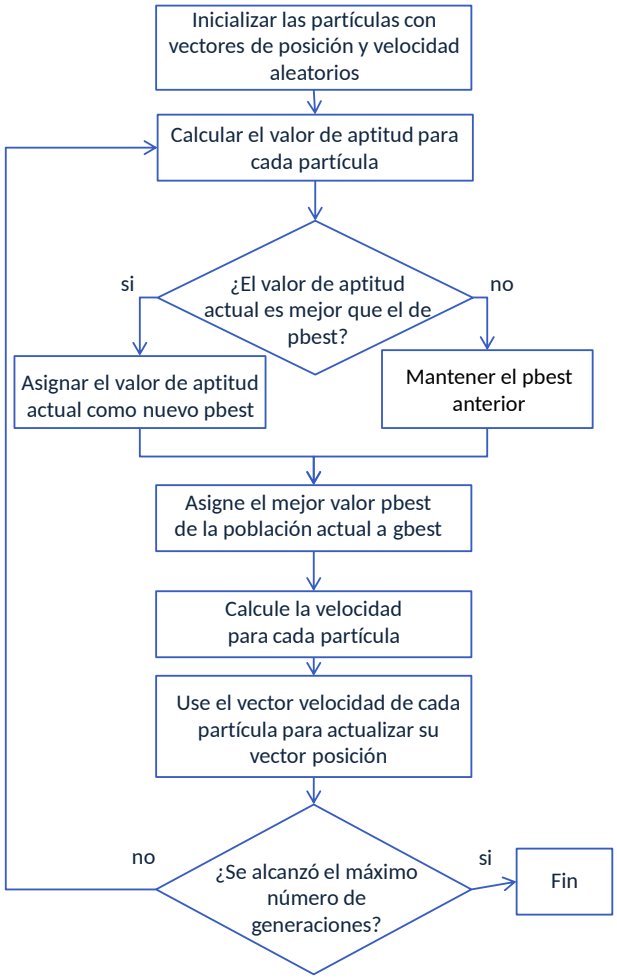


Figura 2.1: Diagrama de flujo del algoritmo de optimización por cúmulos de partículas.

de partículas. Como se aprecia en la línea 17, después de actualizar las posiciones de las partículas se debe aplicar algún método para el manejo de restricciones de límite, con la finalidad de reposicionar las partículas que pudieran haber salido del espacio de búsqueda.

Las reglas de actualización de PSO (Ecuaciones 2.5, 2.6 y 2.7) se aplican en cada generación hasta que se cumple un criterio de terminación predefinido, el cual puede ser un número máximo de evaluaciones, como muestra el diagrama de flujo de la Figura 2.1.

1: $g = 0$
2: **para** i = 1 hasta NP **hacer**
3: Inicializar la posición $\vec{X}_i^g \in \mathbb{R}^D$ aleatoriamente
4: Inicializar la velocidad $\vec{V}_i^g \in \mathbb{R}^D$ aleatoriamente
5: $\vec{pbest}_i^g = \vec{X}_i^g$
6: **fin para**
7: $gbest^g = argmin(f(\vec{pbest}_i^g))\forall_i, i = 1,...,NP$
8: **repetir**
9: $g = g + 1$
10: **para** i = 1 hasta NP **hacer**
11: **para** j = 1 hasta D **hacer**
12: $r_1 = rand[0,1]$
13: $r_2 = rand[0,1]$
14: $V_{i,j}^g = wV_{i,j}^{g-1} + c_1 r_1 (pbest_{i,j}^{g-1} - X_{i,j}^{g-1}) + c_2 r_2 (gbest_j^{g-1} - X_{i,j}^{g-1})$
15: $X_{i,j}^g = X_{i,j}^{g-1} + V_{i,j}^g$
16: **fin para**
17: **Aplicar un método para el manejo de restricciones de límite.**
18: **si** $f(\vec{X}_i^g) \leq f(\vec{pbest}_i^{g-1})$ **entonces**
19: $\vec{pbest}_i^g = \vec{X}_i^g$
20: **fin si**
21: **si** $f(\vec{X}_i^g) \leq f(\vec{gbest}^{g-1})$ **entonces**
22: $\vec{gbest}^g = \vec{X}_i^g$
23: **fin si**
24: **fin para**
25: **hasta que** *se cumpla el criterio de terminación.*

Algoritmo 1: Pseudocódigo del algoritmo PSO clásico.

Topologías

La principal característica de los algoritmos de interacción social es la comunicación que existe entre los miembros del enjambre, con el objetivo de resolver un problema de optimización en común. La interacción se logra indicando con quien tendrá comunicación directa cada partícula durante la evolución del algoritmo, a esto se le conoce como la topología, y es la que define el tipo de interacción en el enjambre.

El algoritmo de PSO, en su versión canónica, presentado en esta sección permite que todas las partículas se comuniquen entre sí, por lo que todas siguen a la que ha encontrado la mejor posición (todas siguen a *gbest*) en algún momento de la búsqueda, a esta *topología* se le conoce como *todos conectados*, ver Figura 2.2 (A).

En la topología de anillo (Figura 2.2 (B)) cada partícula con índice k se conecta con las partículas con índice *k-1* y *k+1* (salvo las partículas 1 y NP, que se conectan entre sí, donde NP es el número de partículas), en esta topología, también conocida como *lbest* cada partícula solo es afectada por sus vecinos inmediatos [35].

En la topología de estrella (Figura 2.2 (C)) la información pasa a través de una sola partícula, ya que un nodo central se conecta con todas las demás partículas. Las partículas entonces siguen al nodo central, si la partícula central es la mejor, entonces el comportamiento de esta topología es

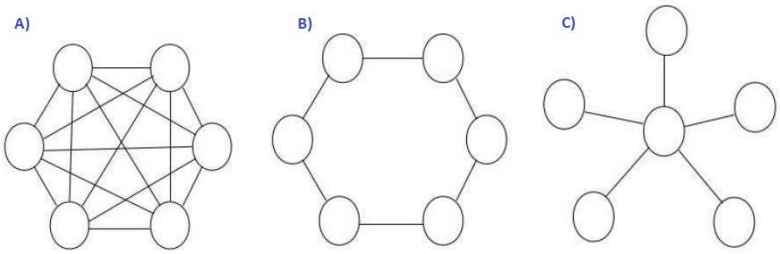


Figura 2.2: Tres distintas topologías empleadas en PSO: A) Todos conectados, B) Anillo y C) Estrella.

similar a la topología todos conectados [35].

En la topología de Malla (Figura 2.3 (D)) o Von Neumann, los nodos son organizados en un arreglo

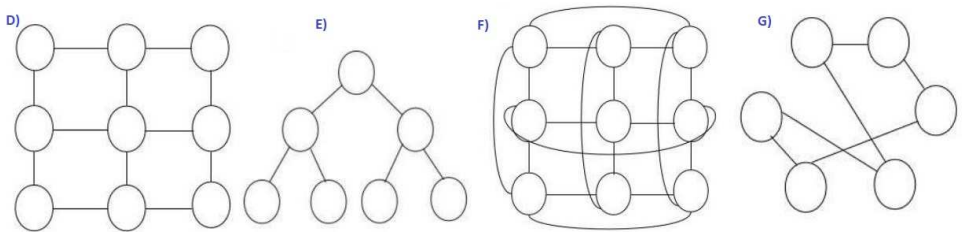


Figura 2.3: Cuatro distintas topologías empleadas en PSO: D) Malla, E) Árbol, F) Toroidal y G) Aleatoria.

rectangular y se conectan con las partículas que se encuentran al norte, sur, este y oeste de ellas (excepto cuando no existan partículas en esas posiciones, como en las esquinas y en los extremos), cada partícula tendrá un máximo de cuatro vecinos y un mínimo de dos, dependiendo de la posición en la que se encuentren en la malla [35].

En la topología de Árbol o jerárquica (Figura 2.3 (E)) se tiene un nodo raíz que es conectado a uno o más individuos. El nodo raíz es el primer nivel de la jerarquía, mientras que los nodos que fueron conectados a este forman parte del segundo nivel. Posteriormente los nodos del segundo nivel se conectan con otros nodos, los cuales ahora son parte del tercer nivel. De esta manera se siguen conectando nodos, hasta completar todas las conexiones entre partículas [35].

La topología Toroidal (Figura 2.3 (F)) es similar a la topología de malla, salvo que en esta todos los nodos están conectados con cuatro vecinos, esto significa que incluso los nodos de las esquinas y de los extremos también tienen conexión con cuatro nodos, considerando la malla como un espacio toroidal[35].

En la topología Aleatoria (Figura 2.3 (G)), al inicio del algoritmo se determina cuantos vecinos

tendrá cada partícula y posteriormente se seleccionan aleatoriamente cada uno de los vecinos para cada partícula [35].

Ejemplo práctico de PSO

Para ejemplificar el uso del algoritmo PSO descrito en esta sección, a continuación se presenta el proceso de optimización de una función de aptitud en dos dimensiones $D = 2$, que consiste en minimizar:

$$f(x_1, x_2) = \frac{{x_1}^3 + {x_2}^3}{100} \tag{2.8}$$

Sujeta a:

$$-20 \leq x_1 \leq 20 \tag{2.9}$$

$$-20 \leq x_2 \leq 20 \tag{2.10}$$

En la Figura 2.4 se presenta una gráfica en tres dimensiones, considerando las dimensiones x_1 y

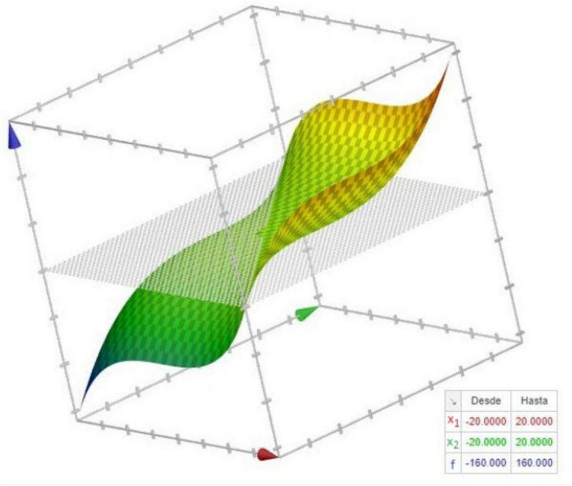


Figura 2.4: Gráfica de la función $f(x_1, x_2) = ({x_1}^3 + {x_2}^3)/100$.

x_2, y el valor que toma la función $f(x1, x2)$, de la Ecuación 2.8. Esta gráfica esta limitada en x_1 y x_2 con las restricciones de límite planteadas en las Ecuaciones 2.9 y 2.10. Debido a que se trata de un problema de minimización, el valor mínimo u óptimo para $f(x_1, x_2)$ se encuentra en $x_1 = -20$ y $x_2 = -20$; $f(-20, -20) = -160$.

Antes de aplicar el algoritmo PSO se debe determinar que parámetros se usarán para w, c_1, c_2 y NP. Para este ejemplo, con fines didácticos, se empleará una población de seis partículas, $NP = 6$; el resto de los parámetros tomarán los siguientes valores: $w = 0.95, c_1 = c_2 = 1.4944$; atendiendo

Tabla 2.1: Valores de las velocidades, posiciones, valor de aptitud y *pbest* para la generación inicial (*generación 0*) del problema de ejemplo. El valor de aptitud de la mejor partícula, la partícula *p6* y por tanto *gbest* se encuentra resaltado en negritas.

	velocidad		***posición***			***pbest***		
i	$V^0_{i,1}$	$V^0_{i,2}$	$X^0_{i,1}$	$X^0_{i,2}$	$f(\vec{X}^0_i)$	$pbest^0_{i,1}$	$pbest^0_{i,2}$	$f(\vec{pbest}^{\,0}_i)$
1	-5.0756	-0.5853	9.8997	-1.8142	9.6424	9.8997	-1.8142	9.6424
2	10.0234	19.4522	15.2641	15.4015	72.0976	15.2641	15.4015	72.0976
3	3.9291	0.9015	13.2968	2.5088	23.6673	13.2968	2.5088	23.6673
4	-4.7117	2.6163	1.7505	4.0216	0.7041	1.7505	4.0216	0.7041
5	-0.3772	0.7803	1.7725	8.7352	6.7210	1.7725	8.7352	6.7210
6	6.8429	-3.5085	-1.0062	-0.3894	**-0.0108**	-1.0062	-0.3894	**-0.0108**

a las recomendaciones de trabajos previos [13, 14, 50]. También deberá determinarse cuando finalizará el algoritmo, en este caso el algoritmo terminará cuando haya realizado 24 evaluaciones de la función de aptitud (Ecuación 2.8).

El primer paso consiste en inicializar las posiciones, para ello, se asignan valores aleatorios a las componentes $X^0_{i,1}$ y $X^0_{i,2}$ de las seis partículas, dentro de sus límites permitidos. En la Tabla 2.1 se pueden observar las posiciones iniciales de las seis partículas, los valores fueron asignados de manera aleatoria entre *-20* y *20*, atendiendo a las restricciones de límite de las Ecuaciones 2.9 y 2.10.

Una vez que se tienen las posiciones iniciales se puede calcular su valor de aptitud $f(\vec{X}^0_i)$ aplicando la Ecuación 2.8 para cada una de las partículas, por ejemplo el valor de aptitud para la partícula *5* (ver Tabla 2.1) se calcula de la siguiente forma:

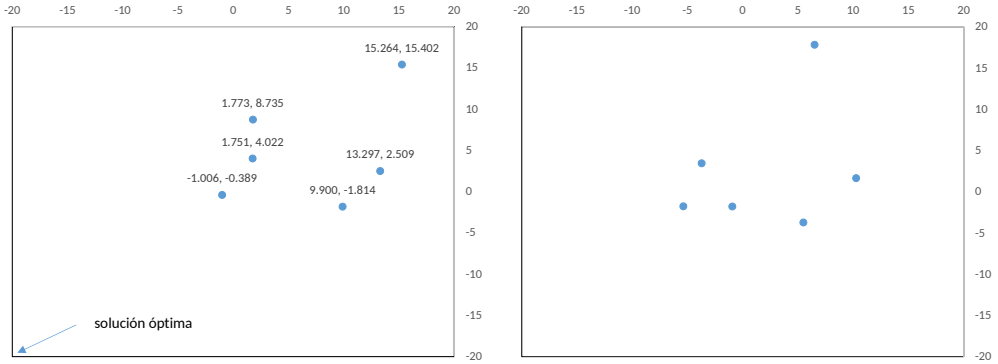
$f(\vec{X}^0_5) = f(1.7725, 8.7352) = (1.7725^3 + 8.7352^3)/100 = 6.7210$

Posteriormente se asignan las velocidades iniciales. Siguiendo el mismo proceso, se asignan valores aleatorios a las componentes $V^0_{i,1}$ y $V^0_{i,2}$ de la velocidad de cada partícula. La primer componente recibe valores aleatorios atendiendo las restricciones de límite de la Ecuación 2.9 y la segunda componente atiende a las restricciones de límite de la Ecuación 2.10. En la Tabla 2.1 se pueden apreciar las velocidades iniciales.

Por tratarse de la población inicial, los valores de la mejor posición previa de cada partícula *pbest*, son los mismos valores de las posiciones iniciales, por tanto, los valores de aptitud son también los mismos, como se aprecia en la Tabla 2.1.

Para el caso de la mejor posición de la población, es decir *gbest*, se elige la partícula con el mejor valor de aptitud, en este caso el valor mínimo por tratarse de un problema de minimización. Como se aprecia en la Tabla 2.1, la partícula *6* cuenta con el mejor valor de aptitud, por tanto: $\vec{gbest}^{\,0} = (-1.0062, -0.3894)$.

Hasta este momento se ha inicializado el algoritmo. En la Figura 2.5(a) se pueden apreciar las



(a) Generación 0. (b) Generación 1.

Figura 2.5: Posiciones de las partículas en las generaciones 0 y 1.

posiciones iniciales de las partículas[6] dentro del espacio de búsqueda, estas partículas conforman la generación 0.

A partir de este momento, generación tras generación se repite un proceso que consiste en actualizar las velocidades, las posiciones, los valores de aptitud, los valores para *pbest* y *gbest*, de forma repetida hasta que se cumple con un criterio de terminación.

Los valores de la siguiente generación, la generación 1, se presentan en la Tabla 2.2. El primer valor en actualizarse son las velocidades, para esto se aplica la Ecuación 2.5, por ejemplo, la componente $V^1_{1,1}$ de la velocidad para la partícula *1* se calcula de la siguiente manera:

$V^1_{1,1} = wV^0_{1,1} + c_1r_1(pbest^0_{1,1} - X^0_{1,1}) + c_2r_2(gbest^0_1 - X^0_{1,1})$

$V^1_{1,1} == w(-5.0756) + c_1r_1(9.8997 - 9.8997) + c_2r_2(-1.0062 - 9.8997)$

donde -5.0756 es la componente $V^0_{1,1}$ de la velocidad para la partícula *1* en la generación anterior (generación 0), 9.8997 es el valor de la componente $X^0_{1,1}$ de la posición anterior, y en este caso coincide el valor con su anterior $pbest^0_{1,1}$, y -1.0062 es la componente $gbest^0_1$ del anterior *gbest*, todos estos valores se pueden consultar en la Tabla 2.1.

Considerando los valores: w, c_1 y c_2 mencionados anteriormente, y los siguientes valores aleatorios: $r_1 = 0.7722$ y $r_2 = 0.6415$, obtendríamos lo siguiente[7]:

$V^1_{1,1} = -5.0756 \times 0.95 + 1.4944 \times 0.7722 \times (0) + 1.4944 \times 0.6415 \times (-10.959)$

$V^1_{1,1} = -4.606 + 0 - 10.6701 = -15.2761$

siguiendo este mismo procedimiento se ha actualizado la segunda componente $V^1_{1,2}$ de la velocidad para la partícula *1*, empleando los siguientes valores aleatorios[8] $r_1 = 0.0010$ y $r_2 = 0.2868$.

[6] Éstas corresponden a los valores $X^0_{i,1}$ y $X^0_{i,2}$ de las posiciones de la Tabla 2.1.

[7] El lector podría encontrar pequeñas diferencias en los decimales, esto es debido a cuestiones de redondeo, los números presentados aquí se han truncado a cuatro decimales.

[8] Se comparten los valores r_1 y r_2 para que el lector pueda comprobar los cálculos.

Tabla 2.2: Valores de las velocidades, posiciones, valor de aptitud y *pbest* para la *generación 1*. El valor de aptitud de *gbest* se encuentra resaltado en negritas.

	velocidad		***posición***			***pbest***		
i	$V_{i,1}^1$	$V_{i,2}^1$	$X_{i,1}^1$	$X_{i,2}^1$	$f(\vec{X}_i^1)$	$pbest_{i,1}^1$	$pbest_{i,2}^1$	$f(\vec{pbest}_i^1)$
1	-15.2761	0.0548	-5.3764	-1.7594	**-1.6085**	-5.3764	-1.7594	**-1.6085**
2	-8.7552	6.7725	6.5089	17.8260	59.4025	6.5089	17.8260	59.4025
3	-3.0371	-0.8639	10.2597	1.6449	10.8439	10.2597	1.6449	10.8439
4	-5.4747	-0.5621	-3.7242	3.4595	-0.1025	-3.7242	3.4595	-0.1025
5	-2.7176	-10.5112	-0.9451	-1.7760	-0.0645	-0.9451	-1.7760	-0.0645
6	6.5008	-3.3330	5.4946	-3.7224	1.1430	-1.0062	-0.3894	-0.0108

Una vez actualizadas las velocidades de las seis partículas, se actualizan sus nuevas posiciones, aplicando la Ecuación 2.6, para lo cual se consideran las posiciones anteriores y las nuevas velocidades. Por ejemplo, las componentes $X_{2,1}^1$ y $X_{2,2}^1$ de la posición para la partícula *2* se calculan de la siguiente manera:

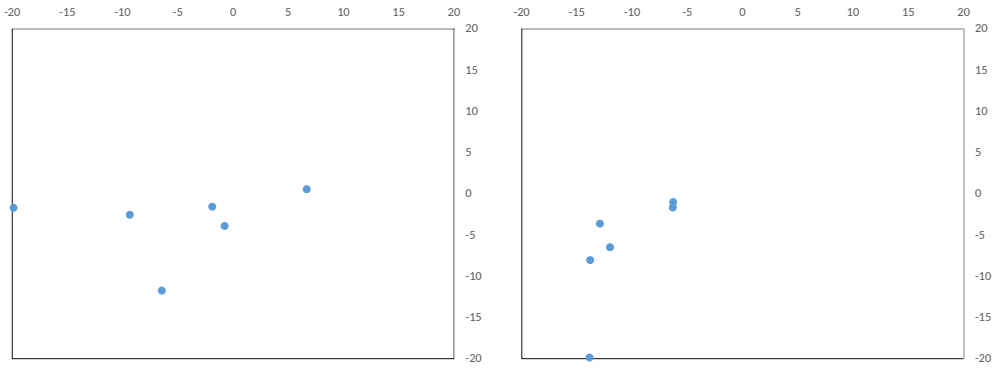
$X_{2,1}^1 = X_{2,1}^0 - V_{2,1}^1 = 15.2641 + (-8.7552) = 6.5089$

$X_{2,2}^1 = X_{2,2}^0 - V_{2,2}^1 = 15.4015 + 6.7725 = 22.1740$

Como puede apreciarse, al momento de actualizar la nueva posición de la partícula *2*, su componente $X_{2,2}^1$ sale del espacio de búsqueda pues toma un valor mayor a su límite superior que es 20. Cuando esto sucede se debe aplicar algún método para el manejo de restricciones de límite. En este caso, aplicando el método *Reflection*, que se explicará más adelante, en la Sección 3.2.2, la componente $X_{2,2}^1$ se regresaría la misma magnitud que salió (2.1740), quedando finalmente su valor $x2 = 20 - 2.1740 = 17.826$. En la Tabla 2.2 se presenta (resaltada en amarillo) la posición ya corregida y en la Figura2.5(b) se pueden apreciar las posiciones de las partículas en la generación 1, en la parte superior se encuentra la partícula que había salido del espacio de búsqueda y volvió a introducirse mediante la aplicación del método *Reflection*.

Después de actualizar todas las nuevas posiciones y actualizar también su valor de aptitud aplicando la Ecuación 2.8, ya explicada anteriormente, se eligen los nuevos valores pbest aplicando la Ecuación 2.7. Para cualquier partícula, si el valor de aptitud de la nueva posición es mejor que el valor de aptitud de su anterior *pbest* el nuevo *pbest* toma los valores de la nueva posición, en caso contrario toma los valores de su anterior *pbest*. En los valores *pbest* de la generación 1 (Tabla 2.2) solo la partícula *6* (resaltado en amarillo) conservó su anterior *pbest*, el resto fueron actualizados con las nuevas posiciones.

Este proceso de calcular las nuevas velocidades, posiciones, valor de aptitud, *pbest* y *gbest* se repite hasta que se cumpla una condición de paro, para el caso del presente ejemplo, el algoritmo se repite hasta completar 24 evaluaciones en la función de aptitud; como se tiene una población de 6 partículas, considerando las evaluaciones de la generación 0, la condición de paro se cumple en la



(a) Generación 2. (b) Generación 3.

Figura 2.6: Posiciones de las partículas en las generaciones 2 y 3.

generación 3. En la Figura 2.6 se pueden observar las posiciones de las partículas en las generaciones 2 y 3, como puede apreciarse, varias de ellas ya se encuentran cerca de la posición óptima.

2.2.2. Evolución Diferencial

La Evolución Diferencial (DE) es un algoritmo de búsqueda estocástico aplicado en la resolución de problemas de optimización sobre espacios continuos [55]. El algoritmo DE tiene como objetivo evolucionar una población de NP vectores en D dimensiones que representan soluciones candidatas en la generación g (Ecuación 2.11).

$$\vec{X}_i^g = \{X_{i,1}^g, X_{i,2}^g, ..., X_{i,D}^g\}, i = 1, ..., NP \tag{2.11}$$

Donde $X_{i,j}$ es la j-ésima componente de la solución i. La población inicial se genera de manera uniforme al azar dentro de los límites predefinidos inferior l_j y superior u_j para cada variable $X_{i,j}$. DE consiste principalmente en tres operaciones que se repiten generación tras generación hasta que se cumple un criterio de terminación. Estas operaciones se describen a continuación.

- *Operador de mutación.* Por cada vector objetivo (padre) $\vec{X}_i^{g-1}$ en la generación $g-1$, se genera un vector mutante $\vec{V}_i^g$ en la generación g, empleando alguna estrategia como la *DE/rand/1/bin*, descrita en la Ecuación 2.12.

$$\vec{V}_i^g = \vec{X}_{r_1}^{g-1} + F \times (\vec{X}_{r_2}^{g-1} - \vec{X}_{r_3}^{g-1}) \tag{2.12}$$

  En donde $r_1 \neq r_2 \neq r_3 \neq i$ son índices generados en el rango $[1, NP]$ y $F > 0$ es un valor real

que representa el factor de mutación[9].

El operador de mutación puede generar vectores mutantes fuera de los límites de las variables, haciendo necesaria la aplicación de algún método para el manejo de restricciones de límites, el cual reposicionará los vectores solución dentro del espacio de búsqueda.

Existen otras estrategias de mutación, como DE/best/1/bin y DE/rand/1/exp [39], estas estrategias son descritas en la Tabla 2.3.

Tabla 2.3: Estrategias de mutación para Evolución Diferencial. $rand_j \in [0,1]$ representa un número real seleccionado de forma uniformemente aleatoria, $j_{rand} \in [1,D]$ es un número entero seleccionado de forma aleatoria. $\vec{x}_{best,g}$ es el mejor vector solución en la generación g. CR es el factor de cruza.

Nomenclatura	Variante (Estrategia)
DE/rand/1/bin	$U^g_{i,j} = \begin{cases} X^{g-1}_{r_1,j} + F \cdot (X^{g-1}_{r_2,j} - X^{g-1}_{r_3,j}) & \text{si } rand_j[0,1] < CR \ \vee \ j = j_{rand} \\ X^{g-1}_{i,j} & \text{en cualquier otro caso} \end{cases}$
DE/best/1/bin	$U^g_{i,j} = \begin{cases} X^{g-1}_{best,j} + F \cdot (X^{g-1}_{r_1,j} - X^{g-1}_{r_2,j}) & \text{si } rand_j[0,1] < CR \ \vee \ j = j_{rand} \\ x^j_{i,g} & \text{en cualquier otro caso} \end{cases}$
DE/target-to-best/1	$U^g_{i,j} = X^{g-1}_{i,j} + F \cdot (X^{g-1}_{best,j} - X^{g-1}_{i,j}) + F \cdot (X^{g-1}_{r_1,j} - X^{g-1}_{r_2,j})$

- *Operador de cruza.* El operador de cruza genera un vector de prueba (hijo) mediante la combinación de un vector objetivo con su correspondiente vector mutante, como se indica en la Ecuación 2.13.

$$U^g_{i,j} = \begin{cases} V^g_{i,j} & \text{si } (\text{rand}_j[0,1] \leq CR) \ \vee \ j = j_{rand}, \\ X^{g-1}_{i,j} & \text{en cualquier otro caso} \end{cases} \tag{2.13}$$

en donde $j = 1,...,D$, $CR \in [0,1]$ es el factor de cruza[10], un valor definido por el usuario, el cual indica qué tan similar es el vector de prueba con respecto del vector mutante, rand_j es un valor real generado de forma aleatoria con distribución uniforme entre 0 y 1, y j_{rand} es un número entero generado de forma aleatoria dentro del rango $[1,D]$, lo cual evita tener valores duplicados entre el vector de prueba y el vector objetivo.

[9] El factor de mutación F está relacionado con la velocidad de convergencia. Si F toma valores pequeños el algoritmo permite realizar una mejor búsqueda local (explotación), por el contrario, al asignar un valor mayor a F el algoritmo permite una mejor búsqueda global (Exploración) [63]. Es recomendable emplear valores entre: $0.4 < F < 0.95$, algunos autores recomiendan emplear $F = 0.9$, ya que mantiene un compromiso entre velocidad y probabilidad de convergencia [47, 63].

[10] El vector objetivo se mezcla con el vector mutante para generar el vector de prueba, en esta operación el factor de cruza $CR \in [0,1]$ determina que tanto se parecerá el vector de prueba al vector mutante. Algunos autores [26, 39] recomiendan emplear un $CR = 0.9$, con esto, el vector de prueba será similar, en un 90 %, al vector mutante.

- *Operador de selección.* Finalmente, el operador de selección se resume en la Ecuación 2.14, en donde se selecciona el mejor vector de acuerdo a su valor de aptitud entre el vector objetivo y su correspondiente vector de prueba. El vector con el mejor valor de aptitud permanecerá para la siguiente generación.

$$\vec{X}_i^g = \begin{cases} \vec{U}_i^g & \text{si } f(\vec{U}_i^g) < f(\vec{X}_i^{g-1}), \\ \vec{X}_i^{g-1} & \text{en cualquier otro caso} \end{cases} \tag{2.14}$$

Las operaciones de *mutación*, *cruza* y *selección* se repiten generación tras generación hasta que se cumpla algún criterio de terminación predefinido, que podría ser un número máximo de generaciones, como se muestra en el diagrama de flujo de la Figura 2.7.

El Algoritmo 2 presenta el seudocódigo del algoritmo DE/rand/1/bin.

```
g = 0
Generar una población inicial de forma aleatoria X_i^g ∀i, i = 1,...,NP
Evaluar f(X_i^g) ∀i, i,...,NP
repetir
    g = g + 1
    para i=1 hasta NP hacer
        Seleccionar aleatoriamente r1 ≠ r2 ≠ r3 ≠ i
        j_rand = randint[1,n]
        para j = 1 hasta D hacer
            si (rand_j[0,1]) < CR ∨ j = j_rand entonces
                U_{i,j}^g = X_{r1,j}^{g-1} + F(X_{r2,j}^{g-1} − X_{r3,j}^{g-1})
            si no
                U_{i,j}^g = X_{i,j}^{g-1}
            fin si
        fin para
        Aplicar un método para el manejo de restricciones de límite.
        si f(U_i^g) ≤ f(X_i^{g-1}) entonces
            X_i^g = U_i^g
        si no
            X_i^g = X_i^{g-1}
        fin si
    fin para
hasta que Se cumpla el criterio de terminación.
```

Algoritmo 2: DE/rand/1/bin.

Ejemplo práctico de DE

Para ejemplificar el uso del algoritmo DE descrito en esta sección, a continuación se presenta el proceso de optimización de la función descrita en la Ecuación 2.8, sujeta a las restricciones de límite presentadas en las Ecuaciones 2.9 y 2.10. Esta ecuación de dos dimensiones, que se encuentra representada en la Figura 2.4, implica un problema de minimización cuyo valor óptimo se

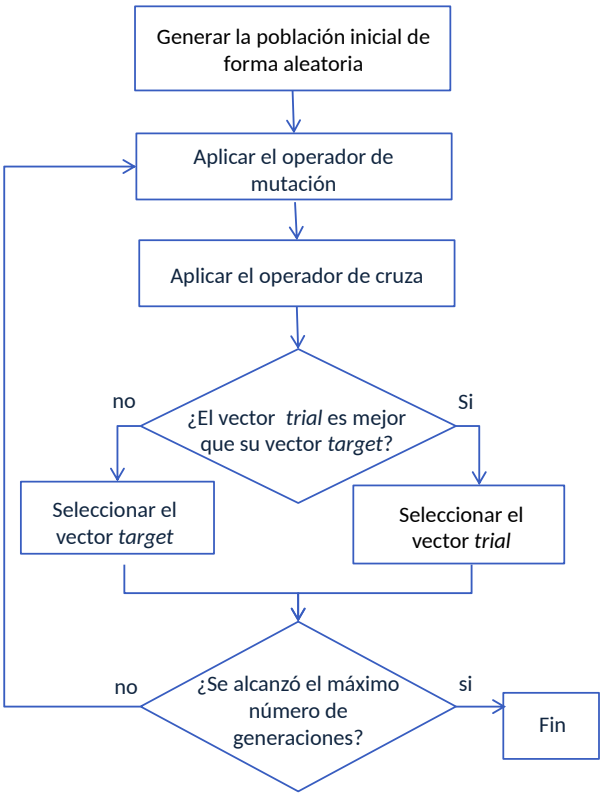


Figura 2.7: Diagrama de flujo del algoritmo de Evolución Diferencial.

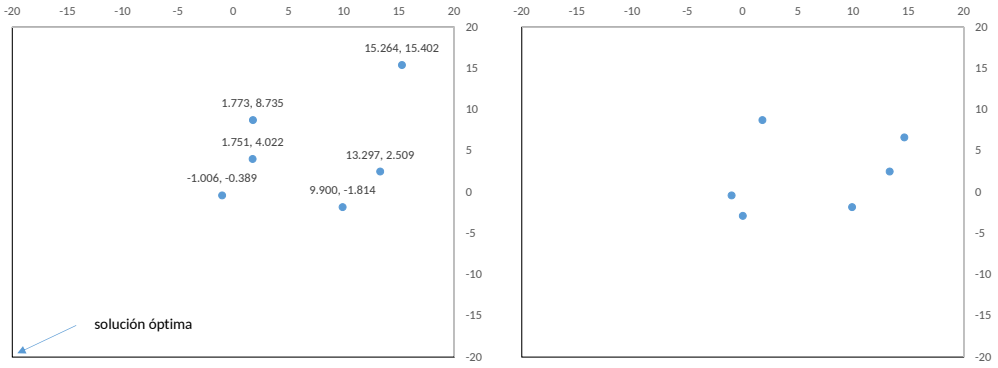
encuentra en $f(-20,-20) = -160$.

Antes de aplicar el algoritmo DE se debe determinar que parámetros se usarán para F, CR y NP. Para este ejemplo, con fines didácticos, se empleará una población de seis vectores solución, $NP = 6$; el resto de los parámetros tomarán los siguientes valores: $F = 0.9$ y $CR = 0.9$; atendiendo a las recomendaciones de trabajos previos [39, 47, 63]. También deberá determinarse cuando finalizará el algoritmo, en este caso el algoritmo terminará cuando haya realizado 24 evaluaciones de la función de aptitud (Ecuación 2.8).

El primer paso consiste en generar la población inicial (generación 0) de forma aleatoria. Esto se realiza asignando valores aleatorios a cada una de las variables x_1 y x_2 de los seis vectores solución, dentro de sus límites permitidos, -20 y 20, atendiendo a las restricciones de límite de las Ecuaciones 2.9 y 2.9.

Tabla 2.4: Vectores solución y valor de aptitud para la población inicial (*generación 0*). El mejor valor de aptitud se encuentra resaltado en negritas.

	Vec. Solución		
i	$X^0_{i,1}$	$X^0_{i,2}$	$f(\vec{X}^0_i)$
1	9.8997	-1.8142	9.6424
2	15.2641	15.4015	72.0976
3	13.2968	2.5088	23.6673
4	1.7505	4.0216	0.7041
5	1.7725	8.7352	6.7210
6	-1.0062	-0.3894	**-0.0108**



(a) Generación 0. (b) Generación 1.

Figura 2.8: Posiciones de los vectores solución en el ejemplo con DE para las generaciones 0 y 1.

Una vez que se tienen las posiciones iniciales se puede calcular el valor de aptitud $f(x_1,x_2)$ aplicando la Ecuación 2.8 a cada uno de los vectores solución, por ejemplo, para el vector $\vec{X}^0_5$, su valor de aptitud sería:

$f(\vec{X}^0_5) = f(1.7725, 8.7352) = (1.7725^3 + 8.7352^3)/100 = 6.7210$

En la Tabla 2.4 se pueden observar los vectores solución y su correspondiente valor de aptitud en la población inicial (generación 0) y en la Figura 2.8(a) se pueden ver las posiciones de ésta población inicial dentro del espacio de búsqueda[11].

Hasta este momento se ha inicializado el algoritmo. A partir de aquí, generación tras generación se repite un proceso que consiste en generar los vectores mutantes, los vectores de prueba y seleccionar los descendientes hasta que se cumpla un criterio de terminación.

En la Tabla 2.5 se presentan los valores de los vectores mutantes, los vectores de prueba y los vectores solución (sobrevivientes) de la generación 1.

Para generar los vectores mutantes se aplica el operador de mutación descrito en la Ecuación 2.12.

[11] De manera intencionada se emplearon las mismas posiciones iniciales de la Tabla 2.1 en el ejemplo del algoritmo PSO, por tanto, su ubicación dentro del espacio de búsqueda es la misma, considerando que se trata del mismo problema.

Tabla 2.5: Vectores mutantes, de prueba y vectores solución en la (*generación 1*). El mejor valor de aptitud se encuentra resaltado en negritas.

	Vec. Mutante		***Vec. de Prueba***			***Vec. Solución***		
i	$V^1_{i,1}$	$V^1_{i,2}$	$U^1_{i,1}$	$U^1_{i,2}$	$f(\vec{U}^1_i)$	$X^1_{i,1}$	$X^1_{i,2}$	$f(\vec{X}^1_i)$
1	0.7644	11.2140	0.7644	11.2140	14.1066	9.8997	-1.8142	9.6424
2	14.6232	6.6300	14.6232	6.6300	34.1843	14.6232	6.6300	34.1843
3	7.9496	15.1040	7.9496	15.1040	39.4810	13.2968	2.5088	23.6673
4	0.0019	-2.8682	0.0019	-2.8682	-0.2360	0.0019	-2.8682	**-0.2360**
5	0.7644	11.2140	1.7725	11.2140	14.1578	1.7725	8.7352	6.7210
6	18.3215	19.2922	18.3215	19.2922	133.3045	-1.0062	-0.3894	-0.0108

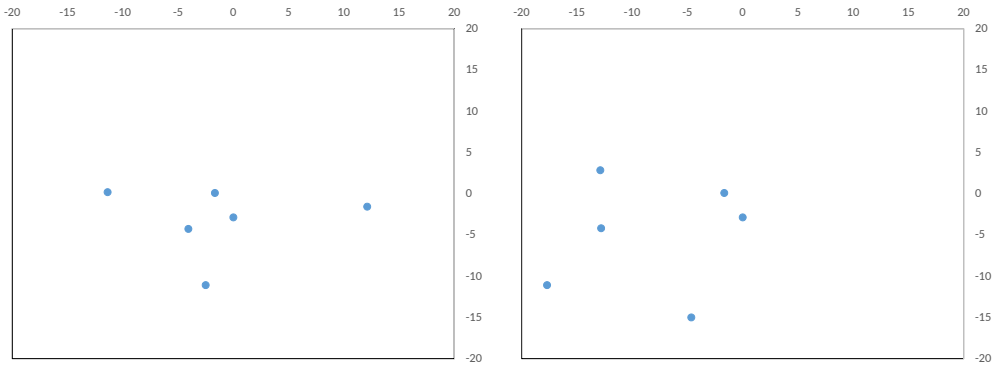
Por ejemplo, para generar el vector mutante $\vec{V}^1_1$ siendo $r_1 = 6$, $r_2 = 2$ y $r_3 = 3$ se obtendría:
$V^1_{1,1} = X^0_{6,1} + F \times (X^0_{2,1} - X^0_{3,1}) = -1.0062 + 0.9 \times (15.2641 - 13.2968) = 0.7644$
$V^1_{1,2} = X^0_{6,2} + F \times (X^0_{2,2} - X^0_{3,2}) = -0.3894 + 0.9 \times (15.4015 - 2.5088) = 11.2140$
$\vec{V}^1_1 = (0.7644, 11.2140)$
Al crear el vector mutante $\vec{V}^1_2$ se emplearon los valores: $r_1 = 4$, $r_2 = 3$ y $r_3 = 6$; en el caso del vector mutante $\vec{V}^1_3$ se emplearon los valores: $r_1 = 2$, $r_2 = 5$ y $r_3 = 1$. Este último caso es interesante, ya que genera un vector mutante fuera del espacio de búsqueda:
$V^1_{3,1} = X^0_{2,1} + F \times (X^0_{5,1} - X^0_{1,1}) = 15.2641 + 0.9 \times (1.7725 - 9.8997) = 7.9496$
$V^1_{3,2} = X^0_{2,2} + F \times (X^0_{5,2} - X^0_{1,2}) = 15.4015 + 0.9 \times (8.7352 - (-1.8142)) = 24.8960$
$\vec{V}^1_3 = (7.9496, 24.8960)$
Como puede apreciarse, al momento de generar el vector mutante $\vec{V}^1_3$, su componente $V^1_{3,2}$ sale del espacio de búsqueda, pues toma un valor mayor a su límite superior que es 20. Cuando esto sucede se debe aplicar algún método para el manejo de restricciones de límite. En este caso, aplicando el método *Reflection*, que se explicará más adelante, en la Sección 3.2.2, esta componente se regresaría la misma magnitud que salió, esto es 4.8960, quedando finalmente su valor en 15.1040 (20-4.8960=15.1040). En la Tabla 2.5 se puede observar, resaltado en amarillo, el valor corregido de la componente $V^1_{3,2}$.

Después de haber generado todos los vectores mutantes, se aplica el operador de cruza, como se indica en la Ecuación 2.13, para generar los vectores de prueba.

Para esto se generan dos valores aleatorios; el primero ($rand_j$) es un valor real entre 0 y 1, y el segundo (j_{rand}) es un entero entre 1 y D (1 y 2 en este caso); por cada componente de cada vector mutante.

Cada vector de prueba toma, en su mayoría[12], los valores de los vectores mutantes, a menos que el primer número aleatorio sea mayor que CR y el segundo sea igual a la dimensión de la componente que se esté procesando, en cuyo caso tomaría el valor del vector solución de la generación anterior.

[12]En este caso, ya que $CR = 0.9$, los vectores de prueba serán similares, en un 90 % a los vectores mutantes.



(a) Generación 2. (b) Generación 3.

Figura 2.9: Posiciones de los vectores solución en el ejemplo con DE para las generaciones 2 y 3.

En la tabla 2.5 se puede observar que sólo la primer componente del quinto vector de prueba $U^1_{5,1}$, resaltada en amarillo, toma su valor de la primer componente del quinto vector solución de la generación anterior $X^0_{5,1}$ (Tabla 2.4), el resto de las componentes toman sus valores de los vectores mutantes.

Una vez que se tienen los vectores de prueba se calcula su valor de aptitud aplicando la Ecuación 2.8, como se explicó anteriormente. En la tabla 2.5 se observan los vectores de prueba con su valor de aptitud, este valor de aptitud es necesario para aplicar el operador de selección descrito en la Ecuación 2.14.

El operador de selección, compará el valor de aptitud de cada vector solución de la generación anterior, con el valor de aptitud de su correspondiente vector de prueba en la generación actual. El operador de selección elegirá, ya sea el vector de prueba o vector solución de la generación anterior, dependiendo de cual vector tenga el mejor valor de aptitud.

A la derecha de la Tabla 2.5 se pueden apreciar los vectores solución seleccionados como sobrevivientes; en este caso se observa que cuatro de ellos fueron seleccionados de los vectores solución de la generación anterior y solo dos (resaltados en amarillo) de los vectores de prueba. En la Figura 2.8(b) se pueden apreciar las posiciones de los vectores solución en la generación 1.

Este proceso de aplicar el operador de mutación, el de cruza y selección se repite hasta que se cumpla una condición de paro, para el caso del presente ejemplo, el algoritmo se repite hasta completar 24 evaluaciones en la función de aptitud; como se tiene una población de 6 vectores solución, considerando las evaluaciones de la generación 0, la condición de paro se cumple en la generación 3. En la Figura 2.9 se pueden observar las posiciones de los vectores solución en las generaciones 2 y 3.

2.3. Métodos para el manejo de restricciones funcionales

Los dos algoritmos de optimización inspirados en la naturaleza (PSO y DE) presentados en la sección anterior, y en general todo este tipo de algoritmos de optimización carecen de mecanismos para tratar con las restricciones funcionales.
Actualmente existe una cantidad considerable de investigaciones que abordan los métodos para el manejo de restricciones funcionales, en esta sección se presentará una revisión de los más relevantes.
Los métodos para el manejo de restricciones funcionales pueden ser agrupados en cuatro categorías [37]: Funciones de penalización, Decodificadores, Métodos que emplean operadores especiales y métodos que separan la función objetivo y las restricciones.

2.3.1. Funciones de penalización

La idea central de los métodos que emplean funciones de penalización es transformar un problema de optimización restringido en un problema no restringido agregando (en un problema de minimización) un valor de penalización a la función objetivo, como se expresa en la siguiente ecuación:

$$\phi(\vec{X}) = f(\vec{X}) + p(\vec{X}) \tag{2.15}$$

donde $\phi(\vec{X})$ representa a la función objetivo expandida y $p(\vec{X})$ es el el valor de penalización añadido, el cual que puede ser calculado como:

$$p(\vec{X}) = \sum_{j=1}^{J} r_j \cdot max(0, g_j(\vec{X}))^2 + \sum_{k=1}^{K} c_k \cdot \left| h_k(\vec{X}) \right| \tag{2.16}$$

en donde $g_j(\vec{X})$ representa a la j-ésima restricción de desigualdad $j = 1,...,J$ y $h_k(\vec{X})$ representa la k-ésima restricción de igualdad $k = 1,...,K$, r_i y c_j son valores constantes, positivos, llamados "factores de penalización". El objetivo de estos métodos es que la función objetivo arroje un valor mayor (en un problema de minimización) para las soluciones no factibles favoreciendo de esta forma la selección de soluciones factibles [7].
La implementación de las funciones de penalización es muy simple, sin embargo, los factores de penalización requieren de un ajuste bastante fino, que depende mucho del tipo de problema a tratar. Los factores de penalización se suelen ajustar de forma estática, dinámica, adaptativa, coevolucionada o difusa [37].

2.3.2. Decodificadores

Estos métodos mapean la región factible del espacio de búsqueda en un espacio más fácil de muestrear donde un algoritmo de optimización puede proporcionar un mejor rendimiento [29].
El proceso de mapeo de un decodificador debe garantizar que cada solución factible en el espacio de búsqueda se incluya en el espacio decodificado y que una solución decodificada corresponda a una solución factible en el espacio de búsqueda. La implementación real de este tipo de métodos no es trivial e implica un alto costo computacional, por tales razones no son muy utilizados en la actualidad [37].

2.3.3. Operadores especiales

Un operador especial es un mecanismo que permite, ya sea conservar la factibilidad de una solución o moverse dentro de una región específica de interés dentro del espacio de búsqueda [49]. Este tipo de operadores, aunque han reportado resultados altamente competitivos el algunos casos, lo cierto es que la mayoría de ellos requieren de un proceso de inicialización ad-hoc o al menos una solución factible o parcialmente factible en la población inicial, lo cual puede ser computacionalmente costoso de obtener cuando se trata de problemas de optimización altamente restringidos.

2.3.4. Separación de la función objetivo y las restricciones

En este tipo de métodos se mantiene el valor de la función objetivo y el de las restricciones de manera separada, al contrario de la función de penalización que los combina. Se ha observado que la separación de restricciones y la función objetivo generan una pérdida importante de diversidad. Por lo tanto, es importante emplear mecanismos para mantener la diversidad con este tipo de métodos, sin embargo, esto no ha evitado que este tipo de métodos se mantengan como los más populares.
Uno de los métodos, de este tipo, que goza de una mayor popularidad, además de ser muy efectivo y de uso en la actualidad, fue propuesto originalmente por Deb [10]. Este enfoque permite conocer si una solución es mejor a otra, aplicando las siguientes reglas:

1. Entre dos soluciones factibles, se elige como mejor solución a la que tenga el valor más bajo en su función objetivo (cuando el problema es de minimización).

2. Entre una solución factible y otra infactible, la mejor solución es la factible.

3. Entre dos soluciones infactibles la mejor será aquella con la menor suma de violaciones de las restricciones.

La suma de violaciones de las restricciones se puede calcular de la siguiente manera:

$$sp(\vec{X}) = \sum_{j=1}^{J} max(0, g_j(\vec{X}))^2 + \sum_{k=1}^{K} \left| h_k(\vec{X}) \right| \tag{2.17}$$

donde los valores de cada restricción de desigualdad $g_j(\vec{X}), j = 1,2\ldots,J$ y cada restricción de igualdad $h_k(\vec{X}), k = 1,2\ldots,K$ se normalizan[13].

Resumen

En este capítulo se ha presentado el concepto de optimización numérica restringida y los métodos empleados en su solución.

Se explicaron a fondo dos algoritmos de optimización inspirados en la naturaleza: optimización por cúmulos de partículas y evolución diferencial. Se observó que ambos algoritmos, y en general los algoritmos de optimización inspirados en la naturaleza carecen de mecanismos para tratar con las restricciones funcionales, algunos de tales mecanismos también fueron abordados en la última sección de este capítulo.

Para un mejor entendimiento de los algoritmos PSO y DE se presentó un ejemplo de cada uno de ellos. En el ámbito de estos dos ejemplos, se observó que las soluciones candidatas suelen abandonar el espacio de búsqueda, debido a la aplicación de operadores de variación.

Para controlar que las soluciones candidatas no salgan del espacio de búsqueda o vuelvan a reubicarse en él, es necesario emplear algún método para el manejo de restricciones de límite, estos métodos se describirán en el próximo capítulo.

[13] A diferencia de la ecuación 2.17, en este caso no existen factores de penalización como r_i y c_j; ya que para conocer si una solución es mejor que otra se emplea otro aspecto como las reglas de Deb; la suma de violaciones de restricciones $sp(\vec{X})$ se utiliza solo para conocer si la solución es factible ($suma = 0$) o no lo es.

Capítulo 3

Métodos para el manejo de restricciones de límite

Como se ha mencionado anteriormente, los algoritmos de optimización inspirados en la naturaleza utilizan operadores de variación para explorar el espacio de búsqueda a fin de encontrar una solución óptima, tales operadores pueden generar vectores solución que caen fuera del espacio de búsqueda produciendo de esta manera un vector solución inválido.
Cuando se genera un vector solución inválido, o para evitar que se generen, es necesario aplicar algún método para el manejo de restricciones de límite.
El objetivo de los métodos para el manejo de restricciones de límite (BCHM) consiste en controlar que la búsqueda se realice en el espacio válido [1], ya sea evitando que las soluciones salgan del espacio, reubicando las que salen o asegurándose que vuelvan a reintroducirse en futuras generaciones.
En el presente capítulo se hace una revisión de los trabajos relacionados con los BCHMs, se describe el funcionamiento de los principales métodos para el manejo de posiciones y las estrategias comúnmente empleadas en el manejo de la velocidad.

3.1. Trabajos relacionados

La mayoría del trabajo de investigación relacionado con el estudio de los métodos para el manejo de restricciones de límite (*Boundary Constraint-Handling Methods, BCHM*) en algoritmos de optimización inspirados en la naturaleza ha sido abordado desde los algoritmos de Optimización por Cúmulos de Partículas (*PSO*) y Evolución Diferencial (*DE*), siendo éstos los algoritmos de

[1] Dentro del espacio de búsqueda, delimitado por los límites inferior y superior de las variables de diseño.

optimización más populares.
De estos dos algoritmos, la mayor parte de publicaciones se centra en PSO, debido a que es muy común que después de actualizar la posición de las partículas en cada generación, estas puedan abandonar el espacio de búsqueda fácilmente, especialmente en las primeras generaciones [5].

3.1.1. Manejo de límites en Optimización por Cúmulos de Partículas

Uno de los primeros trabajos en BCHM aplicados a PSO fué publicado por Zhang et al.[64], quienes propusieron un método llamado *Periodic*, este método considera al espacio de búsqueda como un espacio toroidal, de manera que si una partícula sale del espacio de búsqueda por el límite superior es reinsertada por el límite inferior. En el estudio los autores comparan su rendimiento contra los métodos *Boundary* y *Random*. Los autores destacan que el método *Periodic* obtuvo el mejor desempeño, sin embargo, también se observó que el método *Boundary* presentó un buen rendimiento en problemas en los cuales el óptimo se encontraba cerca de los límites del espacio de búsqueda.
En el caso del algoritmo PSO se puede manipular la posición de las partículas que salen del espacio de búsqueda, pero también es factible modificar su velocidad. Los BCHMs híbridos en PSO modifican tanto la velocidad como la posición. Uno de los primeros métodos híbridos, llamado *Damping* fué propuesto por Huang and Sanagavarapu [22], el cual modifica tanto la posición como la velocidad de las particulas que salen del espacio de búsqueda. En este trabajo, los autores compararon el rendimiento de *Damping* contra tres métodos: *Invisible*, *Boundary* y *Reflection* en el algorítmo PSO, con la finalidad de resolver dos problemas de optimización no restringida[2]. El buen desempeño de Damping, en comparación con el de los métodos de prueba demostró la importancia de este tipo de mecanismo híbridos en PSO, en donde adicionalmente al manejo de la posición de las partículas, se vuelve necesario modificar su velocidad, evitando de esta manera que vuelvan a salir en futuras generaciones por efecto de la inercia.
Los autores Xu y Rahmat [61] clasifican los BCHMs en dos grupos: restringidos y no restringidos, los primeros reubican las partículas errantes dentro del espacio de búsqueda, mientras que los segundos no lo hacen. En este estudio comparativo se observó nuevamente que el método *Damping*, un método restringido híbrido, presentó el mejor rendimiento. Dentro de los no restringidos el mejor rendimiento lo presentó el método Invisible, en este método se permite que las partículas salgan del espacio de búsqueda, sin embargo, se les asigna un valor de aptitud muy alto (en caso de un problema de minimización), de tal forma que la atracción de *pbest* y *gbest* podrán eventual-

[2]Comúnmente se indica que un problema es *no restringido* cuando no incluye *restricciones funcionales*; esto no implica que incluya o no *restricciones de límite*.

mente volver a reintroducir las partículas en el espacio de búsqueda. Otro estudio experimental fue publicado por Shi et al. [52], en este trabajo los autores analizaron el efecto de aplicar varios BCHMs en PSO con diferentes topologías (centrado en la diversidad de población) para resolver problemas de optimización no restringida. Sin importar la topología empleada, se observó que el método para el manejo de restricciones de límite empleado representó una diferencia en la calidad de los resultados obtenidos.

Chu et al. [5] realizaron varios experimentos para investigar los efectos de los métodos para el manejo de restricciones de límite (BCHM) dentro del algoritmo PSO para resolver problemas de alta dimensionalidad. En este trabajo se probaron tres BCHMs básicos utilizando un conjunto de funciones estándares y de composición. Los autores llegaron a la conclusión de que el efecto del BCHM empleado se vuelve crítico cuando se trata de problemas de alta dimensión, ya que al aumentar la dimensionalidad la mayoría de las partículas del enjambre tiende a volar fuera del espacio de búsqueda.

Helwing et al. [21] compararon una amplia variedad de BCHMs para PSO examinando su rendimiento en paisajes planos. En general, los autores concluyeron que la mayoría de los métodos introducen un sesgo significativo en el proceso de búsqueda de PSO, afectando por tanto la calidad de las soluciones encontradas.

Padhye et al. [42] realizaron una revisión de los BCHMs empleados dentro del algoritmo de PSO. En este trabajo los autores propusieron dos nuevos métodos para el manejo de restricciones de límite, los cuales mostraron ser robustos y consistentes en términos de rendimiento en varios escenarios de simulación que incluían problemas de optimización restringida. Los autores mencionan que el rendimiento del BCHM depende del tipo de problema y de la ubicación de las soluciones óptimas dentro del espacio de búsqueda.

Recientemente Juárez-Castillo et al. [25], publicaron un estudio empírico en donde se comparó el rendimiento de treinta y cinco BCHMs híbridos para problemas de optimización numérica restringida dentro del algoritmo PSO, en este trabajo se encontró que el mejor rendimiento se obtuvo con un método híbrido que modifica la posición de las partículas reubicándolas en posiciones sesgadas hacia espacios más promisorios y su velocidad mediante *Deterministic Back*.

3.1.2. Manejo de límites en Evolución Diferencial

Adicionalmente a los trabajos sobre los BCHMs dentro del algoritmo de optimización por cúmulos de partículas (*PSO*), la mayor parte de publicaciones relacionadas con el manejo de restricciones de límite han sido abordadas desde el algoritmo de Evolución Diferencial (*DE*).

Originalmente, Price et al. [43] sugirieron que reemplazar una solución no factible por una gene-

rada al azar es el enfoque más imparcial para DE. Los autores también definieron una estrategia a la que denominaron *bounce back*, en donde una solución "y" no factible, generada al mutar una solución factible "x", es reemplazada por una nueva solución factible ubicada en una línea entre "x" y "y".

El método propuesto por Ronkkonen et al. [47] funciona de forma similar a *bounce back*, donde las soluciones no factibles son reflejadas (*reflected*) hacia atrás del límite violado por una magnitud igual a la de la violación. Otro método fue publicado por Brest et al. [4], en donde el componente de un vector mutante que sale del espacio de búsqueda se establece en el límite (*bound*) violado.

En un estudio realizado por Arabas et al. [2] se comparó el rendimiento de los métodos: *Random*, *Reflection*, *Boundary*, *Conservatism*, *Wrapping* y *Resampling* [3] usando el algoritmo canónico de Evolución Diferencial (*DE*) para resolver los problemas de CEC2005. Los autores demostraron que el método *Resampling* obtuvo los mejores resultados en la mayoría de las funciones y concluyeron también que la elección del BCHM puede influir significativamente en la calidad de las soluciones encontradas.

Gandomi y Yang [18], propusieron un nuevo método para el manejo de restricciones de límite al que llamaron *Evolutionary*. Este método reposiciona la solución inválida en un lugar ubicado entre el límite violado y la posición del mejor vector de la población actual. Lo interesante de este método es que a medida que el algoritmo de búsqueda encuentra mejores soluciones el método *Evolutionary* también se va *adaptando* en reposicionar las soluciones inválidas.

El método *Evolutionary* se probó usando DE para resolver problemas de optimización numérica no restringida, logrando una buena propiedad de convergencia y mostró un mejor rendimiento para problemas de optimización en comparación con los métodos clásicos para el manejo de límites.

Recientemente Juárez-Castillo et al., [24] propusieron un BCHM que coloca el vector corregido en el centroide formado por un vector de la población actual (ubicado dentro o cerca de la región factible) y k vectores corregidos al azar. Este método fue diseñado específicamente para trabajar en problemas de optimización restringuida y su principal ventaja es que sus vectores reparados están orientados hacia la región factible.

3.1.3. Manejo de límites en otros algoritmos de optimización inspirados en la naturaleza

El estudio de los BCHMs también ha motivado investigaciones basadas en otros algoritmos de optimización como: Interior Search, Evolution Strategies y Cuckoo Search. En este sentido, Wessing [57] propuso dos BCHMs denominados Intersection-Projection (IP) e Intersection-Reflection

[3] Estos métodos se describen a detalle en la Sección 3.2.

(IR), y realizó una comparación contra tres métodos clásicos para el manejo de restricciones de límite: *Boundary*, *Reflection* y *Wrapping*, utilizando una estrategia evolutiva de adaptación de la matriz de covarianza (CMA-ES).

Los autores concluyeron que IR e IP parecen tener una ligera tendencia a trabajar mejor en problemas de baja dimensionalidad. Por otro lado, observaron que cuando el óptimo global se encuentra en el borde o en una esquina, del espacio de búsqueda, es mejor utilizar el método *Boundary*, en la mayoría del resto de los problemas se observó un buen rendimiento al emplear el método *Reflection*. En general, los métodos para el manejo de límites tuvieron un impacto en la calidad de las soluciones encontradas por el algoritmo de optimización.

Gandomi et al. [17], publicaron una comparación entre dos BCHMs (*Boundary* y *Evolutionary*) empleando el algoritmo *Cuckoo Search* para resolver cinco funciones de optimización no restringida, concluyendo que el método *Evolutionary*, gracias a su adaptabilidad a medida que evoluciona el proceso de búsqueda, puede mejorar el rendimiento del algoritmo de optimización sin la necesidad de aditamentos extras que aumentan la complejidad del algoritmo Cuckoo Search.

Recientemente, Trivedi et al. [56] compararon el rendimiento de cinco BCHMs (*Reflect*, *Random*, *Wrapping*, *Boundary*, y *Evolutionary*) empleando el Algoritmo Búsqueda Interior (*Interior Search*) para resolver dieciséis problemas de optimización numérica y concluyeron que el método *Evolutionary* es adecuado para la mayoría de los problemas de optimización mejorando la propiedad de convergencia.

Los métodos de manejo de restricciones de límites también se han usado como soporte en algoritmos evolutivos para resolver problemas de optimización numérica restringida multiobjetivo, como en el trabajo publicado por Liu et al.[33], en donde se diseñó un método de búsqueda de límites para mejorar la eficiencia de un algoritmo evolutivo.

En resumen, los operadores de variación empleados en los algoritmos de optimización inspirados en la naturaleza pueden hacer que algunos vectores solución salgan del espacio de búsqueda haciendo necesaria la aplicación de algún método para el manejo de restricciones de límite (BCHM). La elección de uno u otro BCHM tiene un fuerte impacto en la calidad de las soluciones encontradas por el algoritmo de optimización [2, 21, 52, 57] ya que la mayoría de los BCHM introducen un sesgo significativo en el proceso de búsqueda [21]; por ejemplo, cuando el óptimo global se encuentra cerca de un límite o en una esquina es mejor usar un método como *Boundary* [57] y sería muy inadecuado emplear un método como *Periodic* [64].

La elección del BCHM adecuado se vuelve una decisión crítica en problemas complejos de alta dimensionalidad [5], ya que a medida que aumenta la dimensionalidad es mayor la cantidad de soluciones que abandonan el espacio de búsqueda.

En este sentido, los BCHMs se presentan como un mecanismo mediante el cual podría mejorarse

la calidad de las soluciones encontradas por un algoritmo de optimización [33].
Además de los BCHMs típicos, se observa que características adicionales como hibridación [22, 25] y adaptación [17, 18] pueden hacer que se mejore la capacidad de los métodos para el manejo de límites.
Se observa también que la investigación de BCHMs sobre optimización restringida es escasa, es decir, para resolver problemas de optimización con restricciones funcionales donde las soluciones buscadas se encuentran dentro de la región factible. Este tema ha sido identificado como un camino de investigación que debe ser estudiado, ya que, únicamente en cuatro [24, 25, 33, 42] de los trabajos mencionados se hicieron pruebas con problemas restringidos.
En la presente sección se observó también que la mayor parte de la investigación en BCHMs ha sido abordada desde los algoritmos de PSO y DE, en ambos casos es necesario volver a ubicar el vector solución no válido dentro del espacio de búsqueda; sin embargo, en el caso del algoritmo de optimización por cúmulos de partículas, también se recomienda modificar la velocidad. Como resultado de esta conclusión, en la siguiente sección se explican a detalle los métodos comúnmente usados para modificar la posición y posteriormente, las estrategias comúnmente usadas para actualizar la velocidad de las partículas dentro del algoritmo PSO.

3.2. Métodos de manejo de posición

En esta sección se describen los métodos comúnmente empleados para modificar la posición de los vectores solución que salen del espacio de búsqueda.

3.2.1. Evolutionary (Evo)

Este método, originalmente diseñado para trabajar con Evolución diferencial [18], reemplaza los componentes que están más allá de los límites a través de un componente aleatorio entre el límite relacionado y el componente similar de la mejor solución hasta el momento. Este método fue formulado como en la Ecuación 3.1.

$$X_j^c = \begin{cases} \alpha \times l_j + (1-\alpha) X_j^{best} & \text{si } X_j < l_j \\ \beta \times u_j + (1-\beta) X_j^{best} & \text{si } X_j > u_j \end{cases} \tag{3.1}$$

dónde X_j^{best} es el valor de la variable j de la mejor solución hasta ahora y α y β son números aleatorios entre 0 y 1. A menor valor de α y β, los componentes reemplazados serán ubicados más cerca del límite violado; y a mayor valor, los componentes serán ubicados más cerca de la mejor solución.

3.2.2. Reflection (Ref)

En este método, las variables que violan los límites (superior o inferior) se reflejan desde el límite violado, por la cantidad de violaciones [46, 47]. Esta propuesta está formulada en la Ecuación 3.2.

$$X_j^c = \begin{cases} X_j & \text{si } l_j \leq X_j \leq u_j \\ 2 \times l_j - X_j & \text{si } X_j < l_j \\ 2 \times u_j - X_j & \text{si } X_j > u_j \end{cases} \tag{3.2}$$

dónde X_j^c es el valor válido, X_j es el valor que infringe la restricción de límite, l_j y u_j son los límites inferior y superior de la variable j, respectivamente. Este proceso se repite hasta que el valor esté dentro de los límites.

3.2.3. Random (Ran)

Esta técnica reemplaza los valores de las variables que están fuera de sus límites por valores aleatorios dentro de los límites inferior y superior [31, 43] a través de la Ecuación 3.3,

$$X_j^c = l_j + rand(0,1) \times (u_j - l_j) \tag{3.3}$$

dónde $rand(0,1)$ devuelve un valor aleatorio real entre 0 y 1 con distribución uniforme.

3.2.4. Wrapping (Wra)

En esta técnica, el espacio de búsqueda es *envuelto* en cada dimensión [44], es decir se simula que el espacio de búsqueda de cada variable tiene una forma periódica y que por tanto se puede tratar como un espacio toroidal [64]. Por esta razón, los valores que salen del espacio de búsqueda por el límite superior se insertan por el límite inferior a través de la Ecuación 3.4.

$$X_j^c = \begin{cases} X_j & \text{si } l_j \leq X_j \leq u_j \\ u_j - (l_j - X_j)\,\%\rho & \text{si } X_j < l_j \\ l_j + (X_j - u_j)\,\%\rho & \text{si } X_j > u_j \end{cases} \tag{3.4}$$

donde $\%$ es el operador de módulo y $\rho = |u_j - l_j|$ representa el rango de la variable.

3.2.5. Boundary (Bou)

En esta técnica, también conocida como *Projection*, el valor de una variable se restablece al límite violado [4, 64]. Esto se expresa en la Ecuación 3.5.

$$X_j^c = \begin{cases} X_j & \text{si } l_j \leq X_j \leq u_j \\ l_j & \text{si } X_j < l_j \\ u_j & \text{si } X_j > u_j \end{cases} \tag{3.5}$$

en donde X_j^c representa el valor válido, X_j es el valor que viola los límites de la variable, l_j y u_j representan los límites inferior y superior de la variable j respectivamente.

3.2.6. Conservatism (Con)

Este método se propuso particularmente para trabajar con Evolución Diferencial. Si la mutación diferencial genera un vector inviable, se rechaza y el vector padre original permanece en la población. Este método asume que el vector no factible no será reparado, pero es rechazado en la etapa de reemplazo [16].

3.2.7. Infinity (Inf)

Este método, también conocido como *Invisible* [61], es similar a *Conservatism*, pero en su versión especializada para PSO. En este método se permite que las partículas abandonen el espacio de búsqueda sin modificar su posición, sin embargo, su valor de aptitud se modifica para asignar el valor $+\infty$ (para problemas de minimización), por lo tanto, se espera que la atracción de las posiciones de *pbest* y *gbest* eventualmente hará que la partícula entre nuevamente al espacio de búsqueda [36]. Bratton y Kennedy han propuesto el uso de *Infinity* como una metodología estándar para PSO [3].

3.2.8. Resampling (Res)

Este método trabaja aplicando el operador de variación (mutación diferencial para DE) hasta que se genere un vector válido completo. Si al menos una variable infringe los límites, el operador se aplica nuevamente para generar una solución completamente nueva, este proceso se repite hasta obtener una solución válida [2].

Existe también una versión del método Resampling para PSO, en la cual se vuelven a muestrear las variables r_1 y r_2 (de la Ecuación 2.5) hasta que la ubicación de la partícula cae dentro del espacio de búsqueda, sin embargo, este método no se ha considerado en el presente trabajo debido

a que requiere una cantidad enorme de remuestreos para las variables r_1 y r_2 antes de generar una partícula válida y comúnmente las soluciones reparadas no son de buena calidad [1].

3.3. Estrategias de actualización de velocidad

Cuando una partícula sale del espacio de búsqueda en el algoritmo PSO, adicionalmente a modificar su posición para volver a colocar la partícula dentro del espacio de búsqueda, se suele modificar también su velocidad, ya que de no hacerlo es posible que la partícula vuelva a salirse en las próximas generaciones [1, 6, 21, 40].
En esta sección se describen las estrategias comúnmente empleadas en PSO para modificar la velocidad de las partículas que salen del espacio de búsqueda.

3.3.1. None (Non)

En esta estrategia, la velocidad de la partícula que sale del espacio de búsqueda permanece sin cambios [6, 21].

3.3.2. Absorb Zero (Aze)

Cuando una partícula sale del espacio de búsqueda, su velocidad se establece a cero en la dimensión respectiva en la que la partícula salió del espacio de búsqueda [6, 21].

$$V_j^c = 0 \tag{3.6}$$

donde V_j^c es la j-ésima componente corregida del vector velocidad $\vec{V}$.

3.3.3. Deterministic Back (Dba)

En esta estrategia la velocidad se invierte para obligar a la partícula a volver al espacio de búsqueda en la siguiente iteración multiplicándola por $-\lambda$ para $\lambda > 0$. Un valor típico para λ is 0.5 [6, 21].

$$V_j^c = -\lambda V_j \tag{3.7}$$

3.3.4. Random Back (Rba)

Esta estrategia es similar a *Deterministic Back*, pero en este caso el valor λ se genera uniformemente al azar en el rango de $[0,1]$ [6, 21].

3.3.5. Adjust (Adj)

En este caso, la velocidad se calcula mediante la diferencia entre la nueva posición y la posición anterior [20].

$$V_j^c = X_j^g - X_j^{g-1} \tag{3.8}$$

donde X^g es la posición de la partícula en la generación g.

Resumen

En el presente capítulo se ha hecho hincapié en la importancia que tienen los métodos para el manejo de restricciones de límite en la calidad de las soluciones encontradas por los algoritmos de optimización, por otro lado, características como la hibridación y la adaptación pueden hacer que los BCHMs obtengan un mejor rendimiento. En este capítulo se describieron también los principales métodos para el manejo de posiciones y las estrategias comúnmente empleadas en el manejo de la velocidad de los vectores solución que salen del espacio de búsqueda.

Capítulo 4

Propuestas

En esta sección se presentan tres propuestas, en la primera se presenta el *Esquema Adaptativo para el Manejo de Restricciones de Límite*, el cual a su vez emplea varios métodos, dos de ellos (*Centroid* y *Res&Ran*) son propuestos también en el presente trabajo.
En la segunda propuesta se describe el método *Centroid*, el cual tiende a guiar los vectores reparados hacia una zona factible, y finalmente, la tercera propuesta describe al método híbrido *Res&Ran*, el cual combina los métodos *Random* y *Resampling*, estableciendo a su vez un límite de remuestreos.

4.1. Esquema Adaptativo para el Manejo de Restricciones de Límite (*ABCHS*)

En esta sección se propone un Esquema Adaptativo para el Manejo de Restricciones de Límite (Adaptive Boundary Constraint-Handling Scheme, *ABCHS* por sus siglas en inglés), en el cual, en lugar de emplear un único método para manejar las restricciones de límite, se cuenta con un conjunto de ellos.
Sea $S_{bchm} = \{bchm_1, bchm_2, ..., bchm_k\}$ un conjunto de k métodos para el manejo de restricciones de límite y sea $P = \{p_1, p_2, ..., p_k\}$ un conjunto de probabilidades asociadas a los métodos, de modo que p_j es la probabilidad de aplicar el método $bchm_j$.
Las probabilidades iniciales se establecen a $1/k$, es decir, en el caso de tener un conjunto de cuatro métodos: $p_1 = p_2 = p_3 = p_4 = 0.25$, por lo tanto, al inicio cada método tiene la misma probabilidad de ser aplicado.
De entre los k métodos considerados, el método $bchm_1$ tiene la particularidad de beneficiar a la exploración del espacio de búsqueda, como los métodos *Random* o *Resampling*. Este método $bchm_1$

se aplica cuando el *Número de soluciones factibles* (*NFS*) es igual a cero; de lo contrario se elige alguno de los k métodos empleando el método de selección por ruleta[65]. El proceso que se sigue se aprecia en la Figura 4.1.

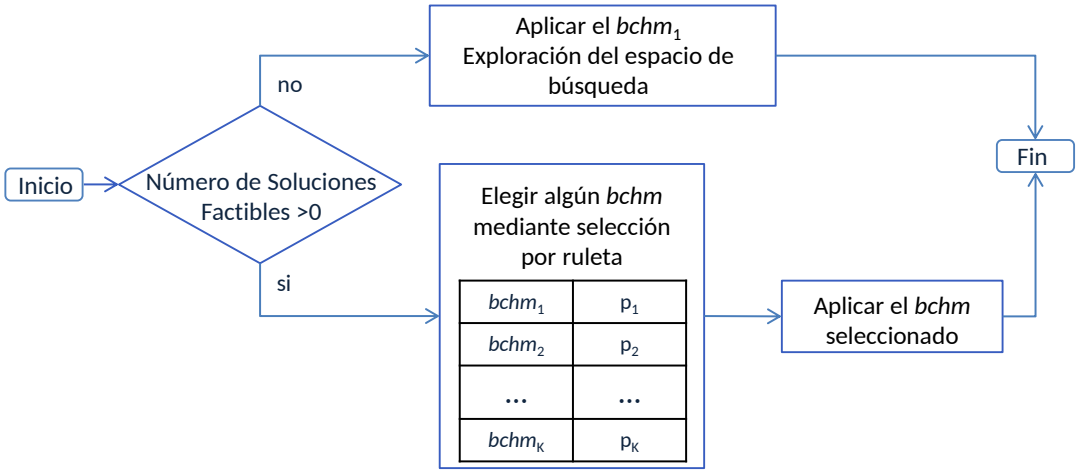


Figura 4.1: Esquema adaptativo para el manejo de restricciones de límite. Al inicio, cuando aún no existen soluciones factibles se aplica el $bchm_1$, que beneficia la exploración del espacio de búsqueda, posteriormente se elige uno de los k métodos mediante selección por ruleta, considerando sus probabilidades asociadas.

El Algoritmo 3, en su primer sección (líneas 1 a 7) describe el proceso que se realiza cada vez que es necesario reparar un vector solución no válido.

Las probabilidades asociadas a los métodos de manejo de restricciones de límites se actualizan cada cierto número de generaciones o *Periodo de aprendizaje*. El Periodo de aprendizaje (*LP*) se calcula de la siguiente manera:

$$LP = nint(0.5 \times D) + 2 \tag{4.1}$$

donde *nint* devuelve el *entero más cercano* a $(0.5 \times D)$ y D representa el número de dimensiones del problema; por ejemplo, el período de aprendizaje para un problema de seis dimensiones será de cinco generaciones.

Después de evaluar todas las soluciones de la población en cada generación, se calcula el número de soluciones reparadas por el método j que fueron mejores que su solución objetivo (target vector) correspondiente, y el número de soluciones que fueron peores; estos valores se acumulan en rsB_j (*best repaired solutions*) y rsW_j (*worst repaired solutions*) respectivamente, en un periodo de *LP*

generaciones. Los valores rsB_j y rsW_j se usan para actualizar la probabilidad p_j de aplicar $bchm_j$ de la siguiente manera:

$$p_j = \frac{S_j}{\sum_{i=1}^{k} S_i} + \zeta \tag{4.2}$$

donde:

$$S_j = \frac{rsB_j}{rsB_j + rsW_j + \zeta} \tag{4.3}$$

donde $\zeta = 0.01$ es un pequeño valor de tolerancia usado en la Ecuación 4.3 para evitar la división por cero y en la Ecuación 4.2 evita descartar prematuramente cualquier método cuando obtenga una probabilidad de cero.

El Algoritmo 3, en su Sección 2 (líneas de la 9 a la 30) describe la forma en que se actualizan las probabilidades en cada periodo *LP*, así como la forma en que los valores *rsB* y *rsW* se acumulan en cada generación. Esta sección debe ejecutarse al final de cada generación.

Este esquema adaptativo puede integrarse en cualquier algoritmo de optimización inspirado en la naturaleza donde se requiera el uso de algún método para el manejo de restricciones de límite a

```
1:  //Sección 1. Se aplica cuando es necesario reparar una solución no válida, permite elegir qué método aplicar.
2:  si NFS = 0 entonces
3:      Aplicar bchm_1
4:  si no
5:      Seleccionar bchm_s : bchm_s ∈ S_bchm aplicando selección por ruleta.
6:      Aplicar bchm_s
7:  fin si
8:
9:  //Sección 2. Se ejecuta al final de cada generación. Calcula las nuevas probabilidades cada "LP"generaciones
10: // y actualiza los valores rsB y rsW.
11: si NFS > 0 entonces
12:     si (g % LP) = 0 entonces {// % representa al operador módulo.}
13:         S_j = rsB_j / (rsB_j + rsW_j + ζ), ∀j ∈ {1,...,k}
14:         p_j = S_j / (Σ_{i=1}^{k} S_i) + ζ, ∀j ∈ {1,...,k}
15:         rsB_j = 0, ∀j ∈ {1,...,k}
16:         rsW_j = 0, ∀j ∈ {1,...,k}
17:     si no
18:         para i = 1 hasta NP hacer
19:             para j = 1 hasta k hacer
20:                 si X_i^g fue reparado por bchm_j entonces
21:                     si f(X_i^g) <= f(X_i^{g-1}) entonces
22:                         rsB_j = rsB_j + 1
23:                     si no
24:                         rsW_j = rsW_j + 1
25:                     fin si
26:                 fin si
27:             fin para
28:         fin para
29:     fin si
30: fin si
```

Algoritmo 3: Esquema Adaptativo para el Manejo de Restricciones de Límite (*ABCHS*).

fin de reparar las soluciones que salen del espacio de búsqueda. La única condición es tener un conjunto de k métodos para el manejo de restricciones de límite.

Con base en pruebas experimentales se observó que cuando se emplearon entre tres y cinco métodos se obtuvieron los mejores resultados, por lo tanto, en el presente trabajo se han considerado conjuntos de cuatro métodos, es decir $k = 4$.

La Tabla 4.1 presenta los conjuntos de métodos para el manejo de restricciones de límite, los cuales recomendamos usar con los algoritmos de Evolución Diferencial y Optimización por Cúmulos de Partículas. Estos conjuntos de métodos son los que se emplearán en la etapa de experimentación.

En el caso de Evolución Diferencial, se emplea *Res&Ran* como primer método, mismo que se propone en la sección 4.3. *Res&Ran* es un método híbrido que elimina las debilidades del método *Resampling* propuesto por Arabas, Szczepankiewicz y Wroniak [2], este método es conocido por su buen rendimiento en DE y por apoyar la exploración del espacio de búsqueda, por esta razón se ha considerado como $bchm_1$.

Los tres métodos restantes para el manejo de restricciones de límite en Evolución Diferencial son: *Centroid*, *Reflection* y *Wrapping*. El método *Centroid* también es propuesto en el presente trabajo, en la sección 4.2, la característica primordial de *Centroid* es dirigir las soluciones reparadas hacia una región factible.

En el caso de la Optimización por Cúmulos de Partículas, los cuatro métodos son híbridos, pues como ha sido publicado recientemente [25] la mejor opción para manejar las restricciones de límite en PSO, es modificar tanto la posición como la velocidad de las partículas que salen del espacio de búsqueda.

En los métodos presentados en la columna PSO de la Tabla 4.1, la primera parte representa el método que manipula la posición de la partícula y la segunda parte se refiere a la estrategia utilizada en la modificación de su velocidad, por ejemplo, *Ran&RaB* representa un método híbrido que modifica la posición utilizando el método Random (*Ran*) y la velocidad a través de Random Back (*RaB*).

El híbrido *Ran&RaB* se considera como el primer método ($bchm_1$) para *PSO* debido a que su com-

Tabla 4.1: Conjuntos de métodos para el manejo de restricciones de límite recomendados para usar con Evolución Diferencial y Optimización por Cúmulos de partículas.

S_{bchm}	Evolución Diferencial (*DE*)	Optimización por Cúmulos de partículas (*PSO*)
$bchm_1$	*Res&Ran*	*Ran&RaB*
$bchm_2$	Centroid	*Cen&DeB*
$bchm_3$	Reflection	*Ref&DeB*
$bchm_4$	Wrapping	*Wra&RaB*

portamiento beneficia la exploración del espacio de búsqueda. Los tres métodos restantes para *PSO* son los mismos que para *DE*: *Centroid* (*Cen*), *Reflection* (*Ref*) y *Wrapping* (*Wra*), con el diferencia que en esta ocasión se combinan con Deterministic Back (*DeB*) o Random Back (*RaB*). La estrategia Deterministic Back emplea $\lambda = 0.5$, tal como lo recomiendan Helwig, Branke y Mostaghim [21].

4.2. *Centroid* (*Cen*)

Este método fue diseñado principalmente para problemas de optimización numérica restringida en donde existe una región factible, de tal forma que los vectores reparados serán guiados hacia esta región.

El método *Centroid* reubica los vectores inválidos dentro del espacio de búsqueda, en el centroide de un área formada por $K+1$ vectores, uno de ellos tomado de la población y K vectores reparados de forma aleatoria [26].

El vector de la población sesga la posición del vector corregido a un área con un buen valor de aptitud, mientras que los K vectores aleatorios guian la posición del vector corregido a nuevas áreas en el espacio de búsqueda.

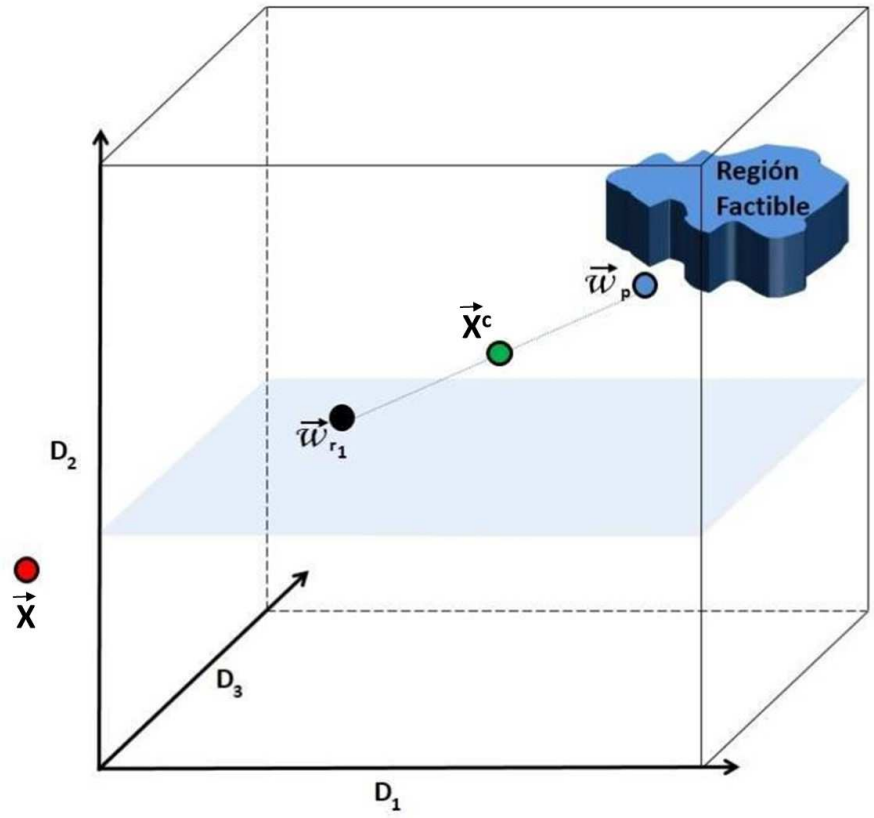
La Ecuación 4.4 describe este método.

$$\vec{X}^c = \begin{cases} \vec{X} & \text{si } \forall X_j : l_j \leq X_j \leq u_j \\ \frac{\vec{W}p + \sum_{i=1}^{K} \vec{W}r_i}{K+1} & \text{en cualquier otro caso} \end{cases} \tag{4.4}$$

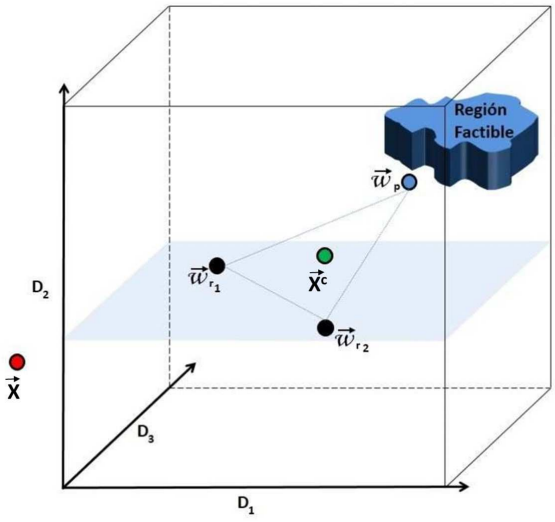
dónde $\vec{X} = [X_1, X_2, ..., X_D]$ es el vector que viola los límites, D representa al número de dimensiones del problema a tratar, $\vec{X}^c$ es el vector corregido, $\vec{W}p$ es un *vector de la población actual* que se selecciona de acuerdo con la *cantidad de soluciones factibles* (*Amount of Feasible Solutions*, AFS) (ver Ecuación 4.5).

$$\vec{W}p = \begin{cases} \vec{X}_{rand} \in SFS & \text{si } AFS > 0 \ \wedge \ rand[0,1] > 0.5 \\ \vec{X}_{best} \in SIS & \text{en cualquier otro caso} \end{cases} \tag{4.5}$$

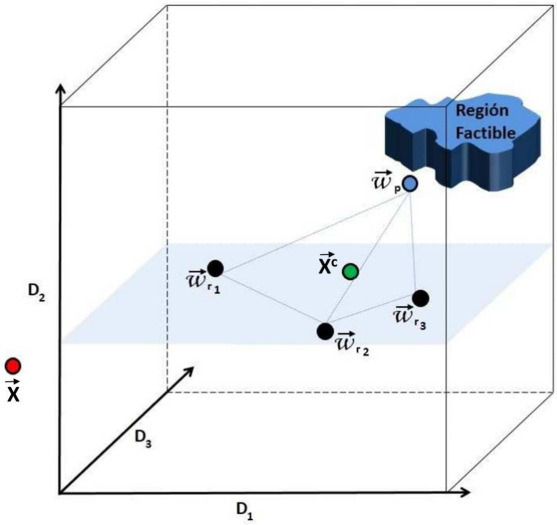
dónde SFS (*Set of Feasible Solutions*) es el *conjunto de soluciones factibles* en la población actual, SIS (*Set of Infeasible Solutions*) es el *conjunto de soluciones no factibles* en la población actual, $\vec{X}_{rand}$ es una *solución factible elegida al azar* y $\vec{X}_{best}$ es la *mejor solución* de SIS, es decir, una solución no factible con la menor violación de restricciones funcionales. Por otra parte, $\vec{W}r_1, ..., \vec{W}r_K$ son K *vectores aleatorios* [24]. Los vectores aleatorios inicialmente toman los mismos valores que



(a) Centroid 1+1.



(b) Centroid 2+1.



(c) Centroid 3+1.

Figura 4.2: Explicación gráfica del método *Centroid* para K=1 (a), K=2 (b) y K=3 (c). En los tres casos los K vectores aleatorios comparten los mismos valores que $\vec{X}$ en la dimensión D_2, que es la dimensión donde $\vec{X}$ mantiene valores válidos, y contiene valores aleatorios en las dimensiones D_1 y D_3. El vector corregido $\vec{X}^c$ se ubica en el centroide formado por los K vectores aleatorios y el vector $\vec{W}p$.

$\vec{X}$, y posteriormente reemplazan sus valores inválidos por valores elegidos de forma aleatoria entre sus límites inferior l y superior u, de la misma manera en que se genera un vector reparado mediante el método *Random*, explicado en la Sección 3.2.3.

El objetivo principal de este método es reubicar el vector corregido cerca de una región prometedora (dentro de la región factible o cerca de ella) pero sesgada por K posiciones aleatorias para evitar una posible convergencia prematura.

La Figura 4.2 representa un problema hipotético de optimización en tres dimensiones (D_1, D_2 y D_3), donde el vector solución $\vec{X}$ ha salido del espacio de búsqueda, tomando valores no válidos en las dimensiones D_1 y D_3. El vector corregido $\vec{X}^c$ se reubica en el centroide formado por los K vectores aleatorios y el vector $\vec{W}p$.

En la Figura 4.2(a), donde $K = 1$, el vector $\vec{X}^c$ se coloca en el centroide formado por $\vec{W}p$ y un solo vector aleatorio $\vec{W}r_1$; en la Figura 4.2(b) se consideran 2 vectores aleatorios $K = 2$; en la Figura 4.2 (c) se muestra la variante de *Centroid* para $K = 3$.

El Algoritmo 4 presenta el pseudocódigo del método *Centroid*. En la primera etapa se selecciona

```
1:  SFS : Conjunto de soluciones factibles
2:  SIS : Conjunto de soluciones no factibles
3:  AFS : Cantidad de soluciones factibles
4:  K : Cantidad de vectores aleatorios
5:  D : Dimensionalidad del problema
6:  l⃗ : Límites inferiores
7:  u⃗ : Límites superiores
8:  X⃗ : El vector que se encuentra fuera de los límites
9:  si AFS > 0 ∧ rand[0,1] > 0.5 entonces
10:     Wp⃗ = X⃗_rand {X⃗_rand se selecciona aleatoriamente de SFS}
11: si no
12:     Wp⃗ = X⃗_best {X⃗_best es el mejor individuo de SIS}
13: fin si
14: {Creación de K vectores aleatorios}
15: para i=1 hasta K hacer
16:     Wr⃗_i = X⃗
17:     para j=1 hasta D hacer
18:         si w_{r_{i,j}} < l_j ∨ w_{r_{i,j}} > u_j entonces
19:             w_{r_{i,j}} = l_j + rand[0,1] × (u_j − l_j)
20:         fin si
21:     fin para
22: fin para
23: X⃗^c = (Wp⃗ + Wr⃗_1 + ... + Wr⃗_K)/(K+1) {Vector corregido}
```

Algoritmo 4: Pseudocódigo del método Centroid.

el vector $\vec{W}p$ de acuerdo con la *cantidad de soluciones factibles AFS*.

Si existen soluciones factibles, se selecciona el vector $\vec{W}p$ de forma aleatoria del *conjunto de soluciones factibles SFS*, de lo contrario se selecciona el vector con la menor cantidad de violaciones funcionales del conjunto de soluciones no factibles *SIS* (líneas 9 a 13).

Posteriormente, se generan los K vectores aleatorios, cada uno de los cuales se inicializa con los

mismos valores de $\vec{X}$, posteriormente se asignan nuevos valores en las posiciones no válidas, estos valores se seleccionan de forma aleatoria entre los límites inferior l y superior u de cada variable. Como se expresa en las líneas 17 a 21 del Algoritmo 4.
Finalmente se calcula el centroide de los K vectores aleatorios y el vector $\vec{W}p$ para conocer la posición del vector corregido $\vec{X}^c$.

4.3. Método híbrido de remuestreo para DE (*Res&Ran*)

El método *Resampling* para DE, propuesto por Arabas et al. [2] y explicado previamente en la Sección 3.2.8, consiste en repetir la mutación diferencial remuestreando la población hasta que se genere un vector válido completo. Si al menos una variable infringe los límites, el operador se aplica de nuevo para generar una solución completamente nueva.
Este método ha demostrado ser una de las mejores formas de lidiar con las restricciones de límite en DE, su capacidad para llegar a la región factible se ha demostrado incluso en optimización restringida [2, 24].
Sin embargo, en un artículo reciente [26] se demostró que el método *Resampling* puede llegar a ciclar la ejecución del algoritmo DE cuando es muy dificil generar un vector válido después de probar todos los posibles remuestreos (las combinaciones no son infinitas), e incluso en los casos en los que sí se logra obtener un vector válido, el número de remuestreos puede ser tan alto que incremente significativamente el tiempo de ejecución del algoritmo.
Por lo anterior, en el presente trabajo se propone un método *Resampling* híbrido que limita el

1: Input: i //Índice del vector *target*.
2: Output: $\vec{V}$ // //Vector mutante.
3: $NoRes = 0$ //Contador de remuestreos.
4: $valid = false$
5: **repetir**
6: $r_i = randint[1,NP]\ , \forall i \in \{1,2,3\} : r_1 \neq r_2 \neq r_3 \neq i$
7: $\vec{V} = \vec{X}_{r_1} + F(\vec{X}_{r_2} - \vec{X}_{r_3})$
8: **si** $\forall j \in \{1,...,D\} : l_j \leq V_j \leq u_j$ **entonces**
9: valid=true
10: **fin si**
11: $NoRes = NoRes + 1$
12: **hasta que** $NoRes > (3 * D)\ \vee$ *valid=true*
13: **si** $valid = false$ **entonces**
14: $V_j = \{l_j + rand[0,1] \times (u_j - l_j)\ , \forall j \in \{1,...,D\} : v_j < l_j\ \vee\ v_j > u_j\}$
15: **fin si**

Algoritmo 5: Método *Res&Ran* para la estrategia *DE/rand/1/bin*.

número máximo de remuestreos a $3 \times D$, donde D es la dimensionalidad del problema. El número máximo de remuestreos se ajustó experimentalmente buscando un compromiso entre la calidad de

las soluciones reparadas y el tiempo de ejecución del método.
Si después de repetir el operador de mutación diferencial un máximo de $3 \times D$ veces no es posible obtener un vector mutante válido, entonces el vector mutante no válido se repara asignando valores aleatorios entre los límites inferiores l y superiores u de las variables que salen del espacio de búsqueda, como se realiza en el método *Random* (presentado en la Sección 3.2.3), de ahí su nombre de *Res&Ran*. El Algoritmo 5 describe el método *Res&Ran* para la estrategia $DE/rand/1/bin$.
Es importante aclarar que este método sólo se aplica al algoritmo de Evolución Diferencial, pues aún cuando existe una versión del método *Resampling* para PSO, esta versión no ha sido considerada en el presente trabajo debido a su bajo rendimiento [1].

Capítulo 5

Diseño Experimental

En el presente trabajo de investigación, la etapa experimental está dividida en dos secciones, la primera, denominada: *Experimentos preliminares* se presenta en el Capítulo 6. Estos experimentos fueron necesarios para ajustar el valor de K para el método *Centroid* y para seleccionar los mejores métodos híbridos (compuestos por un método para el manejo de posición y una estrategia para actualizar la velocidad) que serían usados en el algoritmo PSO.
En la segunda sección (Capítulo 7), la etapa de experimentación principal se compara el rendimiento del esquema adaptativo, propuesto en el presente trabajo, contra varios métodos para el manejo de restricciones de límite.

5.1. Equipo de cómputo

Todos los experimentos se llevaron a cabo en equipos de cómputo con la configuración presentada en la Tabla 5.1.

Tabla 5.1: Configuración del equipo de cómputo utilizado en cada experimento.

System	Windows 8.1 64 bits
CPU	Intel(R) Core(TM) i5-4210U (2.40 GHz)
RAM	8 GB

5.2. Problemas de prueba

Para poner a prueba los métodos para el manejo de restricciones de límite se utilizaron 60 problemas de optimización numérica (continua) restringida mono objetivo, los cuales han sido to-

mados de dos conjuntos de problemas.

El primer conjunto consta de 24 problemas recopilados del IEEE *Congress on Evolutionary Computation* CEC-2006. Este conjunto contiene funciones lineales, no lineales, cuadráticas, cúbicas y polinomiales. Todas las funciones contienen restricciones de igualdad y/o desigualdad, tanto lineales como no lineales y la dimensionalidad de las mismas varía desde 2 hasta 24. Se realizaron 25 corridas independientes por función, con un máximo de 500,000 evaluaciones, como se indica en el documento de referencia [32].

La Tabla 5.2 presenta los detalles de los 24 problemas de prueba del CEC 2006, en esta tabla n representa al número de variables de decisión, $\rho = |F|/|S|$ es la relación estimada entre la región factible y el espacio de búsqueda, *LI* es el número de restricciones lineales de desigualdad, *NI* el número de restricciones no lineales de desigualdad, *LE* es el número de restricciones lineales de igualdad, *NE* es el número de restricciones no lineales de igualdad y a es el número de restricciones activas en $\vec{x}$. Las funciones y sus restricciones tanto funcionales como de límite se detallan en el Apéndice A.

El segundo conjunto de funciones fue tomado del IEEE *Congress on Evolutionary Computation*

Tabla 5.2: Detalles de los 24 problemas de prueba tomados de CEC-2006, de la sesión especial de optimización numérica restringida. Esta tabla fue tomada de [32].

Prob.	***n***	**Tipo de función**	ρ	**LI**	**NI**	**LE**	**NE**	**a**
g01	13	cuadrática	0.0111%	9	0	0	0	6
g02	20	no lineal	99.9971%	0	2	0	0	1
g03	10	polinomial	0.0000%	0	0	0	1	1
g04	5	cuadrática	52.1230%	0	6	0	0	2
g05	4	cúbica	0.0000%	2	0	0	3	3
g06	2	cúbica	0.0066%	0	2	0	0	2
g07	10	cuadrática	0.0003%	3	5	0	0	6
g08	2	no lineal	0.8560%	0	2	0	0	0
g09	7	polinomial	0.5121%	0	4	0	0	2
g10	8	lineal	0.0010%	3	3	0	0	6
g11	2	cuadrática	0.0000%	0	0	0	1	1
g12	3	cuadrática	4.7713%	0	1	0	0	0
g13	5	no lineal	0.0000%	0	0	0	3	3
g14	10	no lineal	0.0000%	0	0	3	0	3
g15	3	cuadrática	0.0000%	0	0	1	1	2
g16	5	no lineal	0.0204%	4	34	0	0	4
g17	6	no lineal	0.0000%	0	0	0	4	4
g18	9	cuadrática	0.0000%	0	13	0	0	6
g19	15	no lineal	33.4761%	0	5	0	0	0
g20	24	lineal	0.0000%	0	6	2	12	16
g21	7	lineal	0.0000%	0	1	0	5	6
g22	22	lineal	0.0000%	0	1	8	11	19
g23	9	lineal	0.0000%	0	2	3	1	6
g24	2	lineal	79.6556%	0	2	0	0	2

CEC-2010 y consta de 18 funciones, escalables, de optimización numérica continua restringida. Las 18 funciones fueron puestas a prueba en 10 y 30 dimensiones, haciendo un total de 32 problemas, por cada uno de los cuales se realizaron 25 corridas independientes con un máximo de 200,000 y 600,000 evaluaciones para los problemas de 10 y 30 dimensiones respectivamente.
Once de las 18 funciones contienen objetivos no separables, en seis casos el objetivo es separable y una función tiene un objetivo rotado. Todas las funciones cuentan con restricciones de igualdad y/o desigualdad, ya sean separables, no separables o rotadas; las restricciones rotadas han sido incorporadas con la finalidad de evitar que su solución esté sesgada hacia un tipo de algoritmos en particular [34].
La Tabla 5.3 presenta los detalles de los 18 problemas de prueba del CEC 2010. En esta tabla D representa al número de variables de decisión, ρ es la relación estimada entre la región factible y el espacio de búsqueda, I es el número de restricciones de desigualdad y E es el número de restricciones de igualdad. Las funciones y sus restricciones tanto funcionales como de límite se detallan en el Apéndice B.

5.3. Tiempo computacional

Para medir el tiempo de ejecución consumido por cada uno de los BCHM comparados en el presente trabajo, se utilizó la técnica propuesta por Wu, Mallipeddi y Suganthan [60], en donde se calcula el promedio de realizar 10,000 evaluaciones (*T1*) como:

$$T1 = (\sum_{i=1}^{60} t1_i)/60 \tag{5.1}$$

En donde $t1_i$ representa el tiempo computacional de 10,000 evaluaciones (sólo evaluaciones, sin considerar el algoritmo) para el problema i. En este trabajo el número total de problemas es 60, dado que se emplearon 24 problemas del índice de referencia CEC-2006 y 36 del índice de referencia CEC-2010. El valor de *T1* es el mismo para todos los BCHM comparados, ya que se trata del mismo algoritmo, ya sea PSO o DE, al cual únicamente se le cambia el método para el manejo de restricciones de límite. Por esta razón éste dato no es reportado en las tablas comparativas, pero si es necesario para calcular el parámetro $(T2 - T1)/T1$).
El tiempo computacional considerando el algoritmo ($T2$) también se calcula como:

$$T2 = (\sum_{i=1}^{60} t2_i)/60 \tag{5.2}$$

Tabla 5.3: Detalles de los 18 problemas de prueba tomados de CEC-2010, de la sesión especial de optimización numérica restringida. Esta tabla fue tomada de [34].

Problema/Rango de Búsqueda	Tipo de Objetivo	Número de Restricciones		Región de factibilidad(ρ)	
		E	I	10D	30D
$C01[0,10]^D$	No Separable	0	2 No Separable	0.997689	1.000000
$C02[-5,12,5.12]^D$	Separable	1 Separable	2 Separable	0.000000	0.000000
$C03[-1000,1000]^D$	No Separable	1 No Separable	0	0.000000	0.000000
$C04[-50,50]^D$	Separable	4 2 No Separable, 2 Separable	0	0.000000	0.000000
$C05[-600,600]^D$	Separable	2 Separable	0	0.000000	0.000000
$C06[-600,600]^D$	Separable	2 Rotado	0	0.000000	0.000000
$C07[-140,140]^D$	No Separable	0	1 Separable	0.505123	0.503725
$C08[-140,140]^D$	No Separable	0	1 Rotado	0.379512	0.375278
$C09[-500,500]^D$	No Separable	1 Separable	0	0.000000	0.000000
$C10[-500,500]^D$	No Separable	1 Rotado	0	0.000000	0.000000
$C11[-100,100]^D$	Rotado	1 No Separable	0	0.000000	0.000000
$C12[-1000,1000]^D$	Separable	1 No Separable	1 Separable	0.000000	0.000000
$C13[-500,500]^D$	Separable	0	3 2 Separable, 1 No Separable	0.000000	0.000000
$C14[-1000,1000]^D$	No Separable	0	3 Separable	0.003112	0.006123
$C15[-1000,1000]^D$	No Separable	0	3 Rotado	0.003210	0.006023
$C16[-10,10]^D$	No Separable	2 Separable	2 1 Separable, 1 No Separable	0.000000	0.000000
$C17[-10,10]^D$	No Separable	1 Separable	2 No Separable	0.000000	0.000000
$C18[-50,50]^D$	No Separable	1 Separable	1 Separable	0.000010	0.000000

Donde $t2_i$ es el tiempo computacional completo para el algoritmo con 10,000 evaluaciones para el problema i.

Finalmente, la complejidad del algoritmo se refleja en: $T1$, $T2$ y $(T2-T1)/T1$, estos valores se expresan en segundos.

En los Experimentos 1 y 3 del Capítulo 7 se presenta un comparativo del tiempo computacional de cada uno de los métodos utilizados dentro de los algoritmos clásicos de DE y PSO respectivamente. En los experimentos 2 y 4 no se presenta el comparativo ya que la relación en la comparación es la misma que la presentada en los Experimentos 1 y 3, pues en los Experimentos 2 y 4 únicamente se modificó la versión de los algoritmos PSO y DE, los BCHMs comparados permanecen sin cambios.

5.4. Presentación de los resultados

Los experimentos de los Capítulos 6 y 7 incluyen tablas que presentan el promedio de los valores de aptitud obtenidos en 25 ejecuciones independientes por cada uno de los problemas puestos a prueba con cada uno de los métodos (BCHM) comparados. Los mejores resultados se destacan en negritas. *(r)* significa que sólo en *r* de las 25 ejecuciones independientes se encontraron soluciones factibles.

Para validar el rendimiento de cada uno de los métodos (*BCHM*) puestos a prueba, a los que denominaremos *métodos objetivo*, sus resultados fueron comparados contra los obtenidos por un *método base* (en la mayoría de los casos, el *ABCHS* se consideró como el método base), aplicando una prueba estadística de Kruskal-Wallis con un nivel de confianza del 95 %. "†"significa que hubo una diferencia estadísticamente significativa a favor del *método base*. "∗"significa que hubo una diferencia estadísticamente significativa a favor del *método objetivo*.

Los resultados obtenidos en las pruebas estadísticas Kruskal-Wallis se resumen en la última fila de cada tabla como $w/t/l$, lo que significa que el *método objetivo* obtuvo mejores resultados en w funciones (en estas funciones la diferencia estadística fue a favor del método objetivo), en t funciones no se encontraron diferencias estadísticamente significativas, y perdió en l funciones (en estas funciones la diferencia estadística fue a favor del método base), el comparar sus resultados contra los obtenidos por el *método base*.

Todos los experimentos están acompañados por gráficos que permiten comparar el rendimiento de cada *BCHM* en forma resumida, estos gráficos incluyen dos aspectos: la capacidad de alcanzar la región factible y el rendimiento general comparado:

1. *Capacidad para llegar a la región factible:* En este caso, se comparó el número de ejecuciones independientes en las cuales se encontraron soluciones factibles (*NoFR*) por cada uno de los métodos comparados.
   El número total de ejecuciones independientes para cada método fue 1500, ya que se realizaron 25 ejecuciones independientes para cada una de las 60 funciones probadas.
   La variable *NoFR* (Number of Feasible Runs) puede tomar valores de 0 a 1500. Entre más alto sea el valor de *NoFR* indica que es mejor la capacidad del método en cuestión para llegar a la región factible.

2. *Rendimiento general comparado:* Con la finalidad de hacer un comparativo que involucre tanto la cantidad como la calidad de las soluciones factibles encontradas por cada método, se calculó un valor del rendimiento general comparado (*Overall Comparative Performance,*

OCP) de la siguiente manera:

$$OCP = \frac{\sum_{j=1}^{nf} \sum_{i=1}^{nr} pr_{i,j}}{nf} \tag{5.3}$$

En donde *nf* representa el número de funciones probadas, en este caso *nf = 60*, *nr* representa el número de ejecuciones independientes por función, en este caso *nr = 25* y $pr_{i,j}$ representa un valor de rendimiento asignado a la ejecución independiente *i* de la función *j*. $pr_{i,j}$ toma el valor de *0* cuando el BCHM no encuentra una solución factible, por el contrario, el valor de $pr_{i,j}$ se asigna de acuerdo con lo siguiente:

$$pr_{i,j} = \begin{cases} 0.50 & \text{Si se encontró una Diferencia Estadísticamente} \\ & \text{Significativa (}SSD\text{) en la prueba Kruskal-Wallis} \\ & \text{que favoreciera al } \textit{método base}. \\ 0.75 & \text{Si no se encuentra una } SSD. \\ 1.00 & \text{Si se encuentra una } SSD \\ & \text{que favorece al } \textit{método objetivo}. \end{cases} \tag{5.4}$$

La variable *OCP* toma valores de 0 a *nr*. Entre mayor sea el valor de *OCP*, mejor será el rendimiento general comparado del método objetivo.

Capítulo 6

Experimentos Preliminares

6.1. Ajuste del parámetro *K* en el método *Centroid*

El método *Centroid*, presentado en la Sección 4.2, reubica los vectores reparados en el centroide de un área formada por $K+1$ vectores.
Los K vectores consisten en individuos corregidos mediante el método *Random*, los cuales atraen al vector reparado hacia espacios divergentes dentro del espacio de búsqueda, fomentando así la exploración. El vector restante es tomado de la población actual, siendo éste un individuo factible o con la menor violación de restricciones, mismo que atrae al vector reparado hacia la región factible.
Siendo K un parámetro no determinado por el método, se pueden crear tantas variantes de Centroid como se estime necesario, tales como (*1+1*, *2+1*, *3+1*, ... *K+1*). Por esta razón, en el presente experimento se compara el rendimiento de nueve de estas variantes en donde $K=\{1, 2, \ldots, 9\}$, a fin de determinar el mejor valor para K.
En el presente experimento se utiliza únicamente el algoritmo clásico de DE con la estrategia *DE/rand/1/bin* empleando las reglas de *Deb* para el manejo de las restricciones funcionales [10]. Los parámetros del algoritmo DE fueron: *NP=100* y *F=0.7*, considerando las recomendaciones de Mezura-Montes et al. [39]; para evitar el impacto del factor de cruza [26] se utilizó *CR=1.0*.
En este experimento se utilizó unicamente el conjunto de funciones de prueba del *CEC-2010*, realizando un máximo de 200,000 y 600,000 evaluaciones para los problemas de 10 y 30 dimensiones respectivamente en cada ejecución independiente.
Las Tablas 6.1 y 6.2 muestran el valor de aptitud promedio de 25 ejecuciones independientes para los problemas en 10 dimensiones mientras que las Tablas 6.3 y 6.4 presentan estos mismos valores para los problemas en 30 dimensiones. Las Tablas 6.1 y 6.3 muestran los resultados obtenidos en las funciones en las cuales al menos una de las variantes de *Centroid* obtuvo soluciones factibles

en las 25 ejecuciones independientes, lo cual indica que en estas funciones fue mas fácil llegar a la región factible. En las Tablas 6.2 y 6.4 se presentan los resultados obtenidos en las funciones con una mayor complejidad para alcanzar la región factible.

En las cuatro tablas se utilizó la variante *Centroid 2+1* como método base para las pruebas estadísticas de *Kruskal-Wallis* (intervalo de confianza de 95 %).

En este experimento se analizaron cuatro aspectos: 1) el valor de aptitud obtenido por cada una de las variantes de *Centroid*, 2) la capacidad de cada variante para obtener soluciones factibles, 3) el porcentaje global de vectores reparados por cada variante y 4) el porcentaje de vectores reparados que fueron mejores que su correspondiente vector objetivo (*target*).

En la Tabla 6.1 se observa que la variante $K = 1$ obtuvo el mejor rendimiento en los problemas de 10 dimensiones en los cuales fue más fácil llegar a la región factible, ya que fue la única variante que obtuvo un rendimiento estadísticamente superior a $K = 2$ en tres funciones. En el resto de las funciones no se observaron diferencias estadísticamente significativas. Adicionalmente, también se observa que la variante $K = 1$ obtuvo el mejor valor de aptitud en seis problemas.

Por otro lado, la variante $K = 3$ obtuvo un rendimiento similar a $K = 2$, las variantes con un valor de $K > 3$ obtuvieron un peor rendimiento en este subconjunto de problemas.

En la Tabla 6.2 se presentan los resultados para el resto de los problemas de 10 dimensiones, en los

Tabla 6.1: Resultados obtenidos por el método *Centroid* en cada una de las variantes $K=\{1, 2, \ldots, 9\}$ con *DE* clásico en los problemas del *CEC2010 10D* en donde todas las variantes encontraron soluciones factibles en las 25 ejecuciones independientes. El símbolo † significa que existe diferencia estadísticamente significativa (*SSD*) a favor del método base (Centroid 2+1) y $*$ representa *SSD* en contra del método base.

Problema	**Centroid 2+1**	**Centroid 1+1**	**Centroid 3+1**	**Centroid 4+1**	**Centroid 5+1**	**Centroid 6+1**	**Centroid 7+1**	**Centroid 8+1**	**Centroid 9+1**
C01	-7.46E-01	-7.46E-01	**-7.47E-01**	**-7.47E-01**	**-7.47E-01**	**-7.47E-01**	**-7.47E-01**	**-7.47E-01**	**-7.47E-01**
C03	3.78E-24	**2.30E-24**	5.20E-24	6.73E-24*	8.62E-24*	7.70E-24*	5.50E-24*	6.72E-24*	5.41E-24*
C04	**-1.00E-05**	**-1.00E-05**	**-1.00E-05**	**-1.00E-05**	**-1.00E-05***	**-1.00E-05**	**-1.00E-05***	**-1.00E-05**	**-1.00E-05***
C07	7.79E-23	**8.73E-24†**	3.02E-23	3.99E-23	5.71E-23	6.75E-23*	8.29E-23*	1.30E-22*	6.42E-23
C08	7.52E+00	7.92E+00	9.73E+00	8.47E+00	8.90E+00*	9.10E+00*	9.33E+00*	8.38E+00*	**7.36E+00**
C11	**-1.52E-03**	**-1.52E-03†**	**-1.52E-03**	**-1.52E-03**	**-1.52E-03**	**-1.52E-03***	**-1.52E-03***	**-1.52E-03***	**-1.52E-03***
C12	**-1.32E+01**	-6.69E+00	-6.56E+00	-5.63E+00	-7.69E+00	-2.56E+00	-1.18E+01	-5.69E+00	-1.12E+00*
C13	-6.10E+01	-6.02E+01	**-6.20E+01**	-6.03E+01	-6.09E+01	-6.15E+01	-6.17E+01	-6.07E+01	-6.16E+01
C14	2.70E+11	**1.34E+11†**	4.02E+11	4.09E+11	3.64E+11	6.22E+11*	4.46E+11	5.36E+11	5.32E+11*
C15	4.91E+12	**4.77E+12**	7.30E+12	7.84E+12*	8.38E+12*	8.95E+12*	1.20E+13*	6.97E+12	1.02E+13*
w/t/l	**-**	**3/7/0**	**0/10/0**	**0/8/2**	**0/6/4**	**0/4/6**	**0/4/6**	**0/6/4**	**0/4/6**

cuales fue más difícil alcanzar la región factible. En esta tabla se observa que las pruebas estadísticas reportaron un rendimiento similar para la mayoría de las variantes, excepto para la variante K=3, misma que mostró un rendimiento estadísticamente inferior a la variante base (K=2) en solo una función.

Con referencia al número de corridas factibles, las variantes K=$\{$1,2,6,8$\}$ obtuvieron la mayor

Tabla 6.2: Resultados obtenidos por el método *Centroid* en cada una de las variantes *K*={*1, 2, . . . , 9*} con *DE* clásico en los problemas del *CEC2010 10D* en donde ninguna de las variantes encontró soluciones factibles para todas las 25 ejecuciones independientes. El símbolo ∗ significa que existe *SSD* en contra del método base (Centroid 2+1).

Problema	Centroid 2+1	Centroid 1+1	Centroid 3+1	Centroid 4+1	Centroid 5+1	Centroid 6+1	Centroid 7+1	Centroid 8+1	Centroid 9+1
C02	3.10E+00(12)	2.76E+00(**16**)	3.50E+00(13)	3.26E+00(11)	2.80E+00(12)	**2.60E+00(16)**	3.45E+00(9)	2.98E+00(15)	3.15E+00(12)
C05	3.23E+02(2)	**2.37E+02**(1)	3.50E+02(3)	3.95E+02(3)	2.80E+02(3)	2.62E+02(3)	2.44E+02(5)	3.29E+02(**7**)	2.70E+02(4)
C06	4.45E+02(2)	**1.83E+02**(2)	3.03E+02(2)	3.33E+02(**4**)	3.90E+02(3)	4.60E+02(2)	4.03E+02(2)	3.64E+02(2)	4.14E+02(3)
C09	4.36E+12(**9**)	2.84E+12(5)	2.96E+12(4)	5.56E+12(6)	4.01E+12(7)	6.33E+12(8)	**1.64E+12**(4)	2.85E+12(8)	5.08E+12(**9**)
C10	2.16E+12(3)	4.91E+12(5)	**1.96E+12**(6)	4.17E+12(5)	4.90E+12(4)	4.79E+12(**7**)	2.90E+12(3)	3.95E+12(4)	4.47E+12(3)
C16	**1.02E+00(15)**	**1.02E+00**(12)	1.04E+00(7)	1.03E+00(4)	**1.02E+00**(11)	1.04E+00(10)	1.03E+00(14)	1.03E+00(12)	1.04E+00(7)
C17	**1.95E+02**(13)	3.87E+02(10)	3.11E+02(**18**)*	2.40E+02(10)	2.86E+02(12)	2.39E+02(14)	3.34E+02(13)	2.23E+02(**18**)	2.82E+02(13)
C18	6.26E+03(20)	**5.73E+03(24)**	7.89E+03(18)	5.97E+03(19)	7.32E+03(19)	5.83E+03(20)	7.51E+03(21)	6.14E+03(20)	7.13E+03(19)
w/t/l	-	**0/8/0**	**0/7/1**	**0/8/0**	**0/8/0**	**0/8/0**	**0/8/0**	**0/8/0**	**0/8/0**

cantidad de corridas factibles en dos funciones, mientras que las variantes K={3,4,9} obtuvieron la mayor cantidad de corridas factibles en una función.

En cuanto al valor de aptitud, la variante K=1 obtuvo el mejor resultado en cuatro funciones, seguida por la variante K=2 que obtuvo el mejor valor de aptitud en dos funciones y finalmente, las variantes K={3,5,6,7} obtuvieron el mejor valor de aptitud en una función.

Considerando la cantidad de ejecuciones factibles como el mejor valor de aptitud, en estos problemas se observó que la variante K=1 obtuvo el mejor rendimiento, seguido por las variantes K=2 y K=6.

Como resumen, los resultados mostrados en las Tablas 6.1 y 6.2 indican que la variante K=1 obtuvo el mejor rendimiento en los problemas de 10 dimensiones, seguida por la variante K=2.

En la Tabla 6.3 se muestran los valores de aptitud obtenidos en los problemas de 30 dimensio-

Tabla 6.3: Resultados obtenidos por el método *Centroid* en cada una de las variantes *K*={*1, 2, . . . , 9*} con *DE* clásico en los problemas del *CEC2010 30D* en donde al menos una de las variantes encontró soluciones factibles en las 25 ejecuciones independientes. El símbolo † significa que existe *SSD* a favor del método base (Centroid 2+1) y ∗ representa *SSD* en contra del método base.

Problema	Centroid 2+1	Centroid 1+1	Centroid 3+1	Centroid 4+1	Centroid 5+1	Centroid 6+1	Centroid 7+1	Centroid 8+1	Centroid 9+1
C01	-4.85E-01	-4.36E-01*	**-5.21E-01**†	-5.15E-01†	-5.00E-01	-5.14E-01	-5.11E-01	-4.88E-01	-4.77E-01
C07	5.02E+06	3.17E+07*	2.42E+06	6.85E+05†	1.17E+06†	1.28E+06†	1.78E+06†	1.26E+06†	**6.19E+05**†
C08	5.09E+06	1.93E+07*	2.68E+06	3.35E+06†	1.32E+06†	1.12E+06†	1.59E+06†	**1.08E+06+**	1.25E+06†
C13	-4.99E+01	-5.25E+01	-5.17E+01	-5.13E+01	-4.93E+01	-5.06E+01	-5.20E+01	**-5.29E+01**	-5.00E+01
C14	2.30E+12	**2.09E+12**	4.41E+12*	5.33E+12*	6.52E+12*	7.28E+12*	9.43E+12*	1.11E+13*	1.06E+13*
C15	2.19E+13	**1.81E+13**	2.94E+13*	3.82E+13*	4.72E+13*	4.88E+13*	4.82E+13*	5.23E+13*	5.14E+13*
C18	**1.33E+04**(24)	1.58E+04(22)	1.37E+04(24)	1.68E+04(24)*	1.88E+04(24)*	1.68E+04*	1.79E+04*	1.69E+04*	1.65E+04(24)*
w/t/l	-	**0/4/3**	**1/4/2**	**3/1/3**	**2/2/3**	**2/2/3**	**2/2/3**	**2/2/3**	**2/2/3**

nes en los cuales fue más fácil llegar a la región factible. En este grupo de funciones las pruebas estadísticas reportan que las variantes K=2 y K=4 obtuvieron un rendimiento similar ya que la variante K=4 obtuvo un mejor rendimiento que K=2 en tres funciones, lo empató en una y perdió en tres funciones. Posteriormente, la variante K=3 obtuvo un mejor rendimiento que la variante base

en una función, lo empata en cuatro y pierde en tres.
Las variantes K={5,6,7,8,9} obtuvieron un desempeño similar entre ellas, pues todas obtuvieron un mejor rendimiento que el método base en dos funciones, empataron en dos y perdieron en tres. Finalmente, la variante K=1, aunque obtuvo el mejor valor de aptitud en dos funciones, según las pruebas estadísticas, perdió contra el método base en tres funciones y lo empató en cuatro.

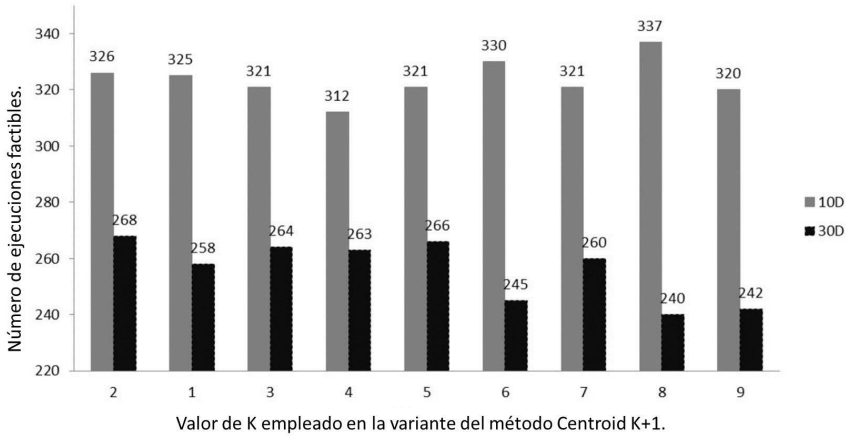
La Tabla 6.4 muestra los resultados obtenidos en los problemas de 30 dimensiones, en los cua-

Tabla 6.4: Resultados obtenidos por el método *Centroid* en cada una de las variantes *K={1, 2, . . . , 9}* con *DE* clásico en los problemas del *CEC2010 30D* en donde ninguna de las variantes encontró soluciones factibles en todas las 25 ejecuciones independientes. El símbolo † significa que existe *SSD* a favor del método base (Centroid 2+1) y ∗ representa *SSD* en contra del método base.

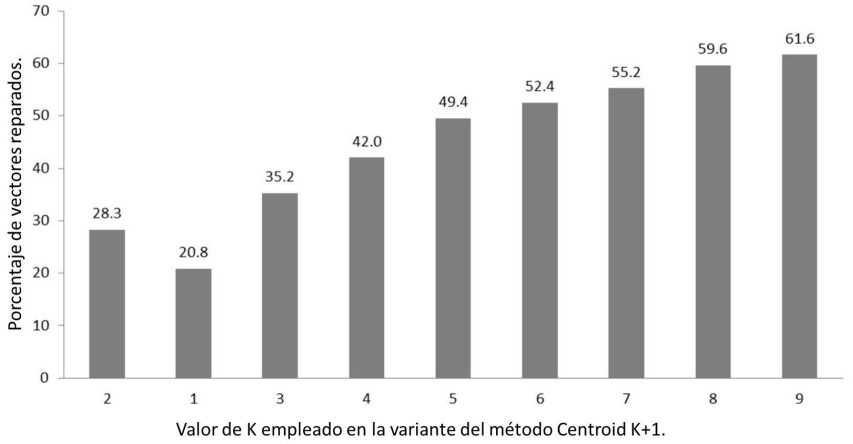
Problema	Centroid 2+1	Centroid 1+1	Centroid 3+1	Centroid 4+1	Centroid 5+1	Centroid 6+1	Centroid 7+1	Centroid 8+1	Centroid 9+1
C02	4.20E+00(**20**)	4.02E+00(13)	4.14E+00(17)	**3.85E+00**(15)	3.93E+00(11)	4.03E+00(13)	4.06E+00(17)	4.34E+00(5)	4.41E+00(12)
C05	3.62E+02(11)	**3.16E+02**(9)	4.56E+02(8)	3.71E+02(9)	4.00E+02(**16**)	4.32E+02(7)	4.54E+02(5)	4.25E+02(9)	4.14E+02(7)
C06	4.51E+02(7)	3.78E+02(7)	4.10E+02(6)	4.66E+02(**9**)	4.84E+02(5)	**3.74E+02**(6)	4.02E+02(5)	4.54E+02(6)	4.26E+02(3)
C09	1.07E+13(13)	**8.00E+12**(12)	9.20E+12(**15**)	9.91E+12(14)	1.27E+13(9)	8.83E+12(9)	1.26E+13(8)	1.43E+13(8)	1.94E+13(7)*
C10	9.23E+12(12)	**8.77E+12(14)**	1.02E+13(10)	1.46E+13(7)*	1.05E+13(**14**)	1.50E+13(4)*	1.76E+13(11)*	1.57E+13(7)	1.83E+13(7)
C16	1.07E+00(15)	**1.05E+00**(15)†	1.07E+00(14)	1.09E+00(15)	1.08E+00(16)	1.09E+00(12)*	1.09E+00(**22**)*	1.09E+00(13)	1.10E+00(12)*
C17	8.08E+02(18)	**7.30E+02**(14)	7.74E+02(20)	8.20E+02(20)	9.08E+02(**21**)	8.14E+02(19)	9.37E+02(17)	9.44E+02(17)	1.07E+03(20)*
w/t/l	-	**1/6/0**	**0/7/0**	**0/6/1**	**0/7/0**	**0/5/2**	**0/5/2**	**0/7/0**	**0/4/3**

les fue más difícil llegar a la región factible. En este grupo de problemas, la variante K=5 obtuvo la mayor cantidad de corridas factibles para tres de las funciones, seguida por las variantes K={1,2,3,4,7}, las cuales obtuvieron la mayor cantidad de corridas factibles en una función.
Con referencia a las pruebas estadísticas, la variante K=1 obtuvo el mejor rendimiento, seguido por las variantes K={2,3,5,8}, para el resto de las variantes su rendimiento fue peor. Con referencia a el valor de aptitud la variante K=1 obtuvo los mejores resultados en cinco funciones, seguida por las variantes K={4,6}, las cuales obtuvieron los mejores resultados en una función.
Por lo anterior, se concluye que en los problemas de 30 dimensiones, en los que fue más fácil llegar a la región factible, la variante K=1 obtuvo el mejor rendimiento, seguido por la variante K=5.
De forma conjunta, para los problemas de 30 dimensiones, las variantes que obtuvieron el mejor rendimiento fueron K=1 y K=2.

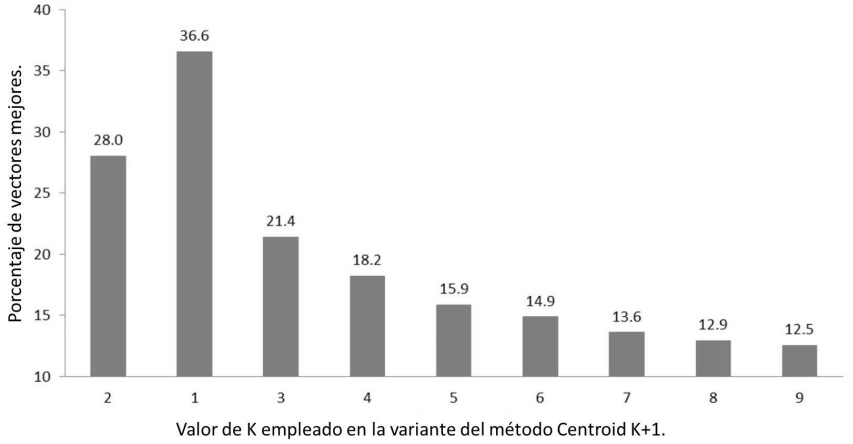
En la Figura 6.2(a) se presenta el número de ejecuciones independientes (tanto para los problemas de 10 como en 30 dimensiones) en las que cada una de las variantes comparadas obtuvo resultados factibles. En este gráfico se observa que la variante K=8 obtuvo la mayor cantidad de ejecuciones factibles en 10 dimensiones y la variante K=2 en 30 dimensiones.
En la Figura 6.2(b) se muestra el porcentaje de individuos reparados por cada una de las variantes. Aquí se observa que la variante con la menor cantidad de reparaciones es K=1, seguido por K=2, y de forma general, al aumentar el valor de K, aumenta la cantidad de individuos reparados. Este parámetro es importante ya que indica la bondad del método para mantener la población dentro del espacio de búsqueda.



(a) Número de ejecuciones independientes en las cuales, cada una de las variantes comparadas, encontró resultados factibles en los problemas de 10 y 30 dimensiones.



(b) Porcentaje general de vectores reparados por cada una de las variantes comparadas.



(c) Porcentaje general de vectores reparados que fueron mejores que su correspondiente vector objetivo (target), por cada una de las variantes comparadas.

Figura 6.1: Gráficas de resultados.

En la Figura 6.2(c) se presenta el porcentaje de vectores que, después de ser reparados por alguna de las variantes, fueron mejores que su vector objetivo (target). En este gráfico se observa que la variante K=1 obtuvo el mayor porcentaje, seguido por K=2, después K=3 y así sucesivamente. De forma general se observa que al aumentar el valor de K disminuye el porcentaje de vectores reparados que fueron mejores que su vector objetivo.
A manera de conclusión del presente experimento preliminar, en el cual se busca el mejor valor para al parámetro K en el método *Centroid*, se ha observado que según las pruebas estadísticas la variante K=1 mostró un mejor rendimiento en los problemas de 10 dimensiones, mientras que la variante K=2 hizo lo propio en los problemas de 30 dimensiones. Considerando los problemas de 10 y 30 dimensiones de forma conjunta, las cinco variantes con el mejor rendimiento fueron: K=$\{$1,2,3,4 y 5$\}$, en ese mismo orden.
La variante K=8 obtuvo soluciones factibles en la mayor cantidad de ejecuciones independientes para los problemas de 10 dimensiones, mientras que la variante k=2 hizo lo propio para los problemas de 30 dimensiones. Considerando tanto los problemas de 10 como de 30 dimensiones, las mejores variantes fueron: K=$\{$2,5,3,1 y 7$\}$, en ese mismo orden.
Se observó también que al aumentar el valor de K disminuye el porcentaje de vectores que después de ser reparados obtienen un mejor valor de aptitud que el de su correspondiente vector objetivo, en otras palabras, al aumentar el valor de K disminuye la calidad de los vectores reparados.
Adicionalmente, al aumentar el valor de K se realizan más reparaciones, esto que implica consumir más tiempo de procesamiento, a la vez que se degrada la calidad de las soluciones.
Considerando los puntos anteriores se concluye que la variante K=1 es una de las opciones con un mejor rendimiento y un menor consumo de recursos, por lo tanto, en los subsecuentes experimentos se utilizará solamente esta variante, a la cual se le referirá únicamente como *Centroid*.

6.2. Ajuste de los métodos híbridos para el manejo de límites dentro de PSO

Como se mencionó anteriormente, en estudios previos [1, 6, 21, 40] se ha demostrado la importancia de modificar tanto la posición como la velocidad de las partículas que salen del espacio de búsqueda en el algoritmo de Optimización por Cúmulos de Partículas (*PSO*).
Atendiendo a esta premisa, en el presente estudio preliminar se evalúa el rendimiento de treinta métodos híbridos formados a partir de la combinación de seis métodos que permiten manipular la posición de las partículas que salen del espacio de búsqueda *(Position Handling Methods)*:

PHM = $\{$*Boundary (Bou), Centroid (Cen), Wrapping (Wra), Evolutionary (Evo),*

Reflection (Ref), and Random (Ran)$\}$

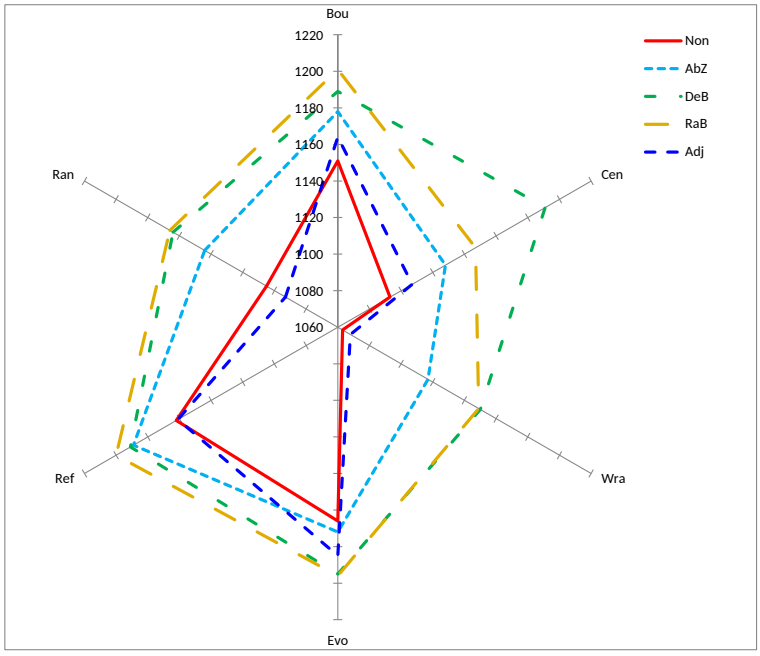
Los cuales han sido combinados con cinco estrategias que se emplean comúnmente en la actualización de la velocidad de las partículas *(Velocity Update Strategies):*

$$VUS = \{\textit{None (Non), Absorb Zero (AbZ), Deterministic Back (DeB),}$$
$$\textit{Random Back (RaB) and Adjust (Adj)}\}$$

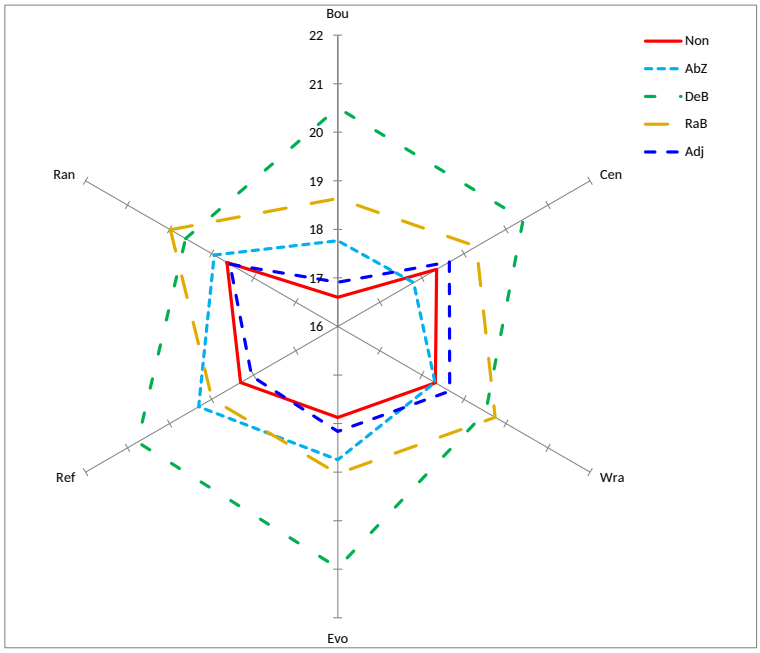
En el caso de *Deterministic Back*, se empleó un valor de $\lambda = 0.5$, tal como se recomienda en trabajos previos [6, 21, 25].
De tal forma que se han generado treinta métodos híbridos para el manejo de restricciones de límite (*Boundary Constraint-Handling Methods*):

$$BCHM = \{Bou\&Non,\ Bou\&Aze,\ \ldots Cen\&Non,\ \ldots, Ran\&Adj\}$$

En el presente experimento se comparó el rendimiento de estos treinta métodos hibridos dentro del algorítmo PSO al resolver problemas de optimización numérica restringida, con la finalidad de seleccionar sólo las mejores combinaciones, mismas que serán empleadas en los subsecuentes experimentos.
La Figura 6.2(a) muestra un gráfico radial con el número de ejecuciones factibles *(Number of Feasible Runs) NoFR* por cada método híbrido, este parámetro muestra la capacidad del método para llegar a la región factible.
La Figura 6.2(b) muestra una gráfica de radar con el valor del rendimiento general comparado (*Overall Comparative Performance, OCP*), obtenido por cada método hibrido *BCHM*. Este valor representa la calidad de las soluciones factibles encontradas por cada método híbrido *BCHM* (*target method*) en comparación con la calidad de las soluciones factibles encontradas al emplear un método base *(base method)*.
En este experimento se empleó como método base *Bou&Non*, el cual modifica la posición de las partículas inválidas mediante *Boundary* y su velocidad mediante la estrategia *None*. El método *Boundary* coloca las partículas invalidas en la frontera del espacio de búsqueda, mientras que la estrategia *None* permite que las partículas conserven su misma velocidad.
En la Figura 6.2(a) se aprecia que el método *Boundary* obtuvo el mayor número de corridas factibles al combinarse con la estrategia *RaB* seguido por la estrategia *DeB*, con un diferencia de sólo 12 corridas factibles. La Figura 6.2(b) muestra también que *Boundary* obtuvo el mayor rendimiento general comparado (OCP) al combinarse con la estrategia *DeB*, seguido por *RaB*.
Debido a que la diferencia en el número de corridas factibles entre las estrategias *RaB* y *DeB* fue muy pequeña, en comparación con la diferencia en la calidad de las soluciones encontradas por las estrategias mencionadas, se concluye que la estrategia *DeB* es la que permite obtener las mejores



(a) Gráfica radial mostrando el número de ejecuciones factibles (*NoFR*) por cada método híbrido *BCHM*, es decir, la capacidad de cada método para llegar a la región factible.



(b) Gráfica radial mostrando el valor del rendimiento general comparado (*OCP*), calculado como se indica en la Ecuación 5.3. Este parámetro representa la calidad de las soluciones factibles encontradas por cada método.

Figura 6.2: Resultados obtenidos al comparar el rendimiento de los métodos que permiten manipular la posición *(PHM)*, en combinación con las estrategias empleadas en la actualización de la velocidad *(VUS)*.

soluciones al combinare con el método *Boundary*, lo cual implica colocar las soluciones invalidas en la frontera del espacio de búsqueda y multiplicar su velocidad por $-\lambda$.
En el caso del método *Centroid*, la Figura 6.2 nos muestra que logró obtener tanto el mayor número de corridas factibles así como el mayor rendimiento general comparado al ser hibridado con la estrategia *DeB*, seguido de *RaB*.
El método *Wrapping*, por su parte, obtuvo la mayor cantidad de corridas factibles al ser combinado con la estrategia *DeB*, seguida por *RaB*, con una diferencia mínima de sólo una corrida factible entre ellas, tal como se aprecia en la Figura 6.2 (a). Con respecto al rendimiento general comparado, la Figura 6.2(b) nos muestra que el valor *OCP* más alto, del método *Wrapping*, se obtuvo al combinarlo con la estrategia *RaB*, seguido de *DeB*. De lo anterior se concluye que la estrategia que mostró el mejor rendimiento al hibridarse con el método *Wrapping* fue *RaB*, ya que permitió obtener soluciones de mejor calidad.
El método *Evolutionary* obtuvo la mayor cantidad de corridas factibles al hibridarse con la estrategia *RaB*, seguida por *DeB*, con una diferencia de sólo una corrida. En cuanto al rendimiento general comparado *OCP*, como se aprecia en la Figura 6.2(b), el valor más alto se obtuvo en combinación con *DeB*, seguido por *RaB*, en este caso, con una amplia diferencia. De lo anterior, se concluye que el método *Evolutionary* obtuvo su mejor rendimiento al combinarse con la estrategia *DeB*.
El método *Reflection* logró obtener la mayor cantidad de corridas factibles en combinación con *RaB*, seguido por *DeB*, con una diferencia de sólo nueve corridas de las 1500 posibles. Con respecto al rendimiento general comparado *OCP*, el valor más alto se obtuvo al combinarlo con *DeB*, seguido por *AbZ*. Sobre la base de estos resultados, se concluye que el método *Reflection* presenta un mejor desempeño al ser hibridado con la estrategia *DeB*, ya que ésta le permitió obtener soluciones de mejor calidad, además que representa un compromiso entre calidad y cantidad de soluciones factibles.
Finalmente, en la Figura 6.2, también podemos observar que el método *Random* obtuvo tanto la mayor cantidad de corridas factibles, así como el más alto rendimiento general comparado, al combinarse con la estrategia *RaB*.
A partir del presente experimento se desprenden las siguientes observaciones:
Las estrategias con el peor rendimiento para manipular la velocidad fueron *Non* y *Adj*. Emplear la estrategia *Non* es lo mismo que no emplear estrategia alguna, es decir, dejar la velocidad de la partícula sin cambio alguno. En cuanto a la estrategia *Adj*, ésta sí modifica la velocidad de la partícula, pero lo hace con base en su nueva posición y su posición anterior, en lugar de emplear su velocidad previa.
Las estrategias que reportaron un mejor rendimiento fueron *DeB* y *RaB*, en ambos casos se emplea la velocidad previa, invirtiéndola al multiplicarla por un factor $-\lambda$, $[0 <= \lambda <= 1]$.

Por tanto, se concluye que, aparte de modificar la posición de las partículas que salen del espacio de búsqueda en el algoritmo PSO, es también primordial emplear alguna estrategia para modificar su velocidad, y en este punto, los resultados indican que la mejor forma de hacerlo es invirtiendo la velocidad de las partículas.
La Tabla 6.5 muestra las mejores combinaciones (entre un método para modificar la velocidad de las partículas y una estrategia para actualizar su velocidad) encontradas en el presente experimento. Estos métodos híbridos serán empleados en subsecuentes experimentos con el algoritmo PSO.

Tabla 6.5: Mejores métodos híbridos para el manejo de restricciones de límite *(BCHM)*, encontrados en el presente experimento, empleando el algoritmo de optimización por cúmulos de partículas *(PSO)*, en su versión canónica, para optimización restringida.

Método híbrido (BCHM)	Método de manejo de posición (PHM)	Estrategia de actualización de velocidad (VUS)
Bou&DeB	Boundary	Deterministic Back
Cen&DeB	Centroid	Deterministic Back
Wra&RaB	Wrapping	Random Back
Evo&DeB	Evolutionary	Deterministic Back
Ref&DeB	Reflection	Deterministic Back
Ran&RaB	Random	Random Back

Capítulo 7

Experimentos empleando el esquema adaptativo (*ABCHS*)

7.1. Análisis del ABCHS dentro del algoritmo PSO

7.1.1. Experimento 1. Comparativo del rendimiento del ABCHS dentro del algoritmo PSO clásico

En este experimento se comparó el rendimiento del esquema adaptativo para el manejo de restricciones de límite (*ABCHS*), contra los seis mejores métodos híbridos para el manejo de restricciones de límite encontrados en los experimentos previos (Tabla 6.5) dentro de algoritmo PSO clásico.

Las Tablas 7.1 y 7.2 muestran los resultados obtenidos en los problemas del CEC2006, en este caso, las funciones g20, g21 y g22 (funciones con la mayor cantidad de restricciones de igualdad) fueron excluidas de este experimento, debido a que ninguno de los métodos comparados encontró soluciones factibles para estas funciones.

La Tabla 7.1 muestra los resultados de 12 funciones, en las cuales todos los métodos encontraron soluciones factibles en las 25 corridas independientes. En esta tabla se observa que los seis métodos híbridos comparados contra el *ABCHS* obtuvieron mejores soluciones que el *ABCHS* en sólo una función (g11, función cuadrática de dos dimensiones con sólo una restricción de igualdad), obtuvieron soluciones de peor calidad a las obtenidas por el *ABCHS* en cinco funciones, y obtuvieron resultados similares en las restantes seis funciones.

También se observó que el esquema adaptativo obtuvo los mejores resultados promedio en ocho funciones, seguido por *Cen&DeB* y *Wra&RaB* que obtuvieron los mejores valores promedio en cuatro funciones y el resto de los métodos comparados en tres funciones. El esquema adaptativo

propuesto mostró su mejor rendimiento en problemas con una mayor cantidad de restricciones y de mayor dimensionalidad.

La Tabla 7.2 muestra los resultados de las nueve funciones restantes, en donde al menos un méto-

Tabla 7.1: Resultados obtenidos por cada uno de los BCHMs con PSO clásico en los problemas del CEC2006 en donde todos los métodos encontraron soluciones factibles en las 25 corridas independientes. El símbolo † significa que existe diferencia estadísticamente significativa (*SSD*) a favor del método base (ABCHS) y ∗ representa *SSD* en contra del método base.

Problema	ABCHS	Cen&DeB	Evo&DeB	Ref&DeB	Ran&RaB	Wra&RaB	Bou&DeB
g02	**-6.46E-01**	-3.68E-01†	-4.80E-01†	-4.96E-01†	-3.60E-01†	-3.52E-01†	-4.51E-01†
g03	-6.44E-01	**-7.71E-01**	-6.52E-01	-6.29E-01	-5.86E-01	-7.02E-01	-6.18E-01
g04	**-3.07E+04**	-3.04E+04†	-3.05E+04†	-3.06E+04†	-3.06E+04†	-3.05E+04†	-3.04E+04†
g06	**-6.96E+03**	**-6.96E+03**	**-6.96E+03**	**-6.96E+03**	**-6.96E+03**	**-6.96E+03**	**-6.96E+03**
g08	**-9.58E-02**	**-9.58E-02**	**-9.58E-02**	**-9.58E-02**	**-9.58E-02**	**-9.58E-02**	**-9.58E-02**
g09	**6.81E+02**	6.93E+02†	6.94E+02†	6.92E+02†	6.92E+02†	6.89E+02†	6.92E+02†
g11	7.50E-01	**7.50E-01***	**7.50E-01***	7.50E-01*	7.50E-01*	**7.50E-01***	**7.50E-01***
g12	**-1.00E+00**	-9.99E-01	-9.99E-01	-9.99E-01	-1.00E+00	**-1.00E+00**	-9.99E-01
g13	1.10E+00	1.64E+00	1.14E+00	**8.94E-01**	1.19E+00	1.14E+00	1.53E+00
g16	**-1.90E+00**	-1.63E+00†	-1.64E+00†	-1.75E+00†	-1.80E+00†	-1.81E+00†	-1.77E+00†
g19	**5.51E+01**	4.20E+02†	9.40E+01†	1.11E+02†	3.98E+02†	3.96E+02†	1.21E+02†
g24	-5.51E+00	-5.51E+00	-5.51E+00	-5.51E+00	**-5.51E+00**	-5.51E+00	-5.51E+00
w/t/l	**-**	**1/6/5**	**1/6/5**	**1/6/5**	**1/6/5**	**1/6/5**	**1/6/5**

Tabla 7.2: Resultados obtenidos por cada uno de los BCHMs con PSO clásico en los problemas del CEC2006 en donde al menos uno de los métodos no encontró soluciones factibles en todas las 25 corridas independientes. El símbolo † significa que existe *SSD* a favor del método base (ABCHS).

Problema	ABCHS	Cen&DeB	Evo&DeB	Ref&DeB	Ran&RaB	Wra&RaB	Bou&DeB
g01	**-1.37E+01**	-1.79E+00†	-6.06E+00†	-8.84E+00†	-5.55E+00†	-5.84E+00(23)†	-4.54E+00†
g05	5.40E+03	5.29E+03(24)	5.42E+03(23)	**5.28E+03**(21)	5.30E+03	5.30E+03(24)	5.30E+03(23)
g07	**2.84E+01**	3.33E+02†	2.84E+02†	4.03E+02†	2.10E+02(23)†	2.01E+02(23)†	2.73E+02†
g10	**7.82E+03**	9.10E+03†	8.80E+03(19)†	1.00E+04(22)†	9.07E+03†	9.03E+03†	9.26E+03(20)†
g14	-4.33E+01	-	**-4.33E+01**	-4.33E+01	-	-	-4.29E+01(22)
g15	**9.63E+02**	9.63E+02	9.64E+02(24)†	9.64E+02†	9.64E+02	9.66E+02†	9.65E+02†
g17	8.94E+03	8.96E+03	9.01E+03(15)	8.98E+03(15)	9.04E+03(23)†	9.03E+03(17)†	**8.94E+03**(5)
g18	**-6.69E-01**	-5.63E-01†	-5.65E-01†	-5.15E-01†	-5.08E-01(15)†	-4.75E-01(12)†	-5.96E-01†
g23	1.14E+02	-	5.40E+01(10)	**1.49E+01**(3)	-	-	1.51E+01(15)
w/t/l	**-**	**0/3/4**	**0/4/5**	**0/4/5**	**0/2/5**	**0/1/6**	**0/4/5**

do no encontró soluciones factibles en al menos una de las 25 corridas independientes, es decir, funciones en las cuales fue más difícil llegar a la región factible.

En esta tabla se observa que el esquema adaptativo y los métodos híbridos *Evo&DeB*, *Ref&DeB* y *Bou&DeB* obtuvieron soluciones factibles en las nueve funciones, sin embargo, los métodos híbridos obtuvieron peores resultados que el esquema adaptativo en cinco de ellas.
Los métodos híbridos *Cen&DeB*, *Ran&RaB* y *Wra&RaB* obtuvieron soluciones factibles en sólo siete funciones y perdieron en las pruebas estadísticas en cuatro, cinco y seis funciones respectivamente, contra el *ABCHS*.
El esquema adaptativo también obtuvo los mejores valores promedio en el mayor número de funciones, en cinco de ellas, seguido por *Ref&DeB*, el cual obtuvo los mejores valores promedio en dos funciones y finalmente los métodos *Evo&DeB* y *Bou&DeB* obtuvieron los mejores valores en una función.
Por lo tanto, el esquema adaptativo propuesto (ABCHS) logró el mejor rendimiento (tanto la mayor cantidad de corridas factibles como las soluciones de mejor calidad) en los problemas del CEC2006, seguido por los métodos híbridos *Evo&DeB*, *Ref&DeB* y *Bou&DeB*.
El esquema adaptativo mostró su mejor desempeño en funciones con mayor número de restricciones y mayor dimensionalidad y en las funciones en las cuales fué más difícil llegar a la región factible.
Las Tablas 7.3 y 7.4 muestran los valores promedio de 25 corridas independientes para cada uno de los problemas del CEC-2010 en 10 dimensiones, las funciones C04 y C11 se excluyeron de este experimento, ya que ninguno de los métodos comparados fue capaz de obtener soluciones factibles. Ambas funciones cuentan únicamente con restricciones de igualdad (C04 con la mayoría de ellas) y en las cuales la proporción entre el tamaño de región factible y el del espacio de búsqueda es muy pequeña, C11 es la única función rotada.
La Tabla 7.3 muestra resultados de 13 funciones en las cuales todos los BCHM encontraron soluciones factibles en todas las 25 corridas independientes. Con base en las pruebas estadísticas, es posible observar que el esquema adaptativo obtuvo el mejor desempeño en este conjunto de problemas, seguido de *Wra&RaB* y *Cen&DeB*.
El método *Wra&RaB* obtuvo un mejor rendimiento que el ABCHS en una función (C02), aunque en ocho funciones su rendimiento fue peor, en las cuatro funciones restantes el rendimiento del método *Wra&RaB* fue similar al de ABCHS. Posteriormente, el método *Cen&DeB* obtuvo peores resultados que el ABCHS en ocho funciones y empató en las restantes cinco. El resto de los métodos comparados tuvieron un peor desempeño.
La Tabla 7.3 también muestra que ABCHS obtuvo los mejores valores promedio en 11 funciones, seguido de *Wra&RaB* que obtuvo los mejores valores promedio en dos funciones.
La Tabla 7.4 muestra los resultados de las tres funciones restantes, en las que al menos uno de los métodos no encontró soluciones factibles en todas las 25 corridas independientes. En esta tabla

Tabla 7.3: Resultados obtenidos por cada BCHM con PSO en los problemas del CEC-2010 en 10D. En esta tabla se presentan únicamente los resultados de las funciones en las cuales todos los métodos encontraron soluciones factibles en todas las 25 corridas independientes. El símbolo † significa que existe *SSD* a favor del método base (ABCHS) y ∗ representa *SSD* en contra del método base.

Problema	ABCHS	Cen&DeB	Evo&DeB	Ref&DeB	Ran&RaB	Wra&RaB	Bou&DeB
C01	**-6.35E-01**	-4.39E-01†	-5.47E-01†	-5.45E-01†	-4.48E-01†	-4.37E-01†	-5.15E-01†
C02	3.11E+00	3.41E+00	3.09E+00	3.32E+00	3.17E+00	**2.51E+00***	3.15E+00
C05	**2.26E+02**	2.75E+02	3.83E+02†	3.22E+02†	3.67E+02†	3.53E+02†	3.72E+02†
C07	**1.10E+02**	1.44E+03†	4.75E+03†	3.14E+03†	1.43E+03†	9.38E+02†	1.71E+04†
C08	**7.51E+02**	5.92E+04†	2.16E+04†	8.12E+04†	8.00E+03†	3.61E+04†	2.93E+04†
C09	**2.58E+12**	3.57E+12	5.59E+12†	5.36E+12†	5.16E+12†	3.54E+12	7.60E+12†
C10	4.94E+12	4.04E+12	4.30E+12	6.25E+12†	4.10E+12	**3.64E+12**	7.45E+12†
C13	**-5.82E+01**	-5.38E+01†	-5.47E+01†	-5.44E+01†	-5.48E+01†	-5.54E+01†	-5.49E+01†
C14	**2.13E+05**	3.77E+11†	3.98E+11†	9.56E+11†	1.25E+11†	4.90E+10†	8.06E+11†
C15	**3.42E+11**	1.11E+13†	2.71E+13†	2.87E+13†	1.34E+13†	5.23E+12†	3.59E+13†
C16	**9.54E-01**	1.00E+00†	1.03E+00†	1.03E+00†	1.01E+00†	1.01E+00†	1.04E+00†
C17	**1.58E+02**	2.19E+02	3.07E+02†	3.13E+02†	2.20E+02	1.75E+02	3.83E+02†
C18	**3.04E+03**	5.04E+03†	5.82E+03†	6.01E+03†	4.70E+03†	4.23E+03	5.79E+03†
w/t/l	**-**	**0/5/8**	**0/2/11**	**0/1/12**	**0/3/10**	**1/4/8**	**0/1/12**

Tabla 7.4: Resultados obtenidos por cada BCHM con PSO en los problemas del CEC-2010 en 10D. En esta tabla se muestran los resultados de los problemas para los cuales al menos uno de los métodos no logró encontrar soluciones factibles en al menos una corrida independiente.

Problema	ABCHS	Cen&DeB	Evo&DeB	Ref&DeB	Ran&RaB	Wra&RaB	Bou&DeB
C03	1.12E+13	4.77E+13(17)	2.68E+13(4)	4.58E+13(5)	4.31E+13(5)	-	**8.29E+12**(4)
C06	**3.58E+02**	3.90E+02	3.95E+02	4.39E+02	4.08E+02	3.92E+02(24)	4.18E+02
C12	**-6.13E+00(14)**	-	-	-	-	3.38E-01(1)	-
w/t/l	**-**	**0/2/0**	**0/2/0**	**0/2/0**	**0/2/0**	**0/2/0**	**0/2/0**

se observa que sólo el esquema adaptativo encontró soluciones factibles en las tres funciones, el resto de los métodos sólo encontraron resultados factibles en dos de ellos. A ABCHS le sigue en rendimiento el método *Cen&DeB*, considerando aquí el número de ejecuciones factibles.
De los resultados obtenidos en las funciones de dimensionalidad media (10D) del CEC2010, el esquema adaptativo obtuvo el mejor rendimiento general en la mayoría de las funciones, seguido de los métodos *Wra&RaB* y *Cen&DeB*.

En la Tabla 7.5 se presenta el promedio de los valores obtenidos en 25 corridas independientes para cada uno de los problemas del CEC-2010 para 30 dimensiones, las funciones C03, C04, C11 y C12 se excluyeron de este experimento ya que ninguno de los métodos fue capaz de obtener

Tabla 7.5: Resultados obtenidos por cada BCHM comparado, en PSO para los problemas del CEC-2010 en 30D. El símbolo † significa que existe *SSD* a favor del método base (ABCHS) y ∗ representa *SSD* en contra del método base.

Problema	ABCHS	Cen&DeB	Evo&DeB	Ref&DeB	Ran&RaB	Wra&RaB	Bou&DeB
C01	**-5.95E-01**	-3.15E-01†	-4.58E-01†	-4.66E-01†	-2.94E-01†	-3.18E-01†	-4.28E-01†
C02	3.69E+00	3.81E+00	4.10E+00	4.16E+00†	3.81E+00	**3.59E+00**	4.05E+00
C05	**2.38E+02**	3.83E+02†	4.63E+02†	4.17E+02†	4.30E+02†	3.89E+02†	4.72E+02†
C06	**3.82E+02**	4.38E+02†	5.06E+02†	4.89E+02†	4.57E+02†	4.76E+02†	5.40E+02†
C07	**7.58E+03**	5.71E+08†	2.66E+08†	3.93E+08†	4.36E+08†	1.97E+08†	4.81E+08†
C08	**2.61E+05**	7.58E+08†	6.08E+08†	6.27E+08†	7.56E+08†	6.55E+08†	7.08E+08†
C09	1.76E+13	**1.12E+13***	1.87E+13†	2.34E+13†	1.69E+13	1.20E+13	2.35E+13†
C10	**9.62E+12**	1.04E+13†	1.77E+13†	1.99E+13†	1.40E+13†	1.35E+13†	2.37E+13†
C13	**-5.43E+01**	-4.30E+01†	-4.50E+01†	-4.38E+01†	-4.32E+01†	-4.39E+01†	-4.55E+01†
C14	**8.66E+08**	3.09E+12†	4.98E+12†	4.04E+12†	2.68E+12†	1.45E+12†	5.25E+12†
C15	**2.93E+11**	3.90E+13†	4.68E+13†	3.94E+13†	2.98E+13†	1.60E+13†	5.78E+13†
C16	**1.03E+00**	1.07E+00†	1.08E+00†	1.08E+00†	1.08E+00†	1.07E+00†	1.09E+00†
C17	**3.75E+02**	7.22E+02†	1.02E+03†	1.12E+03†	8.67E+02†	7.97E+02†	1.11E+03†
C18	**4.41E+03**	1.37E+04†	1.65E+04†	1.48E+04†	1.51E+04†	1.30E+04†	1.59E+04†
w/t/l	**-**	**1/1/12**	**0/1/13**	**0/0/14**	**0/2/12**	**0/2/12**	**0/1/13**

soluciones factibles. En esta tabla, se observa que todos los métodos comparados obtuvieron la misma cantidad de corridas factibles, con esto, los métodos mostraron una capacidad similar para alcanzar la región factible. Sin embargo, al considerar la calidad de las soluciones encontradas, existen grandes diferencias entre los métodos comparados.

Teniendo en cuenta los resultados de las pruebas estadísticas se observa que el esquema adaptativo obtuvo mejores soluciones que el resto de los métodos en al menos 12 de las 14 funciones presentadas en esta tabla, con respecto a los mejores valores promedio también se observa que el esquema adaptativo obtuvo los mejores valores en 12 funciones.

El segundo método en orden de rendimiento fue *Cen&DeB*, que perdió, según las pruebas estadísticas, en 12 funciones contra el esquema adaptativo, lo empató en una función y obtuvo mejores resultados en una función (C09), para la cual el método *Cen&DeB* también obtuvo el mejor valor promedio; posteriormente, el método $Wra\&RaB$ perdió en 12 funciones contra el ABCHS y empató en dos funciones, en una de las cuales obtuvo el mejor valor promedio.

De la Tabla 7.5 se concluye que el esquema adaptativo para el manejo de restricciones de límite (*ABCHS*) obtuvo el mejor rendimiento general en funciones de alta dimensionalidad (30 dimensiones) del CEC-2010, seguido de los métodos *Cen&DeB* y $Wra\&RaB$.

La Figura 7.1 (A) muestra el número de ejecuciones (corridas) independientes (de los 60 problemas considerados en el presente experimento) en las que se encontraron soluciones factibles por

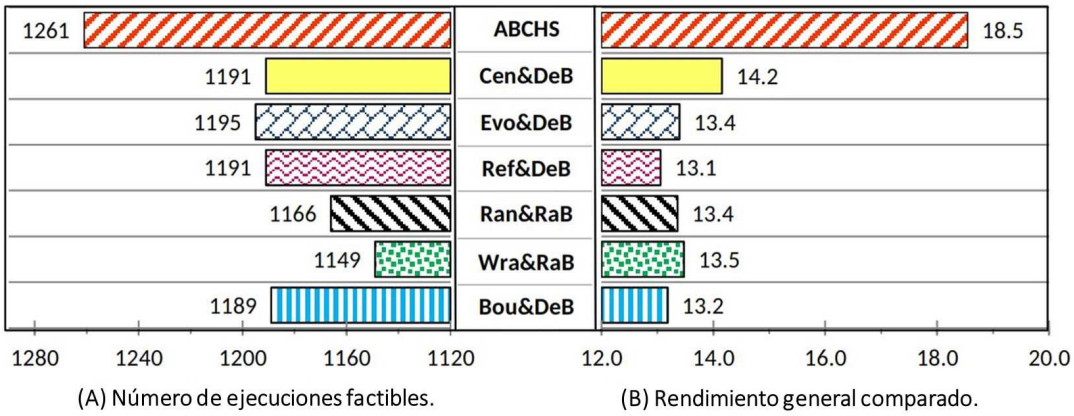


Figura 7.1: Resumen gráfico de los resultados experimentales obtenidos al comparar el rendimiento del *ABCHS* contra seis métodos para el manejo de restricciones de límite dentro del algoritmo PSO.

cada uno de los métodos comparados. Como se puede apreciar en esta figura, el ABCHS obtuvo la mayor cantidad de ejecuciones factibles, seguido por cuatro métodos: $Evo\&DeB$, $Cen\&DeB$, $Ref\&DeB$ y $Bou\&DeB$, que obtuvieron un número similar de ejecuciones factibles entre ellos.
La Figura 7.1 (B) muestra el valor del rendimiento general comparado (*OCP*). En esta figura, se observa que el ABCHS obtuvo el mayor valor *OCP*, lo que indica que sus soluciones factibles fueron las de mayor calidad, seguido por los métodos $Cen\&DeB$ y $Wra\&RaB$.

Tabla 7.6: Tiempo de procesamiento empleado por cada uno de los BCHM comparados dentro del algoritmo de PSO clásico al realizar 10,000 evaluaciones.

BCHM	**T2**	**(T2-T1)/T1**
$Ref\&DeB$	**1.66**	**1.09**
$Bou\&DeB$	1.68	1.11
$Evo\&DeB$	1.68	1.12
ABCHS	1.74	1.19
$Cen\&DeB$	1.87	1.35
$Ran\&RaB$	1.87	1.35
$Wra\&RaB$	1.99	1.50

En la Tabla 7.6 se muestra el tiempo computacional consumido por cada uno de los BCHM comparados al realizar 10,000 evaluaciones, dentro del algoritmo de PSO clásico. Los métodos se encuentran ordenados de menor a mayor consumo de tiempo, por lo tanto, los métodos que se ubican en las filas superiores son los más rápidos. El valor de $T1 = 0.79$ fue el mismo para todos

los BCHM.

Como se aprecia en esta tabla, el método con el menor consumo de tiempo es *Ref&DeB*, seguido de los métodos *Bou&DeB* y *Evo&DeB*. El *ABCHS* propuesto se encuentra a media tabla, siendo más rápido que los métodos *Cen&DeB*, *Ran&RaB* y *Wra&RaB*.

A partir de los resultados presentados en esta tabla también el posible considerar que el tiempo computacional consumido por cada BCHM no sólo depende de su complejidad, pues como se puede observar en la subsección 3.2, el método *Random* tiene una complejidad menor que el método *Evolutionary*, sin embargo *Ran&RaB* consume más tiempo computacional que *Evo&DeB*. Esta observación ya ha sido reportada en trabajos anteriores [26], donde se ha observado que algunos métodos tienden a realizar una cantidad menor de reparaciones que otros, por lo tanto, su consumo de tiempo no sólo depende de la complejidad del método, sino también de la cantidad de reparaciones que realiza.

Con referencia al *ABCHS* propuesto, también se debe considerar que su consumo de tiempo depende del tiempo que consumen los métodos (BCHM) que emplea y principalmente del consumo de tiempo computacional de su método base ($bchm_1$), ya que este método se usa en la primera fase del proceso evolutivo, cuando todavía no existen soluciones factibles, que coincide precisamente con la etapa en la que se realiza una mayor cantidad de reparaciones [5].

En este experimento el ABCHS utilizó al método *Ran&RaB* como su método base, el consumo de tiempo de *Ran&RaB* se muestra en la penúltima fila de la Tabla 7.6; y como se puede ver, el tiempo de procesamiento del *ABCHS* fue ligeramente mejor que el de su método base.

Conclusiones del presente experimento:

En el presente experimento se comparó el rendimiento de siete métodos para el manejo de restricciones de límite (*BCHM*) dentro del algoritmo PSO aplicando las reglas de Deb para la optimización restringida, los resultados hacen evidente que la elección de un BCHM en particular influye directamente en el rendimiento del algoritmo PSO, tanto en su capacidad para llegar a la región factible como en la calidad de las soluciones encontradas.

En general, se demostró que el esquema adaptativo propuesto (ABCHS) obtuvo el mejor desempeño, seguido por el método *Cen&DeB*. El esquema adaptativo superó al resto de los métodos tanto en la cantidad de soluciones factibles encontradas como en la calidad de las mismas.

En cuanto al tiempo de procesamiento, se observó que aun cuando el ABCHS implementa más operaciones que el resto de los métodos, su consumo de tiempo no es el peor, más bien depende del tiempo que consuma el método que utilice como su método base. Por otro lado, también se observó que el ABCHS consumió menos tiempo de procesamiento que el método *Cen&DeB*, que fue el método que obtuvo la segunda posición en cuanto a desempeño en el conjunto de problemas del presente experimento.

Las principales características del ABCHS fueron: su capacidad para encontrar soluciones de mejor calidad que el resto de los métodos en 61 % de las funciones probadas, como su notable rendimiento en funciones de dimensionalidad media a alta, así como en aquellas funciones con una mayor cantidad de restricciones y funciones para las cuales fue más difícil llegar a la región factible.

7.1.2. Experimento 2. Comparativo del rendimiento del ABCHS dentro del algoritmo SAM-PSO

Después de encontrar diferencias estadísticamente significativas en el rendimiento de los métodos para el manejo de restricciones de límite (BCHM) dentro del algoritmo de PSO clásico empleando las reglas de Deb, en esta subsección, para analizar más a fondo el rendimiento de los métodos comparados, se puso a prueba su desempeño dentro de un algoritmo del estado del arte basado en PSO para la optimización restringida denominado: mezcla autoadaptativa de metodologías de cúmulos de partículas, SAM-PSO *(Self-Adaptive Mix of Particle Swarm Methodologies)* para optimización restringida [14].

Las Tablas 7.7 y 7.8 presentan el promedio de los valores obtenidos en 25 ejecuciones independientes para los problemas de referencia del CEC-2006, la función C20 se excluyó de este experimento ya que ninguno de los métodos comparados fue capaz de obtener soluciones factibles.

La Tabla 7.7 presenta los resultados obtenidos en 16 funciones para las cuales no se encontraron diferencias en el número de ejecuciones factibles o en las pruebas estadísticas, la única diferencia que se puede observar en algunas funciones de esta tabla es en los valores promedio de las 25 ejecuciones independientes, en este sentido, el *ABCHS* obtuvo los mejores valores promedio en 15 funciones, seguido por *Cen&DeB* y *Evo&DeB*, que obtuvieron los mejores valores promedio en 12 funciones.

La Tabla 7.8 presenta los valores promedio de 25 ejecuciones independientes para las restantes siete funciones del CEC-2006, en las que se encontraron diferencias, ya sea en el número de ejecuciones factibles o en las pruebas estadísticas. En esta tabla se observa que los métodos *ABCHS*, *Evo&DeB* y *Cen&DeB* fueron los que obtuvieron soluciones factibles en todas las funciones, los mejores resultados en las pruebas estadísticas y los mejores valores promedio.

Considerando las pruebas estadísticas, el *ABCHS* fue capaz de obtener los mejores resultados, seguido de *Cen&DeB*, que obtuvo un mejor desempeño que el *ABCHS* en una función, obtuvo un rendimiento similar en cuatro funciones y perdió en dos de ellas, posteriormente el método *Evo&DeB* obtuvo un mejor desempeño que el *ABCHS* en una función, lo empató en cinco y perdió en una función.

Por lo tanto, el esquema adaptativo propuesto (*ABCHS*) logró obtener el mejor rendimiento en

Tabla 7.7: Resultados obtenidos por cada BCHM con SAM-PSO para los problemas del CEC 2006 donde no se encontraron diferencias ya sea en el número de ejecuciones factibles o en las pruebas estadísticas.

Problema	ABCHS	Cen&DeB	Evo&DeB	Ref&DeB	Ran&RaB	Wra&RaB	Bou&DeB
g01	**-1.50E+01**	**-1.50E+01**	**-1.50E+01**	**-1.50E+01**	**-1.50E+01**	**-1.50E+01**	**-1.50E+01**
g03	**-1.00E+00**	**-1.00E+00**	**-1.00E+00**	**-1.00E+00**	**-1.00E+00**	**-1.00E+00**	**-1.00E+00**
g04	**-3.07E+04**	**-3.07E+04**	**-3.07E+04**	**-3.07E+04**	**-3.07E+04**	**-3.07E+04**	**-3.07E+04**
g05	**5.13E+03**	5.25E+03	**5.13E+03**	**5.13E+03**	5.26E+03	5.25E+03	5.26E+03
g06	**-6.96E+03**	**-6.96E+03**	**-6.96E+03**	**-6.96E+03**	**-6.96E+03**	**-6.96E+03**	**-6.96E+03**
g08	**-9.58E-02**	**-9.58E-02**	**-9.58E-02**	**-9.58E-02**	**-9.58E-02**	**-9.58E-02**	**-9.58E-02**
g09	**6.81E+02**	**6.81E+02**	**6.81E+02**	**6.81E+02**	**6.81E+02**	**6.81E+02**	**6.81E+02**
g10	7.05E+03	7.05E+03	**7.05E+03**	7.05E+03	**7.05E+03**	7.05E+03	7.31E+03
g11	**7.50E-01**	**7.50E-01**	**7.50E-01**	**7.50E-01**	**7.50E-01**	**7.50E-01**	**7.50E-01**
g12	**-1.00E+00**	**-1.00E+00**	**-1.00E+00**	**-1.00E+00**	**-1.00E+00**	**-1.00E+00**	**-1.00E+00**
g14	**-4.78E+01**	-4.78E+01	-4.78E+01	-4.78E+01	-4.78E+01	-4.38E+01	-4.78E+01
g15	**9.62E+02**	9.62E+02	9.62E+02	9.63E+02	9.62E+02	9.63E+02	9.63E+02
g16	**-1.91E+00**	**-1.91E+00**	**-1.91E+00**	**-1.91E+00**	**-1.91E+00**	**-1.91E+00**	**-1.91E+00**
g17	**8.85E+03**	**8.85E+03**	8.94E+03	8.85E+03	8.85E+03	8.94E+03	8.94E+03
g18	**-8.66E-01**	**-8.66E-01**	-8.66E-01	-8.66E-01	-8.66E-01	-7.16E-01	**-8.66E-01**
g24	**-5.51E+00**	**-5.51E+00**	**-5.51E+00**	**-5.51E+00**	**-5.51E+00**	**-5.51E+00**	**-5.51E+00**
w/t/l	**-**	**0/16/0**	**0/16/0**	**0/16/0**	**0/16/0**	**0/16/0**	**0/16/0**

Tabla 7.8: Resultados obtenidos por cada BCHM con SAM-PSO para los problemas del CEC-2006 en los cuales se encontraron diferencias, ya sea en el número de ejecuciones factibles o en las pruebas estadísticas. El símbolo † significa que existe *SSD* a favor del método base (ABCHS) y $*$ representa *SSD* en contra del método base.

Problema	ABCHS	Cen&DeB	Evo&DeB	Ref&DeB	Ran&RaB	Wra&RaB	Bou&DeB
g02	**-8.04E-01**	-7.96E-01†	-7.96E-01†	-7.96E-01†	-7.36E-01†	-7.66E-01†	-7.96E-01†
g07	**2.43E+01**	**2.43E+01**	**2.43E+01**	**2.43E+01**	2.48E+01†	2.84E+01†	2.49E+01†
g13	5.39E-02	5.47E-02†	5.39E-02	**5.39E-02**	5.89E-02†	5.81E-02†	5.55E-02†
g19	**3.27E+01**	3.27E+01	3.27E+01	3.27E+01	3.71E+01†	3.54E+01†	3.27E+01
g21	1.94E+02	1.94E+02	**1.94E+02**	1.94E+02†	1.94E+02†	1.94E+02	1.94E+02
g22	**2.59E+02(18)**	**2.59E+02**(14)	2.85E+02(15)	-	-	-	-
g23	-4.00E+02	**-4.00E+02***	**-4.00E+02***	-4.00E+02	-4.17E+01†	-4.27E+01†	-4.29E+01†
w/t/l	**-**	**1/4/2**	**1/5/1**	**0/4/2**	**0/0/6**	**0/1/5**	**0/2/4**

los problemas del conjunto de funciones del CEC-2006, seguido de los métodos *Cen&DeB* y *Evo&DeB*.

Las Tablas 7.9 y 7.10 muestran el promedio de los valores obtenidos mediante 25 ejecuciones independientes por cada método de manejo de restricciones de límite que se ejecutó dentro del algoritmo SAM-PSO para los problemas de referencia CEC-2010 en 10 dimensiones. La Tabla 7.9

Tabla 7.9: Resultados obtenidos por cada BCHM con SAM-PSO para el conjunto de problemas del CEC-2010 en 10 dimensiones. En esta tabla sólo se presentan los resultados para los problemas en los cuales no se encontraron diferencias en el número de ejecuciones factibles o en las pruebas estadísticas.

Problema	ABCHS	Cen&DeB	Evo&DeB	Ref&DeB	Ran&RaB	Wra&RaB	Bou&DeB
C01	**-6.67E-01**	-5.35E-01	**-6.67E-01**	-6.41E-01	**-6.67E-01**	-5.07E-01	-4.53E-01
C02	-2.26E+00	-2.27E+00	-2.26E+00	**-2.28E+00**	**-2.28E+00**	-2.26E+00	-2.26E+00
C03	**0.00E+00**	**0.00E+00**	2.07E-01	**0.00E+00**	1.56E-01	4.91E-02	2.64E-02
C04	-9.99E-06	-9.97E-06	-9.97E-06	**-9.99E-06**	-9.94E-06	-9.98E-06	-9.99E-06
C06	-5.63E+02	-5.71E+02	**-5.73E+02**	-5.36E+02	-5.58E+02	-5.36E+02	-5.63E+02
C07	**0.00E+00**	6.51E-01	**0.00E+00**	2.99E-01	1.56E-01	7.50E-02	1.04E-01
C08	**7.86E+00**	7.88E+00	**7.86E+00**	7.87E+00	7.87E+00	7.87E+00	7.88E+00
C09	3.36E-28	4.51E-28	3.52E-28	3.69E-28	3.98E-28	3.48E-28	**3.10E-28**
C17	**9.94E-02**	3.21E-01	2.10E-01	2.34E-01	**9.94E-02**	1.00E-01	3.02E-01
C18	**3.60E-03**	**3.60E-03**	2.45E-02	2.43E-02	9.93E-02	**3.60E-03**	1.49E-02
w/t/l	**-**	**0/10/0**	**0/10/0**	**0/10/0**	**0/10/0**	**0/10/0**	**0/10/0**

Tabla 7.10: Resultados obtenidos por cada BCHM con SAM-PSO para los problemas del CEC-2010 10D en los cuales se encontraron diferencias estadísticamente significativas. El símbolo † significa que existe *SSD* a favor del método base (ABCHS) y $*$ representa *SSD* en contra del método base.

Problema	ABCHS	Cen&DeB	Evo&DeB	Ref&DeB	Ran&RaB	Wra&RaB	Bou&DeB
CO5	**-4.28E+02**	-4.12E+02	-4.23E+02	-4.19E+02	-4.18E+02	-4.08E+02†	-4.12E+02
C10	**6.93E-29**	6.40E+01†	3.57E+01†	9.66E+01†	5.97E+01†	1.04E+02†	1.57E+02†
C11	-1.43E-03	**-1.45E-03***	-1.06E-03	-1.16E-03	-1.18E-03	-1.27E-03	-5.44E-04†
C12	**-1.99E-01**	-1.99E-01	-1.99E-01	-1.98E-01	-1.98E-01†	**-1.99E-01**	-1.99E-01
C13	**-6.63E+01**	-5.40E+01†	-5.81E+01†	-5.67E+01†	-5.87E+01†	-5.88E+01†	-5.63E+01†
C14	**0.00E+00**	**0.00E+00**	3.71E+02†	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.43E+02†
C15	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	7.95E+02†	**0.00E+00**
C16	**0.00E+00**	2.69E-01†	2.97E-01†	1.95E-01	2.77E-01†	**0.00E+00**	8.62E-02
w/t/l	**-**	**1/4/3**	**0/4/4**	**0/6/2**	**0/4/4**	**0/4/4**	**0/4/4**

muestra los resultados obtenidos en 10 funciones en las que no se encontraron diferencias en las pruebas estadísticas ni en el número de ejecuciones factibles. En cuanto a los valores promedio de

las 25 ejecuciones independientes, se observa que el *ABCHS* obtuvo los mejores resultados en seis funciones, seguido por el método *Evo&DeB* que obtuvo los mejores valores promedio en cuatro funciones.

La Tabla 7.10 presenta los valores promedio obtenidos en las ocho funciones en las que se encontraron diferencias estadísticamente significativas.

Considerando las pruebas estadísticas, se observa que en estas ocho funciones el esquema adaptativo *ABCHS* obtuvo el mejor rendimiento, seguido por el método *Cen&DeB*, que obtuvo mejores resultados que el *ABCHS* en una función (C11), lo empató en cuatro funciones y perdido en tres. En orden de rendimiento siguió el método *Ref&DeB*, quien empató al *ABCHS* en seis funciones y perdió en dos.

Además, es posible observar que el *ABCHS* obtuvo los mejores valores promedio en siete funciones, seguido por los métodos *Cen&DeB* y *Wra&RaB* que obtuvieron los mejores valores promedio en tres funciones.

Considerando lo anterior, se concluye que el *ABCHS* obtuvo el mejor rendimiento en los problemas de CEC-2010 en 10 dimensiones, seguido por los métodos *Cen&DeB* y *Ref&DeB*.

Las Tablas 7.11 y 7.12 presentan los resultados obtenidos en los problemas CEC-2010 en 30 dimensiones. En la Tabla 7.11 se presentan los valores promedio obtenidos en 11 funciones en las que no se encontraron diferencias estadísticamente significativas, sin embargo, como se observa, el *esquema adaptativo para el manejo de restricciones de límite* (*ABCHS*) obtuvo los mejores valores promedio en cinco funciones, seguidos de los métodos *Cen&DeB* y *Ref&DeB* que obtuvieron los mejores valores promedio en tres funciones.

La Tabla 7.12 muestra los resultados obtenidos en las siete funciones restantes en las que se encontraron diferencias estadísticamente significativas.

Teniendo en cuenta los resultados de las pruebas estadísticas el *ABCHS* obtuvo el mejor rendimiento, seguido por los métodos *Evo&DeB* y *Bou&DeB*, que fueron superados por el *ABCHS* en una función y lo empataron en seis.

Teniendo en cuenta los valores promedio el *ABCHS* obtuvo los mejores resultados en tres funciones, seguido de *Cen&DeB* que obtuvo los mejores valores promedio en dos funciones, posteriormente le siguieron los métodos *Evo&DeB* y *Ref&DeB*, los cuales obtuvieron el mejor promedio en una función.

Con base en los resultados de las Tablas 7.11 y 7.12, concluimos que el *ABCHS* obtuvo el mejor rendimiento en los problemas CEC-2010 en 30 dimensiones, seguido por el método *Evo&DeB*.

En la Figura 7.2 se muestran dos gráficas de barras que representan *el número de ejecuciones factibles* (A) y el valor del *rendimiento general comparado* (OCP) (B). El gráfico de barras izquierdo (A) muestra que no hay una diferencia importante en el número de ejecuciones factibles entre los

Tabla 7.11: Resultados obtenidos por cada BCHM con SAM-PSO para el conjunto de problemas del CEC-2010 en 30 dimensiones. En esta tabla sólo se presentan los resultados obtenidos en los problemas en donde no se encontraron diferencias ni en el número de ejecuciones factibles ni en las pruebas estadísticas.

Problema	ABCHS	Cen&DeB	Evo&DeB	Ref&DeB	Ran&RaB	Wra&RaB	Bou&DeB
C01	**-8.05E-01**	**-8.05E-01**	-8.03E-01	-7.98E-01	-7.98E-01	-7.99E-01	-8.00E-01
C02	**-2.28E+00**	-2.27E+00	**-2.28E+00**	-2.28E+00	-2.28E+00	-2.28E+00	-2.28E+00
C03	2.40E-07	2.29E-07	2.41E-07	**2.07E-07**	2.24E-07	2.14E-07	2.50E-07
C04	-3.33E-06	-3.33E-06	-3.33E-06	**-3.33E-06**	-3.33E-06	-3.33E-06	-3.33E-06
C07	3.47E-14	**3.39E-14**	3.39E-14	3.66E-14	3.55E-14	3.44E-14	3.45E-14
C09	**2.69E-04**	3.05E-04	3.10E-04	2.85E-04	2.86E-04	2.74E-04	2.71E-04
C11	**-3.92E-04**	-3.92E-04	-3.92E-04	-3.92E-04	-3.92E-04	-3.92E-04	-3.92E-04
C12	**-3.03E+00**	-1.96E+00	-2.11E+00	-1.91E+00	-2.22E+00	-2.77E+00	-1.77E+00
C13	-6.53E+01	-6.34E+01	**-6.60E+01**	-6.59E+01	-6.57E+01	-6.55E+01	-6.49E+01
C16	2.81E-04	**2.75E-04**	2.80E-04	2.87E-04	**2.75E-04**	2.87E-04	2.77E-04
C17	3.35E-03	2.41E-03	4.73E-03	**1.00E-03**	3.51E-03	3.63E-03	5.02E-03
w/t/l	**-**	**0/11/0**	**0/11/0**	**0/11/0**	**0/11/0**	**0/11/0**	**0/11/0**

Tabla 7.12: Resultados obtenidos por cada BCHM con SAM-PSO en los problemas del CEC-2010 30D donde se encontraron diferencias estadísticamente significativas. El símbolo † significa que existe *SSD* a favor del método base (ABCHS) y ∗ representa *SSD* en contra del método base.

Problema	ABCHS	Cen&DeB	Evo&DeB	Ref&DeB	Ran&RaB	Wra&RaB	Bou&DeB
C05	-1.97E+02	**-4.48E+02***	-2.44E+02	-1.35E+02†	-1.04E+02†	-1.94E+02	-2.58E+02
C06	**-4.46E+02**	-6.48E+01†	-4.29E+02	4.64E+01†	-2.87E+02	-2.53E+02	-2.57E+00†
C08	**8.23E+01**	8.23E+01†	8.23E+01	8.23E+01	8.23E+01†	8.23E+01†	8.23E+01
C10	6.29E+02	5.66E+02	**4.92E+02**	1.07E+03	5.40E+03†	5.52E+03†	2.69E+03
C14	1.21E+03	**2.84E+01**	1.01E+03	1.49E+03†	3.38E+01	1.26E+03	3.85E+01
C15	**1.03E+03**	1.53E+04†	5.20E+03	1.36E+04	1.32E+04	1.58E+04†	1.42E+04
C18	6.21E+01	2.32E+01	7.88E+01†	**1.52E+01***	4.08E+01	6.08E+01	3.24E+01
w/t/l	**-**	**1/3/3**	**0/6/1**	**1/3/3**	**0/4/3**	**0/4/3**	**0/6/1**

métodos comparados. Sin embargo, en términos de la calidad de las soluciones encontradas (el valor OCP), el gráfico de barras derecho (B) muestra que el mejor rendimiento lo obtiene el *ABCHS*, seguido de los métodos *Cen&DeB* y *Evo&DeB*.

Conclusiones del presente experimento:

Los resultados obtenidos en el presente experimento mostraron que el *BCHM* empleado dentro del algoritmo SAM-PSO no logró hacer una diferencia importante en la capacidad para alcanzar la región factible, pues el número de corridas factibles fue muy similar entre métodos; probablemente

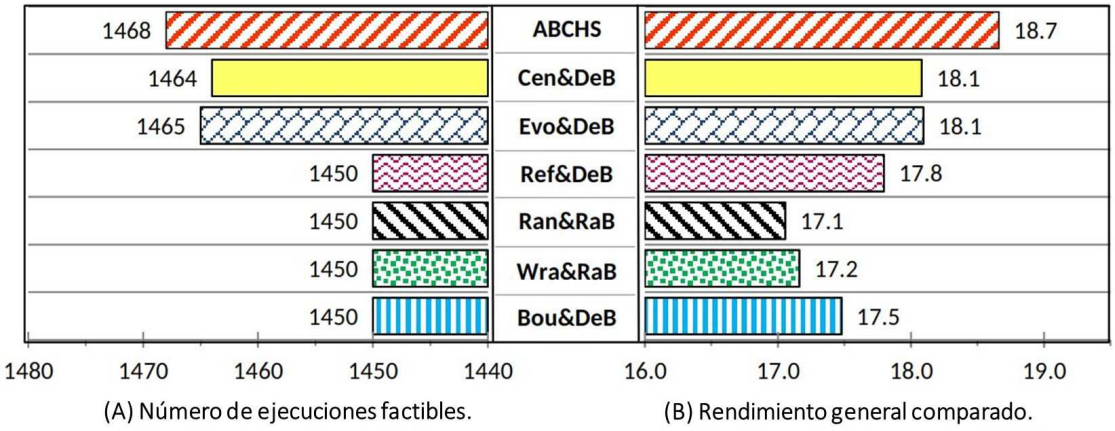


Figura 7.2: Resumen gráfico de los resultados experimentales obtenidos al comparar el rendimiento del *ABCHS* contra seis métodos para el manejo de restricciones de límite dentro del algoritmo SAM-PSO.

debido a que SAM-PSO es un algoritmo especializado en optimización restringida con excelente capacidad para alcanzar la región factible y, como sus autores han comentado, SAM-PSO puede manejar problemas de manera efectiva con regiones factibles pequeñas y puede tratar con problemas escalables [14].

Sin embargo, al comparar la calidad de las soluciones encontradas, las diferencias son obvias; con el esquema adaptativo propuesto (*ABCHS*) se obtuvieron las soluciones factibles de mejor calidad, seguido por los métodos *Cen&DeB*y *Evo&DeB*.

7.2. Análisis de ABCHS dentro del algoritmo DE

7.2.1. Experimento 3. Comparativo del rendimiento del ABCHS dentro del algoritmo DE clásico

En este experimento se comparó el rendimiento del esquema adaptativo para el manejo de restricciones de límite (*ABCHS*) contra siete métodos para el manejo de restricciones de límite (*BCHMs*) dentro del algoritmo clásico de Evolución Diferencial empleando reglas de Deb para el manejo de las restricciones funcionales.

Las Tablas 7.13 y 7.14 muestran el promedio de los valores de aptitud obtenidos en 25 ejecuciones independientes para cada una de las funciones del CEC-2006. Las funciones g20 y g22 (funciones con el mayor número de restricciones de igualdad) fueron excluidas, ya que ninguno de los métodos fue capaz de obtener soluciones factibles.

La Tabla 7.13 muestra los resultados de 12 funciones en las que no se encontraron diferencias en el número de ejecuciones factibles o en las pruebas estadísticas. Considerando, por lo tanto, sólo los mejores promedios de aptitud, encontramos que los métodos *Centroid*, *Reflection*, *Random* y *Wrapping* obtuvieron los mejores promedios en nueve funciones, seguidos por los métodos: *ABCHS*, *Res&Ran* y *Evolutionary*, que obtuvieron el mejor promedio de aptitud en siete funciones, y finalmente el método *Boundary* obtuvo el mejor promedio en seis funciones.

La Tabla 7.14 muestra el promedio de los valores de aptitud obtenido en las 10 funciones restantes

Tabla 7.13: Resultados obtenidos por cada *BCHM* con Evolución Diferencial para los problemas del CEC-2006 en los cuales no se encontraron diferencias ya sea en el número de ejecuciones factibles o en las pruebas estadísticas.

Problema	ABCHS	Res&Ran	Centroid	Evolutionary	Reflection	Random	Wrapping	Boundary
g04	**-3.07E+04**	**-3.07E+04**	**-3.07E+04**	**-3.07E+04**	**-3.07E+04**	**-3.07E+04**	**-3.07E+04**	**-3.07E+04**
g05	5.13E+03	5.13E+03	**5.13E+03**	5.13E+03	5.13E+03	5.13E+03	5.13E+03	5.13E+03
g07	2.43E+01	2.43E+01	2.43E+01	2.43E+01	2.43E+01	2.43E+01	**2.43E+01**	2.43E+01
g08	**-9.58E-02**	**-9.58E-02**	**-9.58E-02**	**-9.58E-02**	**-9.58E-02**	**-9.58E-02**	**-9.58E-02**	**-9.58E-02**
g09	6.81E+02	**6.81E+02**	**6.81E+02**	6.81E+02	**6.81E+02**	**6.81E+02**	**6.81E+02**	6.81E+02
g10	**7.05E+03**	**7.05E+03**	**7.05E+03**	**7.05E+03**	**7.05E+03**	**7.05E+03**	**7.05E+03**	7.06E+03
g12	**-1.00E+00**	**-1.00E+00**	**-1.00E+00**	**-1.00E+00**	**-1.00E+00**	**-1.00E+00**	**-1.00E+00**	**-1.00E+00**
g14	-4.78E+01	-4.78E+01	-4.78E+01	-4.78E+01	**-4.78E+01**	**-4.78E+01**	**-4.78E+01**	-4.78E+01
g15	**9.62E+02**	**9.62E+02**	**9.62E+02**	**9.62E+02**	**9.62E+02**	**9.62E+02**	**9.62E+02**	**9.62E+02**
g16	**-1.91E+00**	-1.91E+00	-1.91E+00	**-1.91E+00**	**-1.91E+00**	**-1.91E+00**	-1.91E+00	**-1.91E+00**
g18	-8.66E-01	-8.58E-01	**-8.66E-01**	-8.66E-01	-8.66E-01	-8.58E-01	-8.58E-01	-8.66E-01
g24	**-5.51E+00**	**-5.51E+00**	**-5.51E+00**	**-5.51E+00**	**-5.51E+00**	**-5.51E+00**	**-5.51E+00**	**-5.51E+00**
w/t/l	**-**	**0/12/0**	**0/12/0**	**0/12/0**	**0/12/0**	**0/12/0**	**0/12/0**	**0/12/0**

Tabla 7.14: Resultados obtenidos por cada *BCHM* con Evolución Diferencial para los problemas del CEC-2006 en los cuales se encontraron diferencias en el número de ejecuciones factibles o en las pruebas estadísticas. El símbolo † significa que existe *SSD* a favor del método base (ABCHS) y $*$ representa *SSD* en contra del método base.

Problema	ABCHS	Res&Ran	Centroid	Evolutionary	Reflection	Random	Wrapping	Boundary
g01	**-1.50E+01**	**-1.50E+01**	-1.50E+01	**-1.50E+01**	**-1.50E+01**	**-1.50E+01**	**-1.50E+01**	-1.36E+01†
g02	-6.96E-01	-6.45E-01	-5.91E-01†	-7.77E-01*	**-7.87E-01***	-7.57E-01	-7.56E-01	-6.83E-01
g03	-2.98E-01	-1.78E-01†	**-3.23E-01**	-1.06E-01(22)†	-5.26E-02†	-1.65E-01(23)†	-1.32E-01(23)†	-5.73E-03†
g06	**-6.96E+03**	**-6.96E+03**	**-6.96E+03**	**-6.96E+03**	**-6.96E+03**	**-6.96E+03**	**-6.96E+03**	**-6.96E+03**(24)
g11	7.84E-01	7.57E-01	7.60E-01	**7.50E-01**	7.51E-01	7.61E-01	7.56E-01	9.62E-01†
g13	3.46E-01	2.85E-01	3.38E-01	2.69E-01	4.28E-01(24)	3.50E-01(24)	**2.63E-01**(24)	3.02E-01(24)
g17	8.91E+03	8.91E+03	8.90E+03(24)	8.90E+03	**8.90E+03**	8.92E+03	8.91E+03	8.90E+03
g19	3.27E+01	3.27E+01†	3.27E+01†	3.27E+01	3.27E+01	3.27E+01†	3.27E+01†	**3.27E+01**
g21	**1.94E+02**	1.99E+02	1.94E+02	2.62E+02†	2.62E+02†	1.99E+02	2.09E+02	1.94E+02(10)
g23	**-4.00E+02**	-4.00E+02†	-4.00E+02	-4.00E+02	-4.00E+02	-4.00E+02	-4.00E+02	8.69E+01(23)†
w/t/l	**-**	**0/7/3**	**0/8/2**	**1/7/2**	**1/7/2**	**0/8/2**	**0/8/2**	**0/6/4**

en las que se encontraron diferencias entre los métodos, ya sea en el número de ejecuciones factibles o en las pruebas estadísticas.

Teniendo en cuenta los resultados de las pruebas estadísticas, el número de ejecuciones factibles y el promedio de los valores de aptitud obtenidos, el *ABCHS* obtuvo el mejor rendimiento general en este conjunto de funciones, seguido de los métodos *Reflection* y *Evolutionary* que superaron al esquema adaptativo en una función (C02), lo empataron en siete funciones y perdieron en dos. El método *Reflection* fue ligeramente mejor que *Evolutionary* debido al número de ejecuciones factibles y al número de funciones con los mejores valores de aptitud.

Los métodos *Centroid*, *Wrapping* y *Random* también obtuvieron un buen rendimiento, teniendo en cuenta las pruebas estadísticas, estos métodos sólo perdieron frente al esquema adaptativo en dos funciones y lo empataron en las ocho funciones restantes.

En conclusión, en el conjunto de funciones del CEC-2006 el método con el mejor rendimiento fue el *ABCHS*, seguido de los métodos *Reflection* y *Evolutionary*. Cabe aclarar que no se encontraron grandes diferencias en el número de ejecuciones factibles o en la cantidad de funciones con diferencias estadísticamente significativas.

Las Tablas 7.15 y 7.16 muestran el promedio de los valores de aptitud obtenidos en 25 ejecuciones independientes para cada uno de los problemas del CEC-2010 de 10 dimensiones.

La Tabla 7.15 muestra los resultados de 9 funciones en las cuales fue más fácil llegar a la región factible, es decir, en estos problemas todos los métodos encontraron soluciones factibles en al menos 19 (más del 75 %) de las 25 ejecuciones independientes.

En este conjunto de funciones, los métodos ABCHS, *Centroid* y *Evolutionary* obtuvieron los mejores resultados. De acuerdo con las pruebas estadísticas, el método *Centroid* obtuvo resultados

Tabla 7.15: Resultados obtenidos por cada uno de BCHM comparados, dentro del algoritmo clásico de Evolución Diferencial para los problemas del CEC-2010 en 10 dimensiones. En esta tabla sólo se presentan los resultados de los problemas en los cuales todos los métodos encontraron soluciones factibles en al menos 19 de las 25 ejecuciones independientes. El símbolo † significa que existe *SSD* a favor del método base (ABCHS) y ∗ representa *SSD* en contra del método base.

Problema	ABCHS	Res&Ran	Centroid	Evolutionary	Reflection	Random	Wrapping	Boundary
C01	-7.29E-01	-7.22E-01	-7.15E-01	**-7.44E-01***	-7.43E-01	-7.43E-01	-7.15E-01	-7.29E-01
C03	**0.00E+00**	**0.00E+00**	7.96E-20	**0.00E+00**	**0.00E+00**	**0.00E+00**	5.80E-22	5.76E+13(24)†
C04	2.71E-02	**-1.00E-05**	6.67E-02	**-1.00E-05**	3.96E-02	5.43E-02	-1.00E-05	7.91E-02
C07	**0.00E+00**	**0.00E+00**	1.59E-01	1.59E-01	**0.00E+00**	**0.00E+00**	4.78E-01	**0.00E+00**
C08	7.24E+00	8.66E+00	6.97E+00	1.48E+01	7.24E+00	**5.41E+00**	6.94E+00	7.40E+00
C11	-1.52E-03	-1.52E-03(24)	-1.52E-03	-1.52E-03(24)	-1.52E-03(24)	-1.52E-03	**-1.52E-03**	-1.52E-03(24)
C12	-2.83E+01	-1.56E+01	-2.81E+01	-3.68E+01	-3.01E+01	-2.04E+01	-1.63E+01	**-4.12E+01**
C13	-6.50E+01	-6.40E+01	-6.45E+01	-6.44E+01	-6.49E+01	-6.45E+01	**-6.54E+01**	-6.50E+01
C14	1.19E+12	1.20E+13†	**6.94E+11**	7.14E+12†	1.66E+13†	8.55E+12†	1.66E+13†	2.67E+13†
w/t/l	**-**	**0/8/1**	**0/9/0**	**1/7/1**	**0/8/1**	**0/8/1**	**0/8/1**	**0/7/2**

Tabla 7.16: Resultados obtenidos por cada uno de BCHM comparados, dentro del algoritmo clásico de Evolución Diferencial para los problemas del CEC-2010 en 10 dimensiones. En esta tabla sólo se presentan los resultados de los problemas en los cuales al menos un método encontró soluciones factibles en menos de 19 de las 25 ejecuciones independientes. El símbolo † significa que existe *SSD* a favor del método base (ABCHS).

Problema	ABCHS	Res&Ran	Centroid	Evolutionary	Reflection	Random	Wrapping	Boundary
C02	3.10E+00**(7)**	3.37E+00**(7)**	3.26E+00**(7)**	3.65E+00(3)	4.54E+00(2)	3.88E+00(3)	3.55E+00(4)	**3.07E+00**(3)
C05	2.86E+02**(13)**	**2.68E+02**(9)	3.63E+02(4)	4.25E+02(7)†	3.54E+02(3)	4.23E+02 (6)†	4.40E+02 (4)†	5.59E+02(2)†
C06	**3.40E+02(19)**	3.68E+02(14)	3.79E+02(14)	4.00E+02(12)	4.58E+02(9)†	4.53E+02(8)†	4.46E+02(9)	-
C09	**7.77E+11(4)**	-	2.46E+12**(4)**	-	-	-	1.00E+13(3)†	-
C10	**2.15E+12**(4)	6.76E+12(3)	-	3.39E+12(4)	-	-	6.73E+12**(6)**	-
C15	3.92E+13(15)	6.50E+13(5)	**2.00E+13(19)**	1.62E+14(16)†	1.16E+14**(19)**†	8.85E+13**(19)**†	1.35E+14(18)†	3.65E+14(11)†
C16	**9.90E-01(22)**	1.01E+00(20)	1.00E+00(14)	1.03E+00(16)†	1.03E+00(21)†	1.03E+00**(22)**†	1.04E+00(21)†	1.05E+00(15)†
C17	1.73E+02**(13)**	2.88E+02**(13)**	**1.20E+02**(4)	4.66E+02(7)†	6.15E+02(6)†	3.94E+02(8)†	5.26E+02(6)†	-
C18	4.82E+03**(8)**	1.00E+04**(8)**	**2.66E+03**(4)	1.17E+04(2)†	1.24E+04(2)†	1.34E+04(5)†	1.26E+04(5)†	3.87E+04(2)†
w/t/l	**-**	**0/8/0**	**0/8/0**	**0/3/5**	**0/2/5**	**0/1/6**	**0/3/6**	**0/1/4**

similares a los obtenidos por el ABCHS en todas las funciones, así como la misma cantidad de ejecuciones factibles. El método *Evolutionary* obtuvo un mejor rendimiento que el ABCHS en una función, pero también perdió en una función y empató en las restantes siete funciones.

Los métodos *Random*, *Wrapping*, *Res&Ran* y *Reflection* también mostraron un buen rendimiento, pues obtuvieron resultados similares a los del ABCHS en ocho funciones y perdieron en una función.

La Tabla 7.16 presenta los resultados de las nueve funciones restantes en las que al menos uno de los métodos encontró soluciones factibles en menos de 19 de las 25 ejecuciones independientes, es decir, en estas funciones fue más difícil alcanzar la región factible.

En este conjunto de funciones, sólo los métodos ABCHS y *Wrapping* encontraron soluciones factibles en todas las funciones, seguido por los métodos *Centroid*, *Res&Ran* y *Evolutionary*, los cuales no encontraron soluciones factibles en una sola función. Los métodos con la mayor cantidad de ejecuciones factibles fueron ABCHS, *Centroid* y *Res&Ran*. Con referencia a las pruebas estadísticas, el ABCHS obtuvo el mejor rendimiento, seguido de los métodos *Centroid* y *Res&Ran*, que lo empató en ocho funciones.

En conclusión, en el conjunto de funciones del CEC-2010 en 10 dimensiones, el método con el mejor rendimiento fue ABCHS, seguido por *Centroid* y *Res&Ran*. La principal fortaleza del ABCHS en este conjunto de funciones se presentó en las funciones en las cuales fue más difícil llegar a la región factible, en estos problemas el esquema adaptativo obtuvo las soluciones de mejor calidad y el mayor número de ejecuciones factibles.

Las Tablas 7.17 y 7.18 muestran el promedio de los valores de aptitud obtenidos en 25 ejecuciones independientes para cada una de las funciones de referencia del CEC-2010 en 30 dimensiones.

La Tabla 7.17 muestra los resultados de ocho funciones en las que todos los métodos encontraron

soluciones factibles en al menos 19 ejecuciones independientes.

En este conjunto de funciones, el ABCHS obtuvo la mayor cantidad de ejecuciones factibles y las soluciones de mayor calidad, seguido por los métodos *Wrapping*, *Evolutionary* y *Random*. El método *Wrapping* obtuvo soluciones de calidad similar a las obtenidas por el ABCHS en siete funciones y perdió en una; por otro lado, los métodos *Evolutionary* y *Random* empataron al ABCHS en cinco funciones, perdieron en dos funciones y obtuvieron soluciones de mejor calidad que el ABCHS en una función.

La Tabla 7.18 presenta los resultados de las 10 funciones restantes en las que fue más difícil llegar

Tabla 7.17: Resultados obtenidos por cada BCHM con Evolución Diferencial para los problemas del CEC-2010 en 30D en los cuales todos los métodos encontraron soluciones factibles en al menos 19 de las 25 ejecuciones independientes. El símbolo † significa que existe *SSD* a favor del método base (ABCHS) y ∗ representa *SSD* en contra del método base.

Problema	ABCHS	Res&Ran	Centroid	Evolutionary	Reflection	Random	Wrapping	Boundary
C01	-6.35E-01	-5.66E-01†	-4.69E-01†	-7.64E-01*	**-7.70E-01***	-7.12E-01*	-6.76E-01	-7.04E-01*
C03	**9.03E+12**	2.09E+13†	2.13E+13†	2.25E+13	4.18E+13†	1.30E+13	1.30E+13	2.36E+14(24)†
C07	3.19E-01	4.78E-01	4.78E-01	4.78E-01	**3.19E-01**	7.97E-01	1.12E+00	3.19E-01
C08	2.27E+02	7.90E+01	**7.11E+01**	8.29E+01	1.36E+02	8.21E+01	9.06E+01	1.78E+02
C12	4.54E-01	-2.60E-02(24)	7.38E-02(20)	4.15E-02(22)	-7.21E-02(23)	2.48E-01(22)	2.18E+00(21)	**-1.12E-01**(21)
C13	-6.17E+01	-6.15E+01	-6.11E+01	-6.15E+01	**-6.19E+01**	-6.14E+01	-6.16E+01	-6.15E+01
C14	4.48E+05	1.36E+06	3.04E+05	1.13E+06†	1.66E+06†	1.50E+06†	**2.85E+02**	8.72E+07†
C15	2.50E+13	6.85E+13†	**2.12E+13**	1.48E+14†	1.61E+14†	1.35E+14†	1.60E+14†	5.09E+14†
w/t/l	**-**	**0/5/3**	**0/6/2**	**1/5/2**	**1/4/3**	**1/5/2**	**0/7/1**	**1/4/3**

Tabla 7.18: Resultados obtenidos por cada BCHM con Evolución Diferencial para los problemas del CEC-2010 en 30D en los cuales al menos uno de los métodos obtuvo resultados factibles en menos 19 de las 25 ejecuciones independientes. El símbolo † significa que existe *SSD* a favor del método base (ABCHS).

Problema	ABCHS	Res&Ran	Centroid	Evolutionary	Reflection	Random	Wrapping	Boundary
C02	4.43E+00(17)	4.35E+00(13)	**3.88E+00(18)**	4.24E+00(14)	4.50E+00(16)	4.47E+00(11)	4.57E+00(16)	4.92E+00(8)
C04	2.51E-01(**7**)	1.60E-01(6)	1.99E-01(4)	2.24E-01(**7**)	6.66E+00(6)†	**3.69E-02**(5)	5.05E-02(5)	2.89E+00(**7**)
C05	**3.40E+02**(14)	4.19E+02(**18**)†	3.56E+02(9)	4.86E+02(9)†	5.28E+02(6)†	5.19E+02(9)†	5.34E+02(7)†	5.64E+02(10)†
C06	**4.49E+02(21)**	4.85E+02(**21**)	4.66E+02(19)	5.59E+02(11)†	5.04E+02(10)	5.54E+02(9)†	5.32E+02(10)†	6.07E+02(6)†
C09	7.10E+12(3)	1.30E+13(**5**)	**6.93E+12**(3)	-	-	2.29E+13(2)	4.58E+13(2)	9.19E+13(2)
C10	**5.29E+12**(3)	7.61E+12(**5**)	-	-	-	-	3.59E+13(2)	-
C11	-3.92E-04(21)	-3.92E-04(20)	-3.92E-04(20)	-3.92E-04(18)	-3.92E-04(**22**)	**-3.92E-04**(20)	-3.92E-04(19)	-3.92E-04(**22**)
C16	**1.04E+00**(18)	1.07E+00(**23**)†	1.04E+00(11)	1.10E+00(19)†	1.13E+00(22)†	1.12E+00(20)†	1.14E+00(19)†	1.13E+00(19)†
C17	**6.03E+02(15)**	8.03E+02(14)	6.66E+02(12)	1.87E+03(7)†	1.38E+03(6)†	1.47E+03(7)†	1.23E+03(3)†	2.92E+03(4)†
C18	1.60E+04(21)	2.08E+04(13)	**1.58E+04(23)**	4.06E+04(18)†	3.96E+04(20)†	3.03E+04(19)†	4.18E+04(20)†	6.46E+04(13)†
w/t/l	**-**	**0/8/2**	**0/9/0**	**0/3/5**	**0/3/5**	**0/4/5**	**0/5/5**	**0/4/5**

a la región factible, en estas funciones, al menos uno de los métodos encontró soluciones factibles en menos de 19 de las 25 ejecuciones independientes.

En este conjunto de funciones, sólo los métodos ABCHS, *Res&Ran* y *Wrapping* encontraron soluciones factibles en todas las funciones. Los métodos con la mayor cantidad de ejecuciones factibles fueron ABCHS, *Res&Ran* y *Centroid*. Con respecto a las pruebas estadísticas, el ABCHS obtuvo el mejor desempeño, seguido por el método *Centroid* que obtuvo resultados de calidad similar a los obtenidos por el ABCHS en nueve funciones, sin embargo, no obtuvo soluciones factibles en una de las 10 funciones; en orden de desempeño siguió el método *Res&Ran*, que obtuvo soluciones de calidad similar a las obtenidas por el ABCHS en ocho funciones pero perdió en dos funciones.
Como conclusión, en las funciones de alta dimensionalidad (30 dimensiones) del conjunto de funciones de referencia del CEC-2010, el método con mejor rendimiento fue el ABCHS, seguido de los métodos *Centroid* y *Res&Ran*. La principal fortaleza de ABCHS en este conjunto de funciones se presentó nuevamente en las funciones en las que fue más difícil llegar a la región factible.
La Figura 7.3 muestra dos gráficos que resumen el rendimiento de los ocho métodos comparados dentro del algoritmo clásico de Evolución Diferencial en este experimento.

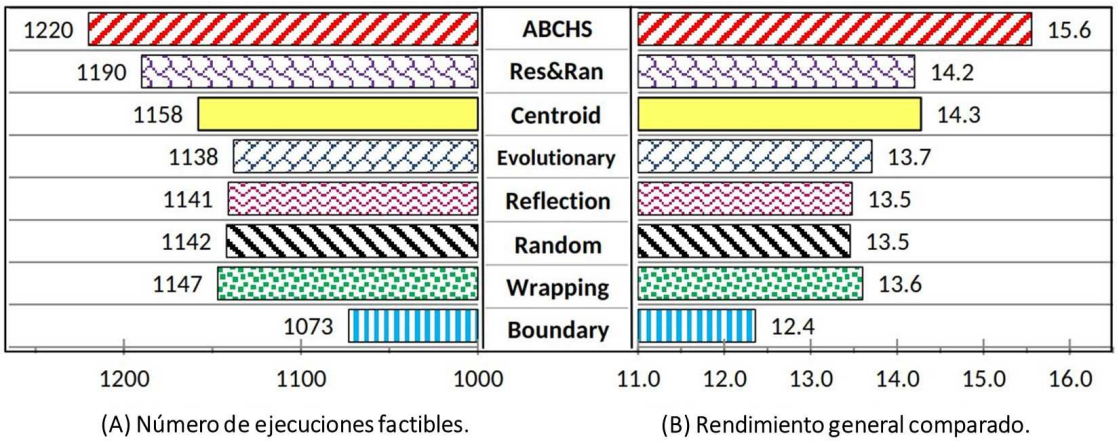


Figura 7.3: Resumen gráfico de los resultados experimentales obtenidos al comparar el rendimiento del *ABCHS* contra siete métodos para el manejo de restricciones de límite dentro del algoritmo clásico de DE.

En la Figura 7.3 (A) se puede apreciar que el método con la mayor cantidad de ejecuciones factibles fue el ABCHS, seguido de *Res&Ran* y *Centroid*. En un segundo bloque se ubican los métodos comúnmente utilizados: *Wrapping*, *Random*, *Reflection* y *Evolutionary*, que obtuvieron una cantidad similar de ejecuciones factibles entre ellos. Finalmente, el método con el menor número de ejecuciones factibles fue Boundary.
Este gráfico resalta el hecho de que cuando se utilizó el ABCHS, se obtuvieron un 6.3 % más de

ejecuciones factibles de las que se obtuvieron al emplear cualquiera de los métodos comúnmente utilizados en el manejo de restricciones de límite.

La Figura 7.3 (B) muestra el valor del rendimiento general comparado. Este parámetro indica la calidad de las soluciones factibles encontradas por cada método. De lo observado en el gráfico se muestra nuevamente que el ABCHS obtuvo el rendimiento general comparado más alto, seguido de los métodos *Centroid* y *Res&Ran*. En un segundo bloque se colocan los métodos: *Evolutionary*, *Wrapping*, *Random* y *Reflection*, que obtuvieron un rendimiento similar, y finalmente, el método Boundary obtuvo el peor rendimiento.

La Tabla 7.19 muestra el tiempo de procesamiento empleado por cada uno de los BCHM compara-

Tabla 7.19: Tiempo de procesamiento empleado por cada uno de los BCHM comparados dentro del algoritmo de DE clásico al realizar 10,000 evaluaciones.

BCHM	**T2**	*(T2-T1)/T1*
Centroid	**3.51**	**3.33**
Evolutionary	3.66	3.51
Wrapping	3.73	3.60
Random	3.75	3.63
Boundary	3.77	3.65
Reflection	3.77	1.66
ABCHS	4.46	4.51
Res&Ran	5.38	5.63

dos dentro del algoritmo de DE clásico al realizar 10,000 evaluaciones. Los métodos se presentan ordenados de menor a mayor consumo de tiempo, por tanto los métodos ubicados en las filas superiores son los más rápidos. El valor de $T1 = 0.81$ fue el mismo para todos los BCHM.

Como se observa en la tabla, el método con el menor consumo de tiempo es *Centroid*, seguido por los métodos *Evolutionary* y *Wrapping*. El *ABCHS* por su parte se encuentra en la penúltima fila, siendo sólo más rápido que el método *Res&Ran*, mismo que en este experimento fue empleado como método base del ABCHS.

Como se mencionó en la Subsección 7.1.1, el tiempo de procesamiento del *ABCHS* puede verse muy afectado por el tiempo de procesamiento de su método base ya que es éste el que se emplea en la primera fase del proceso evolutivo, cuando se realiza la mayor cantidad de reparaciones.

En este experimento se utilizó como método base a *Res&Ran*, que también ha sido propuesto en el presente trabajo como una mejora del método Resampling, haciéndolo más rápido y sin que se cicle por realizar un número infinito de re-muestreos.

Aún con las mejoras mencionadas, el método *Res&Ran* sigue siendo un poco lento y al ser empleado como método base, afecta el tiempo de procesamiento del ABCHS. Sin embargo, la diferencia

en el tiempo de procesamiento entre el método más rápido (*Centroid*) y el *ABCHS* es menor a un segundo en 10,000 evaluaciones. Por último, se observa nuevamente que el tiempo de procesamiento del ABCHS es un poco mejor al de su método base.

Conclusiones del presente experimento:

En el presente experimento se encontró que el método con el mejor rendimiento, tanto en la cantidad como en la calidad de las ejecuciones factibles fue el *ABCHS*, seguido de los métodos: *Centroid* y *Res&Ran*. *Centroid* obtuvo soluciones de mejor calidad que *Res&Ran*, mientras que *Res&Ran* obtuvo una mayor cantidad de ejecuciones factibles que *Centroid*.

Las fortalezas del *ABCHS* se presentaron principalmente en el conjunto de funciones del CEC-2010 en 30 dimensiones (funciones de alta dimensionalidad), y más precisamente en aquellas funciones en las que fue más difícil llegar a la región factible.

7.2.2. Experimento 4. Comparativo del rendimiento del ABCHS dentro del algoritmo ICDE

El objetivo de este experimento fue observar el comportamiento del *ABCHS* en un algoritmo del estado del arte, basado en DE, para problemas de optimización numérica restringida (*CNOP*). En este experimento se utilizó el algoritmo mejorado para Evolución Diferencial Restringida (*improved ($\mu + \lambda$)-constrained differential evolution, ICDE*), propuesto por G. Jia et al. [23]. En este algoritmo cada individuo produce tres descendientes mediante el uso de tres estrategias de mutación y el cruce binomial para generar la población descendiente.

Las Tablas 7.20 y 7.21 muestran los resultados obtenidos por cada uno de los métodos comparados, en los problemas del CEC-2006; la función g20 se excluyó de este experimento ya que ninguno de los métodos fue capaz de obtener soluciones factibles.

La Tabla 7.20 muestra los resultados de 15 funciones en las que las pruebas estadísticas no encontraron diferencias estadísticamente significativas entre los métodos comparados y tampoco se encontraron diferencias en el número de ejecuciones factibles. La única distinción que se encontró entre los métodos fue en la cantidad de funciones en las que se encontraron los mejores promedios de aptitud. En este sentido, el esquema adaptativo (ABCHS) encontró los mejores promedios en 15 funciones, seguido de los métodos *Res&Ran* y *Evolutionary*, que encontraron los mejores promedios en 13 funciones y finalmente, el resto de los métodos encontraron el mejor promedio en 12 funciones.

La Tabla 7.21 muestra los resultados de las ocho funciones restantes, en las cuales se encontraron diferencias tanto en el número de ejecuciones factibles como en las pruebas estadísticas.

En este subconjunto de funciones se puede ver que el ABCHS obtuvo el número máximo de eje-

Tabla 7.20: Resultados obtenidos por cada BCHM con ICDE para los problemas del CEC-2006 en los cuales no se encontraron diferencias estadísticamente significativas.

Problema	ABCHS	Res&Ran	Centroid	Evolutionary	Reflection	Random	Wrapping	Boundary
g03	**-1.00E+00**	**-1.00E+00**	-1.00E+00	**-1.00E+00**	-1.00E+00	-1.00E+00	-1.00E+00	-1.00E+00
g04	**-3.07E+04**	**-3.07E+04**	**-3.07E+04**	**-3.07E+04**	**-3.07E+04**	**-3.07E+04**	**-3.07E+04**	**-3.07E+04**
g05	**5.13E+03**	**5.13E+03**	**5.13E+03**	**5.13E+03**	**5.13E+03**	**5.13E+03**	**5.13E+03**	**5.13E+03**
g06	**-6.96E+03**	**-6.96E+03**	**-6.96E+03**	**-6.96E+03**	**-6.96E+03**	**-6.96E+03**	**-6.96E+03**	**-6.96E+03**
g08	**-9.58E-02**	**-9.58E-02**	**-9.58E-02**	**-9.58E-02**	**-9.58E-02**	**-9.58E-02**	**-9.58E-02**	**-9.58E-02**
g09	**6.81E+02**	**6.81E+02**	**6.81E+02**	**6.81E+02**	**6.81E+02**	**6.81E+02**	**6.81E+02**	**6.81E+02**
g10	**7.05E+03**	7.05E+03	7.05E+03	7.05E+03	7.05E+03	7.05E+03	7.05E+03	7.05E+03
g12	**-1.00E+00**	**-1.00E+00**	**-1.00E+00**	**-1.00E+00**	**-1.00E+00**	**-1.00E+00**	**-1.00E+00**	**-1.00E+00**
g13	**5.39E-02**	6.93E-02	2.23E-01	6.93E-02	5.39E-02	1.00E-01	6.93E-02	5.39E-02
g14	**-4.78E+01**	**-4.78E+01**	**-4.78E+01**	**-4.78E+01**	**-4.78E+01**	**-4.78E+01**	**-4.78E+01**	**-4.78E+01**
g15	**9.62E+02**	**9.62E+02**	**9.62E+02**	**9.62E+02**	**9.62E+02**	**9.62E+02**	**9.62E+02**	**9.62E+02**
g16	**-1.91E+00**	**-1.91E+00**	**-1.91E+00**	**-1.91E+00**	**-1.91E+00**	**-1.91E+00**	**-1.91E+00**	**-1.91E+00**
g17	**8.85E+03**	**8.85E+03**	**8.85E+03**	**8.85E+03**	**8.85E+03**	**8.85E+03**	**8.85E+03**	**8.85E+03**
g18	**-8.66E-01**	**-8.66E-01**	**-8.66E-01**	**-8.66E-01**	**-8.66E-01**	**-8.66E-01**	**-8.66E-01**	**-8.66E-01**
g24	**-5.51E+00**	**-5.51E+00**	**-5.51E+00**	**-5.51E+00**	**-5.51E+00**	**-5.51E+00**	**-5.51E+00**	**-5.51E+00**
w/t/l	**-**	**0/15/0**	**0/15/0**	**0/15/0**	**0/15/0**	**0/15/0**	**0/15/0**	**0/15/0**

Tabla 7.21: Resultados obtenidos por cada BCHM con ICDE para los problemas del CEC-2006 en los cuales se encontraron diferencias estadísticamente significativas. El símbolo † significa que existe *SSD* a favor del método base (ABCHS).

Problema	ABCHS	Res&Ran	Centroid	Evolutionary	Reflection	Random	Wrapping	Boundary
g01	**-1.50E+01**	**-1.50E+01**	-1.50E+01†	**-1.50E+01**	**-1.50E+01**	-1.50E+01†	-1.50E+01†	**-1.50E+01**
g02	-7.96E-01	-7.99E-01	-7.95E-01†	-8.02E-01	**-8.03E-01**	-7.86E-01†	-7.72E-01†	-7.62E-01†
g07	**2.43E+01**	**2.43E+01**	**2.43E+01**	2.44E+01†	2.44E+01†	**2.43E+01**	**2.43E+01**	**2.43E+01**
g11	**7.50E-01**	7.50E-01	**7.50E-01**	7.65E-01	**7.50E-01**	**7.50E-01**	**7.50E-01**	9.58E-01†
g19	3.27E+01	3.27E+01	3.27E+01†	3.27E+01	3.27E+01†	3.27E+01†	3.27E+01†	**3.27E+01**
g21	1.94E+02	1.94E+02	1.94E+02†	2.36E+02	**1.94E+02**	1.94E+02†	1.94E+02†	2.72E+02†
g22	2.86E+02	2.93E+02(19)	2.72E+02(23)	2.55E+02(14)	2.64E+02(23)	5.99E+02(18)†	4.45E+02(21)†	**2.52E+02**(9)
g23	-4.00E+02	-3.98E+02†	-4.00E+02	-3.76E+02†	-4.00E+02	**-4.00E+02**	-4.00E+02	6.80E+01†
w/t/l	**-**	**0/7/1**	**0/4/4**	**0/6/2**	**0/6/2**	**0/3/5**	**0/3/5**	**0/4/4**

cuciones factibles y los mejores promedios en tres funciones, finalmente, considerando las pruebas estadísticas, el ABCHS también obtuvo el mejor rendimiento, seguido del método *Res&Ran*.

Teniendo en cuenta las pruebas estadísticas, el método *Res&Ran* obtuvo un menor rendimiento que el esquema adaptativo en una función, mientras que en las restantes siete funciones su rendimiento fue similar. Sin embargo, *Res&Ran* obtuvo una menor cantidad de ejecuciones factibles y sólo obtuvo el mejor promedio de aptitud en dos funciones.

Después del método *Res&Ran*, en orden de rendimiento, siguieron los métodos *Reflection* y *Evolutionary*, que obtuvieron un rendimiento similar entre ellos, ambos empataron al esquema adaptativo

en seis funciones y perdieron en dos. Sin embargo, el método *Reflection* fue ligeramente mejor que *Evolutionary*, ya que *Reflection* obtuvo una mayor cantidad de ejecuciones factibles.

Como resumen, de los resultados obtenidos en el conjunto de problemas del CEC-2006, observamos que el mejor rendimiento fue el obtenido por el ABCHS, seguido por los métodos *Res&Ran* y *Reflection*. También se observó que el esquema adaptativo mostró superioridad principalmente en las funciones de la Tabla 7.21, en los cuales fue más difícil encontrar soluciones factibles de calidad debido a que fueron problemas con alta dimensionalidad, con un mayor número de restricciones o a que representaban problemas en los cuales la proporción entre el tamaño de región factible y el del espacio de búsqueda es muy pequeña.

Las Tablas 7.22 y 7.23 muestran el promedio del valor de aptitud obtenido en 25 ejecuciones independientes para cada una de las 18 funciones del CEC-2010 en 10 dimensiones.

En la Tabla 7.22 se muestran los resultados obtenidos por cada uno de los ocho métodos com-

Tabla 7.22: Resultados obtenidos por cada BCHM con ICDE para los problemas del CEC-2010 en 10D donde todos los métodos encontraron soluciones factibles en todas las 25 ejecuciones independientes. El símbolo † significa que existe *SSD* a favor del método base (ABCHS).

Problema	ABCHS	Res&Ran	Centroid	Evolutionary	Reflection	Random	Wrapping	Boundary
C01	**-7.47E-01**	**-7.47E-01**	-7.46E-01†	-7.46E-01†	-7.47E-01†	**-7.47E-01**	**-7.47E-01**	-7.47E-01†
C03	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
C04	-1.00E-05	-1.00E-05	-1.00E-05	-1.00E-05	-1.00E-05	**-1.00E-05**	-1.00E-05	-1.00E-05
C07	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
C08	1.07E+01	1.02E+01	1.07E+01	9.07E+00	9.46E+00	1.02E+01	**8.94E+00**	9.51E+00
C12	-5.33E+00	-1.99E-01	**-5.76E+00**	-6.98E-01	-1.70E+00	-1.99E+00	-2.19E+00	-2.19E+00
C13	-6.40E+01	-6.37E+01	**-6.48E+01**	-6.30E+01	-6.27E+01	-6.43E+01	-6.44E+01	-6.34E+01
C14	**6.30E+08**	4.72E+10†	2.10E+09†	9.94E+10†	1.19E+11†	7.42E+10†	1.36E+11†	1.57E+11†
C15	**4.55E+11**	3.79E+12†	9.50E+11	9.52E+12†	2.16E+13†	7.35E+12†	1.32E+13†	7.16E+13†
w/t/l	**-**	**0/7/2**	**0/7/2**	**0/6/3**	**0/6/3**	**0/7/2**	**0/7/2**	**0/6/3**

parados, en los problemas en los cuales fue más fácil llegar a la región factible, es decir, en estos problemas todos los métodos obtuvieron el máximo número de ejecuciones factibles.

En este subconjunto de funciones se observó que el ABCHS obtuvo los mejores promedios de aptitud en cinco funciones y, según las pruebas estadísticas, también obtuvo el mejor rendimiento, seguido de los métodos: *Centroid*, *Random*, *Wrapping* y *Res&Ran*, de los cuales, los tres primeros obtuvieron los mejores promedios de aptitud en cuatro funciones, mientras que el método *Res&Ran* sólo obtuvo los mejores promedios de aptitud en tres funciones.

La Tabla 7.23 presenta los resultados de nueve funciones en las cuales al menos uno de los métodos no obtuvo soluciones factibles en todas las 25 ejecuciones independientes.

En este subconjunto de funciones, sólo los métodos: ABCHS, *Res&Ran*, *Centroid* y *Boundary* encontraron soluciones factibles en las nueve funciones, los métodos: *Random*, *Wrapping* y *Reflection*

Tabla 7.23: Resultados obtenidos por cada BCHM con ICDE para los problemas del CEC-2010 en 10D donde al menos uno de los métodos no encontró soluciones factibles en las 25 ejecuciones independientes. El símbolo † significa que existe *SSD* a favor del método base (ABCHS) y ∗ representa *SSD* en contra del método base.

Problema	ABCHS	Res&Ran	Centroid	Evolutionary	Reflection	Random	Wrapping	Boundary
C02	3.33E+00	2.85E+00	**2.32E+00**(13)*	3.48E+00(12)	3.54E+00(15)	3.30E+00(14)	3.22E+00(19)	3.14E+00(11)
C05	3.38E+02(**21**)	**2.95E+02**(20)	3.67E+02(1)	-	3.27E+02(2)	-	-	5.41E+02(3)†
C06	3.33E+02(**19**)	2.89E+02(**19**)	**1.97E+02**(2)	-	5.11E+02(1)	5.01E+02(1)	4.42E+02(3)	4.10E+02(12)
C09	6.47E+12(**24**)	5.41E+12(23)	**5.07E+12**(6)	6.13E+12(4)	1.56E+13(3)†	1.05E+13(2)	1.13E+13(3)	2.14E+13(8)†
C10	7.04E+12(**23**)	7.62E+12(**23**)	**1.52E+12**(2)	-	-	8.03E+12(3)	1.79E+13(3)	3.04E+13(4)†
C11	-1.52E-03(10)	-1.52E-03(10)	-1.52E-03(11)	**-1.52E-03**(12)	-1.52E-03(8)	-1.52E-03(11)	-1.52E-03(**14**)	-1.52E-03(9)
C16	**8.83E-01**(**19**)	9.33E-01(**19**)	1.01E+00(8)	1.06E+00(3)	8.93E-01(3)	1.03E+00(3)	1.03E+00(4)	1.04E+00(4)
C17	**2.43E+02**(**24**)	3.45E+02(23)	3.95E+02(17)†	5.63E+02(13)†	5.95E+02(13)†	3.50E+02(9)	5.92E+02(9)†	8.38E+02(14)†
C18	5.37E+03(24)	5.35E+03	**3.36E+03**(21)	9.35E+03(18)†	1.02E+04(20)†	9.28E+03(18)†	1.13E+04(20)†	1.83E+04(18)†
w/t/l	**-**	**0/9/0**	**1/7/1**	**0/4/2**	**0/5/3**	**0/7/1**	**0/6/2**	**0/4/5**

encontraron soluciones factibles en ocho funciones y el método *Evolutionary* encontró soluciones factibles en sólo seis de las nueve funciones.

Los métodos con la mayor cantidad de ejecuciones factibles fueron: ABCHS con 189, *Res&Ran* con 187, *Boundary* con 83 y *Centroid* con 81.

En cuanto a los mejores promedios de aptitud, el método *Centroid* encontró los mejores valores de aptitud en cinco funciones, seguido del ABCHS con dos funciones y los métodos *Res&Ran* y *Evolutionary* encontraron el mejor valor de aptitud en una función.

Teniendo en cuenta las pruebas estadísticas, el ABCHS y *Res&Ran* obtuvieron el mejor rendimiento, seguido por *Centroid* que obtuvo un mejor rendimiento que el ABCHS en una función, perdió en una y lo empató en las restantes siete funciones. El método *Random* también obtuvo un rendimiento similar al del ABCHS en siete funciones y también perdió en una función, seguido por el método *Wrapping*, mismo que empató al ABCHS en seis funciones y perdió en dos.

En resumen, el ABCHS mostró el mejor rendimiento general en el conjunto de funciones de 10 dimensiones del CEC-2010, seguido de los métodos *Res&Ran* y *Centroid*.

El esquema de adaptativo propuesto (ABCHS) mostró un mejor rendimiento en al menos 50 % de las funciones probadas en este experimento, principalmente en aquellas funciones en las que fue más difícil llegar a la región factible y en las funciones no separables.

Por otro lado, el método *Res&Ran* obtuvo una mayor cantidad de soluciones factibles que *Centroid*, lo que demuestra su capacidad para llegar a la región factible, mientras que *Centroid* obtuvo soluciones de mejor calidad que el método *Res&Ran*.

Las Tablas 7.24 y 7.25 muestran los resultados obtenidos en las 18 funciones de conjunto de referencia del CEC-2010 en 30 dimensiones. La Tabla 7.24 muestra los resultados de ocho funciones en las que fue más fácil llegar a la región factible y, por lo tanto, todos los métodos encontraron

soluciones factibles para las ocho funciones en cada una de las 25 ejecuciones independientes.
En la Tabla 7.24, al considerar las pruebas estadísticas se observó que el ABCHS obtuvo el mejor rendimiento ya que los métodos más cercanos a él fueron *Centroid* y *Reflection*, que obtuvieron resultados similares a los obtenidos por el ABCHS en sólo tres funciones, mientras que en las cinco funciones restantes sus resultados fueron de menor calidad. En orden de desempeño, fueron seguidos por los métodos *Res&Ran*, *Evolutionary*, *Random* y *Wrapping*, ya que estos cuatro métodos obtuvieron un rendimiento similar al del ABCHS en dos funciones y perdieron en seis funciones.
En esta tabla también se observa que el ABCHS obtuvo los mejores promedios de aptitud en cinco

Tabla 7.24: Resultados obtenidos por cada BCHM con ICDE para los problemas del CEC-2010 en 30D donde todos los métodos encontraron soluciones factibles en todas las 25 ejecuciones independientes. El símbolo † significa que existe *SSD* a favor del método base (ABCHS).

Problema	ABCHS	Res&Ran	Centroid	Evolutionary	Reflection	Random	Wrapping	Boundary
C01	-8.07E-01	-7.94E-01†	-7.75E-01†	-8.12E-01	**-8.13E-01**	-7.51E-01†	-6.73E-01†	-7.58E-01†
C03	6.98E+12	**4.49E+12**	1.22E+13	5.10E+14†	2.66E+13	1.20E+13	8.46E+12	3.89E+14†
C07	**1.72E-12**	5.48E-09†	1.59E-01†	1.71E-05†	7.72E-07†	2.54E-06†	2.38E-04†	1.59E-01†
C08	**1.12E-06**	4.51E+00†	2.12E-02†	8.96E+00†	1.74E+00†	5.10E+00†	2.40E+00†	1.97E+01†
C13	**-6.19E+01**	-6.12E+01	-6.16E+01	-6.09E+01	-6.17E+01	-6.17E+01	-6.18E+01	-6.08E+01
C14	6.78E+05	1.21E+11†	**4.49E+05**	2.40E+12†	2.29E+12†	3.40E+12†	5.83E+12†	2.25E+12†
C15	**3.34E+11**	9.17E+12†	1.84E+12†	4.84E+13†	5.64E+13†	5.57E+13†	8.46E+13†	1.41E+14†
C18	**5.62E+03**	1.22E+04†	7.11E+03†	2.65E+04†	2.47E+04†	2.71E+04†	3.19E+04†	4.21E+04†
w/t/l	**-**	**0/2/6**	**0/3/5**	**0/2/6**	**0/3/5**	**0/2/6**	**0/2/6**	**0/1/7**

Tabla 7.25: Resultados obtenidos por cada BCHM con ICDE para los problemas del CEC-2010 en 30D donde al menos uno de los métodos no encontró soluciones factibles en las 25 ejecuciones independientes. El símbolo † significa que existe *SSD* a favor del método base (ABCHS) y ∗ representa *SSD* en contra del método base.

Problema	ABCHS	Res&Ran	Centroid	Evolutionary	Reflection	Random	Wrapping	Boundary
C02	3.35E+00	**2.99E+00**	3.77E+00(18)	4.34E+00(17)†	4.34E+00(12)†	4.27E+00(21)†	4.68E+00(12)†	4.61E+00(10)†
C04	6.75E-02(17)	7.98E-02(20)	1.74E-01(**24**)†	6.64E-02(21)	**5.26E-02**(17)	9.32E-02(19)	7.29E-02(18)	1.36E-01(16)
C05	**3.71E+02**(23)	4.00E+02(**24**)	-	-	4.22E+02(1)	-	-	5.93E+02(4)†
C06	**3.89E+02(22)**	3.94E+02(19)	-	-	5.05E+02(1)	-	5.99E+02(1)	-
C09	2.35E+13	2.48E+13	**9.90E+12**(2)*	5.01E+13(2)†	2.94E+13(1)	4.12E+13(2)†	4.32E+13(1)	1.09E+14(3)†
C10	2.14E+13	-	**1.41E+13**(3)	2.16E+13(1)	2.63E+13(1)	1.79E+13(1)	3.51E+13(3)	8.73E+13(2)†
C11	-3.84E-04	-3.84E-04(24)	**-3.92E-04***	-3.88E-04	-3.78E-04(24)	-3.81E-04(24)	-3.77E-04	-3.74E-04
C12	-1.88E-01(**24**)	-1.30E-01(23)	-1.81E-01(**24**)	-1.79E-01(22)	**-1.99E-01**(23)	-1.77E-01(22)	-1.23E-01(**24**)	-1.97E-01(23)
C16	1.07E+00(**24**)	1.07E+00(23)	**1.06E+00**(4)	-	1.15E+00(3)†	1.13E+00(1)	1.13E+00(4)†	1.14E+00(3)†
C17	7.51E+02	8.69E+02	**5.48E+02**(21)	1.96E+03(18)†	1.65E+03(11)†	1.67E+03(16)†	1.74E+03(17)†	2.73E+03(17)†
w/t/l	**-**	**0/9/0**	**2/5/1**	**0/4/3**	**0/7/3**	**0/5/3**	**0/6/3**	**0/3/6**

funciones, seguidos por los métodos *Centroid*, *Reflection* y *Res&Ran*, que obtuvieron los mejores promedios en una función.
La Tabla 7.25 presenta los resultados de las 10 funciones restantes, en las cuales al menos uno

de los métodos no encontró soluciones factibles en todas las 25 ejecuciones independientes de al menos una función, por lo tanto, en este conjunto de funciones, fue más difícil llegar a la región factible.

En esta tabla se puede ver que sólo los métodos ABCHS y *Reflection* pudieron encontrar soluciones factibles para todas las funciones. En un segundo grupo se observó que los métodos: *Res&Ran*, *Wrapping* y Boundary encontraron soluciones factibles para nueve funciones, seguido de los métodos: *Centroid* y *Random*, que encontraron soluciones factibles en ocho funciones y, finalmente, el método *Evolutionary* sólo encontró soluciones factibles para siete funciones.

Considerando el número de ejecuciones factibles, se observó que el ABCHS obtuvo el mayor número de ellas con 235, seguido de *Res&Ran* con 196, *Centroid* con 121, *Evolutionary* y *Random* obtuvieron 106 ejecuciones factibles cada uno, *Wrapping* 105, *Boundary* 103 y finalmente *Reflection* con 94.

En cuanto al número de funciones en las que se encontraron los mejores promedios de aptitud, *Centroid* se destaca con cinco, ABCHS y *Reflection* con dos y finalmente *Res&Ran* con una función.

Considerando las pruebas estadísticas, se observó que el método *Centroid* obtuvo resultados de mejor calidad que el ABCHS en dos funciones, mostró un rendimiento similar en cinco funciones y se perdió en una función, posteriormente el método *Res&Ran* obtuvo un rendimiento similar al del ABCHS en nueve funciones, seguido de *Reflection* que empató al ABCHS en siete funciones y perdió en tres, el método *Wrapping* empató al ABCHS en seis funciones y perdió en tres, para el resto de los métodos su rendimiento fue menor.

En este conjunto de funciones de alta dimensionalidad (30 dimensiones), se hizo evidente el buen desempeño del ABCHS sobre todos los demás métodos, mostrando un mejor rendimiento en al menos 72 % de las funciones probadas, principalmente en funciones no separables y en las funciones en las que fue más difícil llegar a la región factible.

Los métodos *Centroid* y *Res&Ran* también mostraron un buen rendimiento. *Centroid* demostró su capacidad para obtener soluciones de buena calidad y *Res&Ran* mostró su capacidad para llegar a la región factible.

La Figura 7.4 muestra dos gráficos que resumen el rendimiento de los ocho métodos comparados dentro del algoritmo ICDE. El gráfico del lado izquierdo, Figura 7.4 (A), muestra el número total de ejecuciones independientes en las que cada uno de los métodos comparados obtuvo soluciones factibles. Como se observa, el ABCHS obtuvo la mayor cantidad de ejecuciones factibles, seguido del método *Res&Ran*; la diferencia entre el número de ejecuciones factibles obtenidas por cualquiera de estos dos métodos y el resto de los métodos fue muy grande. En una tercera posición, se observa el método *Centroid*, mientras que el resto de los métodos, los métodos comúnmente

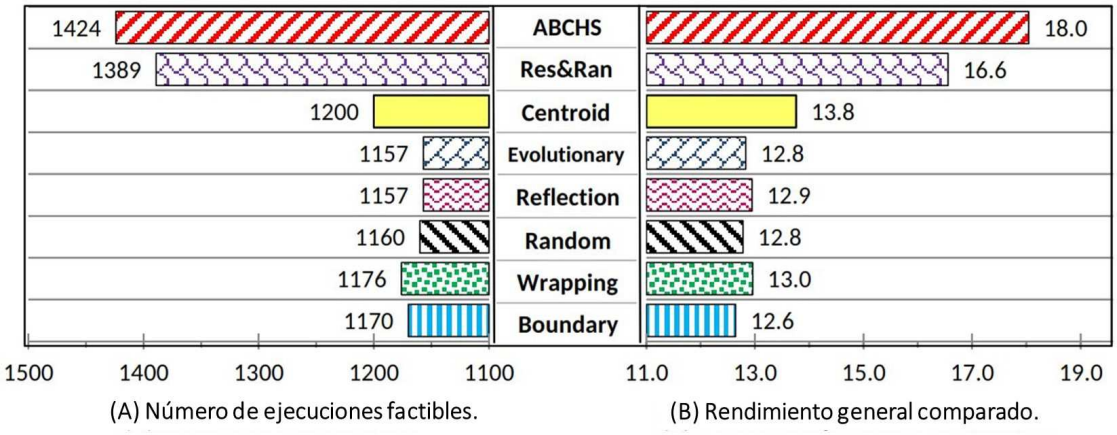


Figura 7.4: Resumen gráfico de los resultados experimentales obtenidos al comparar el rendimiento del *ABCHS* contra siete métodos para el manejo de restricciones de límite dentro del algoritmo ICDE.

utilizados en el manejo de restricciones de límite, obtuvieron un número menor de ejecuciones factibles. En este gráfico se resalta el hecho de que cuando se utilizó el ABCHS se obtuvo un 18 % más de ejecuciones factibles que cuando se utilizó cualquiera de los métodos comúnmente utilizados en el manejo de restricciones de límite.

En términos de la calidad de las soluciones encontradas, la figura 7.4 (B) muestra el valor del rendimiento general comparado (OCP), el comportamiento de este parámetro es muy similar al del lado izquierdo; el ABCHS obtuvo el rendimiento general comparado más alto, seguido de los métodos *Res&Ran* y *Centroid*, y al final, con una diferencia muy marcada, se observan los métodos restantes con un rendimiento inferior.

Conclusiones del presente experimento:

En el presente experimento, se comparó el rendimiento de ocho métodos para el manejo de restricciones de límite dentro del algoritmo *ICDE*. Se observó que el método con el mejor rendimiento fue el *ABCHS*, seguido por los métodos *Res&Ran* y *Centroid*, el resto de los métodos, los métodos utilizados habitualmente para el manejo de restricciones de límite mostraron un rendimiento deficiente.

El mejor rendimiento del *ABCHS* se presentó principalmente en funciones de dimensionalidad media a alta, en funciones no separables y en las funciones en las que fue más difícil llegar a la región factible. El método *Res&Ran* mostró una buena capacidad para alcanzar la región factible y el método *Centroid* mostró buena capacidad para obtener soluciones de buena calidad.

Capítulo 8

Conclusiones y trabajo futuro

En el presente trabajo, se propuso un esquema adaptativo para el manejo de restricciones de límite *ABCHS*, en algoritmos de optimización numérica restringida inspirados en la naturaleza.
El esquema adaptativo propuesto emplea un conjunto de métodos para el manejo de restricciones de límite *BCHMs*, los cuales son aplicados en dos etapas. En la primera etapa, cuando aún no existen soluciones factibles, se emplea un método que beneficia la exploración del espacio de búsqueda. En la segunda etapa se selecciona un método, a partir de un conjunto, según sus probabilidades que se actualizan cada cierto período de aprendizaje, de modo que aquellos métodos que generen los vectores reparados con el mejor valor de aptitud, tienen una mayor posibilidad de ser aplicados.

8.1. Observaciones sobre ABCHS

Se comparó el rendimiento del ABCHS contra el de varios métodos típicos para el manejo de restricciones de límite (BCHMs) dentro de los algoritmos de Evolución Diferencial (DE) y Optimización por Cúmulos de Partículas (PSO), empleando en ambos casos tanto la versión canónica del algoritmo, así como una versión del estado del arte, especializada en optimización restringida.
En la experimentación con el algoritmo DE, el ABCHS se comparó contra: *Res&Ran*, *Centroid*, *Evolutionary*, *Reflection*, *Random*, *Wrapping* y *Boundary*; siendo *Res&Ran* y *Centroid* dos propuestas secundarias del presente trabajo.
En la experimentación con el algoritmo PSO, el ABCHS se comparó contra los siguientes métodos híbridos: *Cen&DeB*, *Evo&DeB* , *Ref&DeB* , *Bou&DeB* , *Ran&RaB* y *Wra&RaB*; siendo el método *Cen&DeB* un híbrido de *Centroid* y *Random Back*.
En estos experimentos se observó que el ABCHS obtuvo un mayor rendimiento, de forma general, que los métodos con los que fue comparado, tanto en su capacidad para llegar a la región factible como en la calidad de las soluciones encontradas. La principal ventaja del esquema adaptativo

propuesto fue su excelente desempeño en funciones de mediana a alta dimensionalidad, así como en funciones con una mayor cantidad de restricciones y funciones que por su naturaleza mostraron una mayor dificultad para alcanzar la región factible.
Se observó también que la elección de uno u otro método para el manejo de restricciones de límite tiene un impacto considerable en el desempeño del algoritmo de optimización, de ahí la importancia de elegir correctamente este elemento u optar por un esquema adaptativo como el ABCHS presentado en este trabajo. El impacto de los BCHMs ha sido documentado también en investigaciones previas [2, 18, 24].

8.1.1. ABCHS con algoritmos canónicos

En los experimentos en los que se utilizó la versión canónica de los algoritmos PSO y DE, se observó que el ABCHS superó ampliamente al resto de los métodos y esta diferencia fue aún más notoria en el caso del algoritmo PSO. Los algoritmos canónicos no tienen mecanismos especializados para manejar problemas restringidos, por lo que cualquier apoyo proporcionado por ABCHS tiene un mayor impacto, por otro lado, el rendimiento del ABCHS se hizo más evidente dentro del algoritmo PSO debido a que un gran porcentaje de partículas tiende a abandonar el espacio de búsqueda en las primeras generaciones del proceso de búsqueda.
En estos experimentos también se comparó el tiempo de procesamiento por cada BCHM. En el algoritmo PSO el método más rápido fue *Centroid* y el más lento *Res&Ran*, en este caso la diferencia entre ellos fue de apenas 1.87 segundos por cada 10,000 evaluaciones. En este algoritmo el esquema adaptativo fue uno de los métodos más lentos sin embargo, la diferencia entre éste y el método más rápido fue de apenas 0.95 segundos por cada 10,000 evaluaciones.
En el algoritmo de Evolución Diferencial el método con el menor consumo de tiempo fue $Ref\&DeB$ y el más lento fue $Wra\&RaB$, sin embargo la diferencia entre uno y el otro fue mínima, apenas de 0.33 segundos por cada 10,000 evaluaciones, por su parte el ABCHM mostró un consumo de tiempo promedio.
De forma general, se observó que el consumo de tiempo del ABCHS depende mucho del consumo de tiempo de los métodos que emplea y principalmente del consumo de tiempo de su método base, pues ya sea en DE o PSO se observó que de forma general, el consumo de tiempo del ABCHS fue un poco menor del que consume su método base.

8.1.2. ABCHS con algoritmos especializados en optimización restringida

En los experimentos en los que se utilizó SAM-PSO, es decir, la versión de PSO especializada en optimización restringida, se observó que todos los métodos comparados, incluido el ABCHS, mostraron la misma capacidad para llegar a la región factible, probablemente debido a que el

algoritmo SAM-PSO en sí mismo tiene una excelente capacidad para este fin, sin embargo, cuando se comparó la calidad de las soluciones factibles encontradas, el ABCHS mostró un rendimiento notablemente superior con respecto al resto de los métodos.
En los experimentos en los cuales se utilizó el algoritmo de Evolución Diferencial especializado en optimización restringida (ICDE), se observó que el ABCHS mostró un mejor rendimiento que el resto de los métodos, tanto en la capacidad para llegar a la región factible como en la calidad de las soluciones encontradas.

8.2. Observaciones sobre Centroid

Adicionalmente a la propuesta principal, se presentaron dos propuestas secundarias, la primera consiste en el método *Centroid K+1*, el cual reubica los vectores inválidos dentro del espacio de búsqueda, en el centroide de un área formada por $K+1$ vectores, uno de ellos tomado de la población y K vectores reparados de forma aleatoria.
El método *Centroid* mostró un rendimiento competitivo, de manera general fue el tercero en rendimiento, solo después de ABCHS y *Res&Ran*. La principal ventaja del método *Centroid* es su capacidad para guiar las soluciones reparadas hacia espacios más promisorios, lo cual representa una buena ventaja, principalmente en problemas de optimización restringida.
Se realizaron pruebas adicionales en DE para ajustar el valor del parámetro K,en estas pruebas se observó que al aumentar el valor de K, disminuye la calidad de los vectores reparados y aumenta el tiempo de procesamiento debido a que es necesario generar más vectores aleatorios y a que se incrementa el número de reparaciones. Por otro lado, las variantes con valores pequeños en K como *K=1* o *K=2* permitieron obtener una mayor cantidad de soluciones factibles, así como soluciones de mejor calidad, por esto mismo, se recomienda usar la variante *Centroid 1+1*, es decir, aquella que sólo emplea un vector aleatorio y otro tomado de la población.

8.3. Observaciones sobre *Res&Ran*

La segunda propuesta secundaria consistió en una mejora al método *Resampling* para Evolución Diferencial; la cual consiste en repetir el operador de mutación diferencial un máximo de $3 \times D$ veces (donde D es la dimensionalidad del problema a resolver), si después de esto no es posible obtener un vector mutante válido se repara el vector de forma aleatoria. Esta propuesta es, por tanto, un híbrido de los métodos *Resampling* y *Random*, al cual hemos llamado *Res&Ran*.
El método *Res&Ran* mostró un rendimiento competitivo, de manera general fue el segundo en rendimiento, solo después de ABCHS, este método destaca por su excelente capacidad para llegar a la

región factible y como apoyo en la exploración del espacio de búsqueda; sin embargo, debe tomarse en cuenta que fue el método que mostró un mayor consumo de tiempo y debería ser aplicado con cautela.

8.4. Métodos híbridos en PSO

Se realizaron experimentos adicionales con el algoritmo *PSO* para comprobar el impacto que tienen las estrategias que permiten modificar la velocidad de las partículas que salen del espacio de búsqueda al ser combinadas con determinados métodos que permiten modificar su posición. En estos experimentos se obtuvieron los mejores seis métodos híbridos: *Cen&DeB*, *Evo&DeB*, *Ref&DeB*, *Bou&DeB*, *Ran&RaB*, *Wra&RaB*, los cuales emplean los siguientes métodos para manipular las posiciones: *Centroid (Cen)*, *Evolutionary (Evo)*, *Reflection (Ref)*, *Boundary (Bou)*, *Random (Ran)* y *Wrapping (Wra)* y como estrategias para manipular la velocidad: *Deterministic Back (DeB)* y *Random Back (RaB)*.
En este experimento se obtuvo evidencia de la importancia de emplear alguna estrategia para manipular la velocidad, además de la posición, de las partículas que salen del espacio de búsqueda; los resultados indican que las mejores estrategias invierten la velocidad de las partículas al multiplicarla por un factor $-\lambda \ \ (0 < \lambda < 1)$ tal como lo hacen DeB y RaB, esto mismo se sustenta en investigaciones previas en las cuales se ha destacado la importancia de modificar la velocidad de las partículas que salen del espacio de búsqueda [21, 25].

8.5. Cumplimiento de objetivos e hipótesis

En cuanto a los objetivos planteados al inicio de la presente investigación, el diseño del ABCHS, presentado en la Sección 4.1 cumple con el objetivo principal y con los objetivos específicos referentes a la forma en que se emplea la información de factibilidad y el mecanismo de aprendizaje para elegir al BCHM más adecuado al problema a resolver. El conjunto de métodos para el manejo de restricciones de límite empleados dentro del ABCHS fueron determinados en el Capítulo 6.
En la Sección 5.2 se identificaron dos conjuntos de problemas de optimización numérica restringida, con las características planteadas en los objetivos, mismos que fueron empleados en la etapa experimental.
En el Capítulo 3 se describieron los principales métodos para el manejo de restricciones de límite, mismos que fueron codificados para la etapa experimental presentada en el Capítulo 7.
Se seleccionaron y codificaron, para la etapa experimental, los principales algoritmos de optimización numérica inspirados en la naturaleza, uno evolutivo y otro de inteligencia colectiva, ambos

en su versión canónica y una versión del estado del arte especializada en problemas de optimización numérica restringida. En el Capítulo 7 también se presentaron los resultados del estudio comparativo entre el esquema adaptativo propuesto en el presente trabajo y los principales métodos empleados en el manejo de restricciones de límite.
Por lo anteriormente expuesto, se considera que se alcanzaron todos los objetivos planteados en el presente trabajo.
Finalmente, la hipótesis planteada al inicio de la presente investigación establece que si se cuenta con un conjunto de BCHMs y algún mecanismo adaptable que permita aplicar aquel BCHM que genere las mejores soluciones, dependiendo del problema y de la información de factibilidad, se mejorará tanto la cantidad como la calidad de las soluciones factibles encontradas por el algoritmo de optimización.
Como respuesta puntual a esta hipótesis, en las Figuras 7.1, 7.2, 7.3, y 7.4 se presentó, de manera resumida, el número de ejecuciones factibles y el rendimiento general comparado, la primera variable mide la cantidad de soluciones factibles y la segunda la calidad de las mismas. En las cuatro imágenes se pudo observar que el esquema adaptativo propuesto en el presente trabajo obtuvo una mayor cantidad de soluciones factibles y sus soluciones fueron de mejor calidad, con lo que queda demostrada la hipótesis.

8.6. Trabajo futuro

El esquema adaptativo presentado en el presente trabajo abre nuevas preguntas que sugieren investigaciones posteriores; en primer lugar, el hecho de que el ABCHS tiene la flexibilidad para aplicar un conjunto de BCHMs y poder aplicar mayormente uno u otro BCHM de manera adaptativa, parece ser una propuesta natural para el manejo de límites en problemas de optimización dinámica.
Por otro lado, siendo que en ocasiones el ABCHS aplica de manera simultánea dos o más BCHMs nos invita a preguntarnos si alguno de éstos BCHMs tiene éxito en una dimensión o en un área específica y otro BCHM tiene un buen rendimiento en otras dimensiones o áreas, por donde puedan estar saliendo las soluciones del espacio de búsqueda, en tal caso ¿se podría aplicar un BCHM en cierta área en específico?, esta pregunta no es fácil de abordar, seguramente requerirá de la implementación de algún mecanismo en particular para poder diferenciar distintas áreas del espacio de búsqueda.
Otra propuesta interesante es realizar experimentos con el esquema adaptativo propuesto en el presente trabajo (ABCHS) para resolver problemas de muy alta dimensionalidad (large-scale) o probarlo en algoritmos diseñados especialmente para resolver problemas multiobjetivo [8] o con

objetivos dinámicos [41]. Por otro lado, también es importante realizar pruebas en otro tipo de algoritmos de optimización inspirados en la naturaleza como Cuckoo Search [62] o Gravitational Search [45].

Bibliografía

[1] Julio E Alvarez-Benitez, Richard M Everson, and Jonathan E Fieldsend. A mopso algorithm based exclusively on pareto dominance concepts. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 459–473. Springer, 2005.

[2] Jarosław Arabas, Adam Szczepankiewicz, and Tomasz Wroniak. Experimental comparison of methods to handle boundary constraints in differential evolution. In *International Conference on Parallel Problem Solving from Nature*, pages 411–420. Springer, 2010.

[3] Daniel Bratton and James Kennedy. Defining a standard for particle swarm optimization. In *2007 IEEE swarm intelligence symposium*, pages 120–127. IEEE, 2007.

[4] Janez Brest, Sao Greiner, Borko Boskovic, Marjan Mernik, and Viljem Zumer. Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. *IEEE transactions on evolutionary computation*, 10(6):646–657, 2006.

[5] Wei Chu, Xiaogang Gao, and Soroosh Sorooshian. Handling boundary constraints for particle swarm optimization in high-dimensional search space. *Information Sciences*, 181(20):4569–4581, 2011.

[6] Maurice Clerc. Confinements and Biases in Particle Swarm Optimisation. Technical Report hal-00122799, 03 2006. URL https://hal.archives-ouvertes.fr/hal-00122799.

[7] Carlos A Coello Coello. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer methods in applied mechanics and engineering*, 191(11-12):1245–1287, 2002.

[8] Carlos A Coello Coello. Recent trends in evolutionary multiobjective optimization. In *Evolutionary multiobjective optimization*, pages 7–32. Springer, 2005.

[9] Carlos A Coello Coello. Constraint-handling techniques used with evolutionary algorithms.

In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, pages 563–587. ACM, 2016.

[10] Kalyanmoy Deb. An efficient constraint handling method for genetic algorithms. *Computer methods in applied mechanics and engineering*, 186(2):311–338, 2000.

[11] Russ C Eberhart, James Kennedy, et al. A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science*, volume 1, pages 39–43. New York, NY, 1995.

[12] Agoston E Eiben, James E Smith, et al. *Introduction to evolutionary computing*, volume 53. Springer, 2003.

[13] Saber M Elsayed, Ruhul A Sarker, and Daryl L Essam. Memetic multi-topology particle swarm optimizer for constrained optimization. In *2012 IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2012.

[14] Saber M Elsayed, Ruhul A Sarker, and Efrén Mezura-Montes. Self-adaptive mix of particle swarm methodologies for constrained optimization. *Information sciences*, 277:216–233, 2014.

[15] Andries P Engelbrecht. *Fundamentals of computational swarm intelligence*. John Wiley & Sons, 2006.

[16] Vitaliy Feoktistov. *Differential Evolution: In Search of Solutions (Springer Optimization and Its Applications)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387368957.

[17] Amir H Gandomi, Ali R Kashani, and Mehdi Mousavi. Boundary constraint handling affection on slope stability analysis. In *Engineering and Applied Sciences Optimization*, pages 341–358. Springer, 2015.

[18] Amir Hossein Gandomi and Xin-She Yang. Evolutionary boundary constraint handling scheme. *Neural Computing and Applications*, 21(6):1449–1462, 2012.

[19] Prabhakar Gupta, Vineet Agarwal, and Manish Varshney. *Design and analysis of algorithms*. PHI Learning Pvt. Ltd., 2012.

[20] Sabine Helwig and Rolf Wanka. Particle swarm optimization in high-dimensional bounded search spaces. In *2007 IEEE Swarm Intelligence Symposium*, pages 198–205. IEEE, 2007.

[21] Sabine Helwig, Juergen Branke, and Sanaz Mostaghim. Experimental analysis of bound handling techniques in particle swarm optimization. *IEEE Transactions on Evolutionary computation*, 17(2):259–271, 2013.

[22] T. Huang and A.S. Mohan. A hybrid boundary condition for robust particle swarm optimization. *Antennas and Wireless Propagation Letters, IEEE*, 4:112–117, 2005. ISSN 1536-1225. doi: 10.1109/LAWP.2005.846166.

[23] Guanbo Jia, Yong Wang, Zixing Cai, and Yaochu Jin. An improved (μ+ λ)-constrained differential evolution for constrained optimization. *Information Sciences*, 222:302–322, 2013.

[24] Efrén Juárez-Castillo, Nancy Pérez-Castro, and Efrén Mezura-Montes. A novel boundary constraint-handling technique for constrained numerical optimization problems. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 2034–2041. IEEE, 2015.

[25] Efrén Juárez-Castillo, Héctor-Gabriel Acosta-Mesa, and Efrén Mezura-Montes. Empirical study of bound constraint-handling methods in particle swarm optimization for constrained search spaces. In *Evolutionary Computation (CEC), 2017 IEEE Congress on*, pages 604–611. IEEE, 2017.

[26] Efrén Juárez-Castillo, Nancy Pérez-Castro, and Efrén Mezura-Montes. An improved centroid-based boundary constraint-handling method in differential evolution for constrained optimization. *International Journal of Pattern Recognition and Artificial Intelligence*, page 1759023, 2017.

[27] Mudita Juneja and SK Nagar. Particle swarm optimization algorithm and its parameters: A review. In *2016 International Conference on Control, Computing, Communication and Materials (ICCCCM)*, pages 1–5. IEEE, 2016.

[28] James Kennedy. Swarm intelligence. In *Handbook of nature-inspired and innovative computing*, pages 187–219. Springer, 2006.

[29] Slawomir Koziel and Zbigniew Michalewicz. A decoder-based evolutionary algorithm for constrained parameter optimization problems. In *International Conference on Parallel Problem Solving from Nature*, pages 231–240. Springer, 1998.

[30] Saku Kukkonen and Jouni Lampinen. Gde3: The third evolution step of generalized differential evolution. In *2005 IEEE Congress on Evolutionary Computation*, volume 1, pages 443–450. IEEE, 2005.

[31] Jouni Lampinen. A constraint handling approach for the differential evolution algorithm. In *Proceedings of the congress on evolutionary computation*, volume 2, pages 1468–1473. IEEE Computer Society Washington, DC, 2002.

[32] JJ Liang, Thomas Philip Runarsson, Efren Mezura-Montes, Maurice Clerc, PN Suganthan, CA Coello Coello, and Kalyanmoy Deb. Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization. *Journal of Applied Mechanics*, 41(8), 2006.

[33] Hai-Lin Liu, Chaoda Peng, Fangqing Gu, and Jiechang Wen. A constrained multi-objective evolutionary algorithm based on boundary search and archive. *International Journal of Pattern Recognition and Artificial Intelligence*, 30(01):1659002, 2016.

[34] Rammohan Mallipeddi and Ponnuthurai Nagaratnam Suganthan. Problem definitions and evaluation criteria for the cec 2010 competition on constrained real-parameter optimization. *Nanyang Technological University, Singapore*, 2010.

[35] Angelina Jane Reyes Medina, Gregorio Toscano Pulido, and José Gabriel Ramírez-Torres. A comparative study of neighborhood topologies for particle swarm optimizers. In *IJCCI*, pages 152–159, 2009.

[36] Rui Mendes. *Population topologies and their influence in particle swarm performance*. PhD thesis, Citeseer, 2004.

[37] Efrén Mezura-Montes and Carlos A Coello Coello. Constraint-handling in nature-inspired numerical optimization: past, present and future. *Swarm and Evolutionary Computation*, 1 (4):173–194, 2011.

[38] Efrén Mezura-Montes, Omar Cetina-Domınguez, and Betania Hernández-Ocana. Nuevas heurısticas inspiradas en la naturaleza para optimización numérica. *Mecatrónica*, pages 249–272, 2010.

[39] Efrén Mezura-Montes, Mariana Edith Miranda-Varela, and Rubí del Carmen Gómez-Ramón. Differential evolution in constrained numerical optimization: an empirical study. *Information Sciences*, 180(22):4223–4262, 2010.

[40] Sanaz Mostaghim, Werner Halter, and Anja Wille. Linear multi-objective particle swarm optimization. *Stigmergy optimization of computational science*, 31:209–237, 2006.

[41] Trung Thanh Nguyen, Shengxiang Yang, and Juergen Branke. Evolutionary dynamic optimization: A survey of the state of the art. *Swarm and Evolutionary Computation*, 6:1–24, 2012.

[42] Nikhil Padhye, Kalyanmoy Deb, and Pulkit Mittal. Boundary handling approaches in particle swarm optimization. In *Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012)*, pages 287–298. Springer, 2013.

[43] Kenneth Price, Rainer M Storn, and Jouni A Lampinen. Differential evolution: A practical approach to global optimization (natural computing series). 2005.

[44] Magdalena Purchla, Mateusz Malanowski, Paweł Terlecki, and Jarosław Arabas. Experimental comparison of repair methods for box constraints. In *Proc. 7th National Conf. on Evolutionary Computation and Global Optimisation*, pages 135–142, 2004.

[45] Esmat Rashedi, Hossein Nezamabadi-Pour, and Saeid Saryazdi. Gsa: a gravitational search algorithm. *Information sciences*, 179(13):2232–2248, 2009.

[46] Jacob Robinson and Yahya Rahmat-Samii. Particle swarm optimization in electromagnetics. *IEEE transactions on antennas and propagation*, 52(2):397–407, 2004.

[47] Jani Ronkkonen, Saku Kukkonen, and Kenneth V Price. Real-parameter optimization with differential evolution. In *Proc. IEEE CEC*, volume 1, pages 506–513, 2005.

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

[49] Marc Schoenauer and Zbigniew Michalewicz. Evolutionary computation at the edge of feasibility. In *International Conference on Parallel Problem Solving from Nature*, pages 245–254. Springer, 1996.

[50] Yuhui Shi. Particle swarm optimization. *IEEE connections*, 2(1):8–13, 2004.

[51] Yuhui Shi and Russell Eberhart. A modified particle swarm optimizer. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pages 69–73. IEEE, 1998.

[52] Yuhui Shi, Shi Cheng, and Quande Qin. Experimental study on boundary constraints handling in particle swarm optimization: From population diversity perspective. *International Journal of Swarm Intelligence Research*, 2(3):43–69, 2011.

[53] Dan Simon. *Evolutionary optimization algorithms*. John Wiley & Sons, 2013.

[54] Jan A Snyman. Practical mathematical optimization. 2005.

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

[56] Indrajit N Trivedi, Amir H Gandomi, Pradeep Jangir, and Narottam Jangir. Study of different boundary constraint handling schemes in interior search algorithm. In *ICAIECES-2016, International Conference on Artificial Intelligence and Evolutionary Computations in Engineering Systems.*, 2015.

[57] Simon Wessing. Repair methods for box constraints revisited. In *European Conference on the Applications of Evolutionary Computation*, pages 469–478. Springer, 2013.

[58] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[59] David H Wolpert and William G Macready. Coevolutionary free lunches. *IEEE Transactions on Evolutionary Computation*, 9(6):721–735, 2005.

[60] Guohua Wu, R Mallipeddi, and PN Suganthan. Problem definitions and evaluation criteria for the cec 2017 competition on constrained real-parameter optimization. *National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore, Technical Report*, 2016.

[61] Shenheng Xu and Yahya Rahmat-Samii. Boundary conditions in particle swarm optimization revisited. *IEEE Transactions on Antennas and Propagation*, 55(3):760–765, 2007.

[62] Xin-She Yang and Suash Deb. Cuckoo search: recent advances and applications. *Neural Computing and Applications*, 24(1):169–174, 2014.

[63] Jingqiao Zhang and Arthur C Sanderson. *Adaptive differential evolution*. Springer, 2009.

[64] Wen-Jun Zhang, Xiao-Feng Xie, and De-Chun Bi. Handling boundary constraints for numerical optimization by particle swarm flying in periodic search space. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 2, pages 2307–2311. IEEE, 2004.

[65] Jinghui Zhong, Xiaomin Hu, Jun Zhang, and Min Gu. Comparison of performance between different selection strategies on simple genetic algorithms. In *Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*, volume 2, pages 1115–1121. IEEE, 2005.

Apéndice A

Definiciones del conjunto de funciones del CEC 2006

g01

Minimizar:

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

sujeta a:

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

donde los límites son: $0 \leq x_i \leq 1 (i = 1, \ldots, 9), 0 \leq x_i \leq 100 (i = 10, 11, 12)$ y $0 \leq x13 \leq 1$

g02

Minimizar:

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

sujeta a:

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

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

donde $n = 20$ y $0 < x_i \leq 10 (i = 1, \ldots, n)$

g03

Minimizar:

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

sujeta a:

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

donde $n = 10$ y $0 \leq x_i \leq 1 (i = 1, \ldots, n)$.

g04

Minimizar:

$$f(\vec{x}) = 5.3578547 x_3^2 + 0.8356891 x_1 x_5 + 37.293239 x_1 - 40792.141$$

sujeta a:

$$g_1(\vec{x}) = 85.334407 + 0.0056858 x_2 x_5 + 0.0006262 x_1 x_4 - 0.0022053 x_3 x_5 - 92 \leq 0$$
$$g_2(\vec{x}) = -85.334407 - 0.0056858 x_2 x_5 - 0.0006262 x_1 x_4 + 0.0022053 x_3 x_5 \leq 0$$
$$g_3(\vec{x}) = 80.51249 + 0.0071317 x_2 x_5 + 0.0029955 x_1 x_2 + 0.0021813 x_3^2 - 110 \leq 0$$
$$g_4(\vec{x}) = -80.51249 - 0.0071317 x_2 x_5 - 0.0029955 x_1 x_2 - 0.0021813 x_3^2 + 90 \leq 0$$
$$g_5(\vec{x}) = 9.300961 + 0.0047026 x_3 x_5 + 0.0012547 x_1 x_3 + 0.0019085 x_3 x_4 - 25 \leq 0$$
$$g_6(\vec{x}) = -9.300961 - 0.0047026 x_3 x_5 - 0.0012547 x_1 x_3 - 0.0019085 x_3 x_4 + 20 \leq 0$$

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

g05

Minimizar:

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

sujeta a:

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

donde $0 \leq x_1 \leq 1200$, $0 \leq x_2 \leq 1200$, $-0.55 \leq x_3 \leq 0.55$ y $-0.55 \leq x_4 \leq 0.55$.

g06

Minimizar:

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

sujeta a:

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

donde $13 \leq x_1 \leq 100$ y $0 \leq x_2 \leq 100$

g07

Minimizar:

$$\begin{aligned}
f(\vec{x}) = {} & x_1^2 + x_2^2 + x_1 x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 \\
& + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45
\end{aligned}$$

sujeta a:

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

donde $-10 \leq x_i \leq 10 (i = 1, \ldots, 10)$

g08

Minimizar:

$$f(\vec{x}) = -\frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$$

sujeta a:

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

donde $0 \leq x_1 \leq 10$ y $0 \leq x_2 \leq 10$

g09

Minimizar:

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

sujeta a:

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

donde $-10 \leq x_i \leq 10 for (i = 1, \ldots, 7)$

g10

Minimizar:

$$f(\vec{x}) = x_1 + x_2 + x_3$$

sujeta a:

$$g_1(\vec{x}) = -1 + 0.0025(x_4 + x_6) \leq 0$$
$$g_2(\vec{x}) = -1 + 0.0025(x_5 + x_7 - x_4) \leq 0$$
$$g_3(\vec{x}) = -1 + 0.01(x_8 - x_5) \leq 0$$
$$g_4(\vec{x}) = -x_1 x_6 + 833.33252 x_4 + 100 x_1 - 83333.333 \leq 0$$
$$g_5(\vec{x}) = -x_2 x_7 + 1250 x_5 + x_2 x_4 - 1250 x_4 \leq 0$$
$$g_6(\vec{x}) = -x_3 x_8 + 1250000 + x_3 x_5 - 2500 x_5 \leq 0$$

donde $100 \leq x_1 \leq 10000$, $1000 \leq x_i \leq 10000 (i = 2,3)$ y $10 \leq x_i \leq 1000 (i = 4,\ldots,8)$

g11

Minimizar:

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

sujeta a:

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

donde $-1 \leq x_1 \leq 1$ y $-1 \leq x_2 \leq 1$

g12

Minimizar:

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

sujeta a:

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

donde $0 \leq x_i \leq 10 (i = 1,2,3)$ y $p,q,r = 1,2,\ldots,9$

g13

Minimizar:

$$f(\vec{x}) = e^{x_1 x_2 x_3 x_4 x_5}$$

sujeta a:

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

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

g14

Minimizar:

$$f(\vec{x}) = \sum_{i=1}^{10} x_i \left(c_i + ln \frac{x_i}{\sum_{j=1}^{10} x_j} \right)$$

sujeta a:

$$h_1(\vec{x}) = x_1 + 2x_2 + 2x_3 + x_6 + x_{10} - 2 = 0$$
$$h_2(\vec{x}) = x_4 + 2x_5 + x_6 + x_7 - 1 = 0$$
$$h_3(\vec{x}) = x_3 + x_7 + x_8 + 2x_9 + x_{10} = 0$$

donde los límites son $0 < x_i \leq 10 (i = 1, \ldots, 10)$, y $c_1 = -6.089$, $c_2 = -17.164$, $c_3 = -34.054$, $c_4 = -5.914$, $c_5 = -24.721$, $c_6 = -14.986$, $c_7 = -24.1$, $c_8 = -10.708$, $c_9 = -26.662$, $c_{10} = -22.179$

g15

Minimizar:

$$f(\vec{x}) = 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1 x_2 - x_1 x_3$$

sujeta a:

$$h_1(\vec{x}) = x_1^2 + x_2^2 + x_3^2 - 25 = 0$$
$$h_2(\vec{x}) = 8x_1 + 14x_2 + 7x_3 - 56 = 0$$

donde los límites son $0 \leq x_i \leq 10 (i = 1,2,3)$

g16

Minimizar:

$$f(\vec{x}) = 0.000117y_{14} + 0.1365 + 0.0002358y_{13} + 0.000001502y_{16} + 0.0321y_{12}$$
$$+ 0.004324y_5 + 0.0001\frac{c_{15}}{c_{16}} + 37.48\frac{y_2}{c_{12}} - 0.0000005843y_{17}$$

sujeta a:

$$g_1(\vec{x}) = \frac{0.28}{0.72}y_5 - y_4 \leq 0$$
$$g_2(\vec{x}) = x_3 - 1.5x_2 \leq 0$$
$$g_3(\vec{x}) = 3496\frac{y_2}{c_{12}} - 21 \leq 0$$
$$g_4(\vec{x}) = 110.6 + y_1 - \frac{62212}{c_{17}} \leq 0$$
$$g_5(\vec{x}) = 213.1 - y_1 \leq 0$$
$$g_6(\vec{x}) = y_1 - 405.23 \leq 0$$
$$g_7(\vec{x}) = 17.505 - y_2 \leq 0$$
$$g_8(\vec{x}) = y_2 - 1053.6667 \leq 0$$
$$g_9(\vec{x}) = 11.275 - y_3 \leq 0$$
$$g_{10}(\vec{x}) = y_3 - 35.03 \leq 0$$
$$g_{11}(\vec{x}) = 214.228 - y_4 \leq 0$$
$$g_{12}(\vec{x}) = y_4 - 665.585 \leq 0$$
$$g_{13}(\vec{x}) = 7.458 - y_5 \leq 0$$
$$g_{14}(\vec{x}) = y_5 - 584.463 \leq 0$$
$$g_{15}(\vec{x}) = 0.961 - y_6 \leq 0$$
$$g_{16}(\vec{x}) = y_6 - 265.916 \leq 0$$
$$g_{17}(\vec{x}) = 1.612 - y_7 \leq 0$$
$$g_{18}(\vec{x}) = y_7 - 7.046 \leq 0$$
$$g_{19}(\vec{x}) = 0.146 - y_8 \leq 0$$
$$g_{20}(\vec{x}) = y_8 - 0.222 \leq 0$$
$$g_{21}(\vec{x}) = 107.99 - y_9 \leq 0$$
$$g_{22}(\vec{x}) = y_9 - 273.366 \leq 0$$
$$g_{23}(\vec{x}) = 922.693 - y_{10} \leq 0$$
$$g_{24}(\vec{x}) = y_{10} - 1286.105 \leq 0$$
$$g_{25}(\vec{x}) = 926.832 - y_{11} \leq 0$$
$$g_{26}(\vec{x}) = y_{11} - 1444.046 \leq 0$$
$$g_{27}(\vec{x}) = 18.766 - y_{12} \leq 0$$
$$g_{28}(\vec{x}) = y_{12} - 537.141 \leq 0$$

$$g_{29}(\vec{x}) = 1072.163 - y_{13} \leq 0$$
$$g_{30}(\vec{x}) = y_{13} - 3247.039 \leq 0$$
$$g_{31}(\vec{x}) = 8961.448 - y_{14} \leq 0$$
$$g_{32}(\vec{x}) = y_{14} - 26844.086 \leq 0$$
$$g_{33}(\vec{x}) = 0.063 - y_{15} \leq 0$$
$$g_{34}(\vec{x}) = y_{15} - 0.386 \leq 0$$
$$g_{35}(\vec{x}) = 71084.33 - y_{16} \leq 0$$
$$g_{36}(\vec{x}) = -140000 + y_{16} \leq 0$$
$$g_{37}(\vec{x}) = 2802713 - y_{17} \leq 0$$
$$g_{38}(\vec{x}) = y_{17} - 12146108 \leq 0$$

donde:

$$y_1 = x_2 + x_3 + 41.6$$
$$c_1 = 0.024x_4 - 4.62$$
$$y_2 = \frac{12.5}{c_1} + 12$$
$$c_2 = 0.0003535x_1^2 + 0.5311x_1 + 0.08705y_2x_1$$
$$c_3 = 0.052x_1 + 78 + 0.002377y_2x_1$$
$$y_3 = \frac{c_2}{c_3}$$
$$y_4 = 19y_3$$
$$c_4 = 0.04782(x_1 - y_3) + \frac{0.1956(x_1 - y_3)^2}{x_2} + 0.6376y_4 + 1.594y_3$$
$$c_5 = 100x_2$$
$$c_6 = x_1 - y_3 - y_4$$
$$c_7 = 0.950 - \frac{c_4}{c_5}$$
$$y_5 = c_6c_7$$
$$y_6 = x_1 - y_5 - y_4 - y_3$$
$$c_8 = (y_5 + y_4)0.995$$
$$y_7 = \frac{c_8}{y_1}$$
$$y_8 = \frac{c_8}{3798}$$

$$
\begin{aligned}
c_9 &= y_7 - \frac{0.0663 y_7}{y_8} - 0.3153 \\
y_9 &= \frac{96.82}{c_9} + 0.321 y_1 \\
y_{10} &= 1.29 y_5 + 1.258 y_4 + 2.29 y_3 + 1.71 y_6 \\
y_{11} &= 1.71 x_1 - 0.452 y_4 + 0.580 y_3 \\
c_{10} &= \frac{12.3}{752.3} \\
c_{11} &= (1.75 y_2)(0.995 x_1) \\
c_{12} &= 0.995 y_{10} + 1998 \\
y_{12} &= c_{10} x_1 + \frac{c_{11}}{c_{12}} \\
y_{13} &= c_{12} - 1.75 y_2 \\
y_{14} &= 3623 + 64.4 x_2 + 58.4 x_3 + \frac{146312}{y_9 + x_5} \\
c_{13} &= 0.995 y_{10} + 60.8 x_2 + 48 x_4 - 0.1121 y_{14} - 5095 \\
y_{15} &= \frac{y_{13}}{c_{13}} \\
y_{16} &= 148000 - 331000 y_{15} + 40 y_{13} - 61 y_{15} y_{13} \\
c_{14} &= 2324 y_{10} - 28740000 y_2 \\
y_{17} &= 14130000 - 1328 y_{10} - 531 y_{11} + \frac{c_{14}}{c_{12}} \\
c_{15} &= \frac{y_{13}}{y_{15}} - \frac{y_1 3}{0.52} \\
c_{16} &= 1.104 - 0.72 y_{15} \\
c_{17} &= y_9 + x_5
\end{aligned}
$$

donde los límites son $704.4148 \leq x_1 \leq 906.3855$, $68.6 \leq x_2 \leq 288.88$, $0 \leq x_3 \leq 134.75$, $193 \leq x_4 \leq 287.0966$ y $25 \leq x_5 \leq 84.1988$.

g17

Minimizar:

$$f(\vec{x}) = f(x_1) + f(x_2)$$

donde:

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

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

sujeta a:

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

donde los límites son $0 \leq x_1 \leq 400$, $0 \leq x_2 \leq 1000$, $340 \leq x_3 \leq 420$, $340 \leq x_4 \leq 420$, $-1000 \leq x_5 \leq 1000$ y $0 \leq x_6 \leq 0.5236$

g18

Minimizar:

$$f(\vec{x}) = -0.5(x_1 x_4 - x_2 x_3 + x_3 x_9 - x_5 x_9 + x_5 x_8 - x_6 x_7)$$

sujeta a:

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

$$g_{10}(\vec{x}) = x_2x_3 - x_1x_4 \leq 0$$
$$g_{11}(\vec{x}) = -x_3x_9 \leq 0$$
$$g_{12}(\vec{x}) = x_5x_9 \leq 0$$
$$g_{13}(\vec{x}) = x_6x_7 - x_5x_8 \leq 0$$

donde los límites son $-10 \leq x_i \leq 10 (i = 1, \ldots, 8)$ y $0 \leq x_9 \leq 20$

g19

Minimizar:

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

sujeta a:

$$g(\vec{x}) = -2\sum_{i=1}^{5} c_{i,j}x_{(10+j)} - e_j + \sum_{i=1}^{10} a_{i,j}x_i \leq 0 \quad j = 1, \ldots, 5$$

donde $b = [-40, -2, -0.25, -4, -4, -1, -40, -60, 5, 1]$. Los límites son $0 \leq x_i \leq 10 (i = 1, \ldots, 15)$

g20

Minimizar:

$$f(\vec{x}) = \sum_{i=1}^{24} a_i x_i$$

sujeta a:

$$g_i(\vec{x}) = \frac{(x_i + x_{(i+12)})}{\sum_{j=1}^{24} x_j + e_i} \leq 0 \quad i = 1, 2, 3$$
$$g_i(\vec{x}) = \frac{(x_{(i+3)} + x_{(i+15)})}{\sum_{j=1}^{24} x_j + e_i} \leq 0 \quad i = 4, 5, 6$$
$$h_i(\vec{x}) = \frac{x_{(i+12)}}{b_{(i+12)}\sum_{j=13}^{24} \frac{x_j}{b_j}} - \frac{c_i x_i}{40 b_i \sum_{j=1}^{12} \frac{x_j}{b_j}} = 0 \quad i = 1, \ldots, 12$$
$$h_{13}(\vec{x}) = \sum_{i=1}^{24} x_i - 1 = 0$$
$$h_{14}(\vec{x}) = \sum_{i=1}^{12} \frac{x_i}{d_i} + k\sum_{i=13}^{24} \frac{x_i}{b_i} - 1.671 = 0$$

donde $k = (0.7302)(530)(\frac{14.7}{40})$. Los límites son $0 \leq x_i \leq 10 (i = 1, \ldots, 24)$

g21

Minimizar:

$$f(\vec{x}) = x_1$$

sujeta a:

$$g_1(\vec{x}) = -x_1 + 35x_2^{0.62} + 35x_3^{0.6} \leq 0$$
$$h_1(\vec{x}) = -300x_3 + 7500x_5 - 7500x_6 - 25x_4x_5 + 25x_4x_6 + x_3x_4 = 0$$
$$h_2(\vec{x}) = 100x_2 + 155.365x_4 + 2500x_7 - x_2x_4 - 25x_4x_7 - 15536.5 = 0$$
$$h_3(\vec{x}) = -x_5 + ln(-x_4 + 900) = 0$$
$$h_4(\vec{x}) = -x_6 + ln(x_4 + 300) = 0$$
$$h_5(\vec{x}) = -x_7 + ln(-2x_4 + 700) = 0$$

donde los límites son $0 \leq x_1 \leq 1000$, $0 \leq x_2, x_3 \leq 40$, $100 \leq x_4 \leq 300$, $6.3 \leq x_5 \leq 6.7$, $5.9 \leq x_6 \leq 6.4$ y $4.5 \leq x_7 \leq 6.25$

g22

Minimizar:

$$f(\vec{x}) = x_1$$

sujeta a:

$$g_1(\vec{x}) = -x_1 + x_2^{0.62} + x_3^{0.6} + x_4^{0.6} \leq 0$$
$$h_1(\vec{x}) = x_5 - 100000x_8 + 1 \times 10^7 = 0$$
$$h_2(\vec{x}) = x_6 + 100000x_8 - 100000x9 = 0$$
$$h_3(\vec{x}) = x_7 + 100000x_9 - 5 \times 10^7 = 0$$
$$h_4(\vec{x}) = x_5 + 100000x_{10} - 3.3 \times 10^7 = 0$$
$$h_5(\vec{x}) = x_6 + 100000x_{11} - 4.4 \times 10^7 = 0$$
$$h_6(\vec{x}) = x_7 + 100000x_{12} - 6.6 \times 10^7 = 0$$
$$h_7(\vec{x}) = x_5 - 120x_2x_{13} = 0$$
$$h_8(\vec{x}) = x_6 - 80x_3x_{14} = 0$$
$$h_9(\vec{x}) = x_7 - 40x_4x_{15} = 0$$
$$h_{10}(\vec{x}) = x_8 - x_{11} + x_{16} = 0$$
$$h_{11}(\vec{x}) = x_9 - x_{12} + x_{17} = 0$$
$$h_{12}(\vec{x}) = -x_{18} + ln(x_{10} - 100) = 0$$

$$h_{13}(\vec{x}) = -x_{19} + ln(-x_8 + 300) = 0$$
$$h_{14}(\vec{x}) = -x_{20} + ln(x_{16}) = 0$$
$$h_{15}(\vec{x}) = -x_{21} + ln(-x_9 + 400) = 0$$
$$h_{16}(\vec{x}) = -x_{22} + ln(x_{17}) = 0$$
$$h_{17}(\vec{x}) = -x_8 - x_{10} + x_{13}x_{18} - x_{13}x_{19} + 400 = 0$$
$$h_{18}(\vec{x}) = x_8 - x_9 - x_{11} + x_{14}x_{20} - x_{14}x_{21} + 400 = 0$$
$$h_{19}(\vec{x}) = x_9 - x_{12} - 4.60517x_{15} + x_{15}x_{22} + 100 = 0$$

donde los límites son $0 \leq x_1 \leq 20000$, $0 \leq x_2, x_3, x_4 \leq 1 \times 10^6$, $0 \leq x_5, x_6, x_7 \leq 4 \times 10^7$, $100 \leq x_8 \leq 299.99$, $100 \leq x_9 \leq 399.99$, $100.01 \leq x_{10} \leq 300$, $100 \leq x_{11} \leq 400$, $100 \leq x_{12} \leq 600$, $0 \leq x_{13}, x_{14}, x_{15} \leq 500$, $0.01 \leq x_{16} \leq 300$, $0.01 \leq x_{17} \leq 400$, $-4.7 \leq x_{18}, x_{19}, x_{20}, x_{21}, x_{22} \leq 6.25$

g23

Minimizar:

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

sujeta a:

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

donde los límites son $0 \leq x_1, x_2, x_6 \leq 300$, $0 \leq x_3, x_5, x_7 \leq 100$, $0 \leq x_4, x_8 \leq 200$ y $0.01 \leq x_9 \leq 0.03$

g24

Minimizar:

$$f(\vec{x}) = -x_1 - x_2$$

sujeta a:

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

donde los límites son: $0 \leq x_1 \leq 3$ y $0 \leq x_2 \leq 4$

Apéndice B

Definiciones del conjunto de funciones del CEC 2010

C01

Minimizar:

$$f(x) = -\left|\sum_{i=1}^{D} cos^4(z_i) - 2\prod_{i=1}^{D} cos^2(z_i)\right|$$

$$z = x - o$$

sujeta a:

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

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

$$x \in [0, 10]^D$$

C02

Minimizar:

$$f(x) = max(z)$$

$$z = x - o$$

$$y = z - 0.5$$

sujeta a:

$$g_1(x) = 10 - \frac{1}{D}\sum_{i=1}^{D}\left[z_i^2 - 10cos(2\Pi z_i) + 10\right] \leq 0$$

$$g_2(x) = \frac{1}{D}\sum_{i=1}^{D}\left[z_i^2 - 10cos(2\Pi z_i) + 10\right] - 15 \leq 0$$

$$h(x) = \frac{1}{D}\sum_{i=1}^{D}\left[y_i^2 - 10cos(2\Pi y_i) + 10\right] - 20 = 0$$

$$x \in [-5.12, 5.12]^D$$

C03

Minimizar:

$$f(x) = \sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2)$$

$$z = x - o$$

sujeta a:

$$h(x) = \sum_{i=1}^{D-1}(z_i - z_{i+1})^2 = 0$$

$$x \in [-1000, 1000]^D$$

C04

Minimizar:

$$f(x) = max(z)$$

$$z = x - o$$

sujeta a:

$$h_1(x) = \frac{1}{D}\sum_{i=1}^{D}(z_i cos(\sqrt{|z_i|})) = 0$$

$$h_2(x) = \sum_{i=1}^{D/2-1}(z_i - z_{i+1})^2 = 0$$

$$h_3(x) = \sum_{i=1}^{D-1}(z_i^2 - z_{i+1})^2 = 0$$

$$h_4(x) = \sum_{i=1}^{D} z = 0$$

$$x \in [-50, 50]^D$$

C05

Minimizar:

$$f(x) = max(z)$$
$$z = x - o$$

sujeta a:

$$h_1(x) = \frac{1}{D}\sum_{i=1}^{D}(-z_i sin(\sqrt{|z_i|})) = 0$$
$$h_2(x) = \frac{1}{D}\sum_{i=1}^{D}(-z_i cos(0.5\sqrt{|z_i|})) = 0$$
$$x \in [-600, 600]^D$$

C06

Minimizar:

$$f(x) = max(z)$$
$$z = x - o$$
$$y = (x + 483.6106156535 - o)M - 483.6106156535$$

sujeta a:

$$h_1(x) = \frac{1}{D}\sum_{i=1}^{D}(-y_i sin(\sqrt{|y_i|})) = 0$$
$$h_2(x) = \frac{1}{D}\sum_{i=1}^{D}(-y_i cos(0.5\sqrt{|y_i|})) = 0$$
$$x \in [-600, 600]^D$$

C07

Minimizar:

$$f(x) = \sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2)$$
$$z = x + 1 - o$$
$$y = x - o$$

sujeta a:

$$g(x) = 0.5 - \exp\left(-0.1\sqrt{\frac{1}{D}\sum_{i=1}^{D} y_i^2}\right) - 3\exp\left(\frac{1}{D}\sum_{i=1}^{D} cos(0.1y)\right) + \exp(1) \leq 0$$
$$x \in [-140, 140]^D$$

C08

Minimizar:

$$f(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2)$$
$$z = x + 1 - o$$
$$y = (x - o)M$$

sujeta a:

$$g(x) = 0.5 - \exp\left(-0.1\sqrt{\frac{1}{D}\sum_{i=1}^{D} y_i^2}\right) - 3\exp\left(\frac{1}{D}\sum_{i=1}^{D} cos(0.1y)\right) + \exp(1) \leq 0$$
$$x \in [-140, 140]^D$$

C09

Minimizar:

$$f(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2)$$
$$z = x + 1 - o$$
$$y = x - o$$

sujeta a:

$$h(x) = \sum_{i=1}^{D} (ysin(\sqrt{|y_i|})) = 0$$
$$x \in [-500, 500]^D$$

C10

Minimizar:

$$f(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2)$$
$$z = x + 1 - o$$
$$y = (x - o)M$$

sujeta a:

$$h(x) = \sum_{i=1}^{D} (y sin(\sqrt{|y_i|})) = 0$$
$$x \in [-500, 500]^D$$

C11

Minimizar:

$$f(x) = \frac{1}{D} \sum_{i=1}^{D} (-z_i cos(2\sqrt{|z_i|}))$$
$$z = (x - o)M$$
$$y = x + 1 - o$$

sujeta a:

$$h(x) = \sum_{i=1}^{D-1} (100(y_i^2 - y_{i+1})^2 + (y_i - 1)^2) = 0$$
$$x \in [-100, 100]^D$$

C12

Minimizar:

$$f(x) = \frac{1}{D} \sum_{i=1}^{D} (z_i sin(\sqrt{|z_i|}))$$
$$z = x - o$$

sujeta a:

$$h(x) = \sum i = 1^{D-1} (z_i^2 - z_{i+1})^2 = 0$$
$$g(x) = \sum i = 1^{D} (z - 100 cos(0.1z) + 10) \leq 0$$
$$x \in [-1000, 1000]^D$$

C13

Minimizar:

$$f(x) = \frac{1}{D}\sum_{i=1}^{D}(-z_i sin(\sqrt{|z_i|}))$$
$$z = x - o$$

sujeta a:

$$g_1(x) = -50 + \frac{1}{100D}\sum_{i}^{D} z_i^2 \leq 0$$
$$g_2(x) = \frac{50}{D}\sum_{i=1}^{D} sin\left(\frac{1}{50}\Pi z\right) \leq 0$$
$$g_3(x) = 75 - 50\left(\sum_{i=1}^{D}\frac{z_i^2}{4000} - \prod_{i=1}^{D} cos\left(\frac{z_i}{\sqrt{i}}\right) + 1\right) \leq 0$$
$$x \in [-500, 500]^D$$

C14

Minimizar:

$$f(x) = \sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z-1)^2)$$
$$z = x + 1 - o$$
$$y = x - o$$

sujeta a:

$$g_1(x) = \sum_{i=1}^{D}(-y_i cos(\sqrt{y_i})) - D \leq 0$$
$$g_2(x) = \sum_{i=1}^{D}(y_i cos(\sqrt{y_i})) - D \leq 0$$
$$g_3(x) = \sum_{i=1}^{D}(y_i sin(\sqrt{y_i})) - 10D \leq 0$$
$$x \in [-1000, 1000]^D$$

C15

Minimizar:

$$f(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z-1)^2)$$
$$z = x + 1 - o$$
$$y = (x-o)M$$

sujeta a:

$$g_1(x) = \sum_{i=1}^{D} (-y_i cos(\sqrt{y_i})) - D \leq 0$$
$$g_2(x) = \sum_{i=1}^{D} (y_i cos(\sqrt{y_i})) - D \leq 0$$
$$g_3(x) = \sum_{i=1}^{D} (y_i sin(\sqrt{y_i})) - 10D \leq 0$$
$$x \in [-1000, 1000]^D$$

C16

Minimizar:

$$f(x) = \sum_{i=1}^{D} \frac{z_i^2}{4000} - \prod_{i=1}^{D} cos\left(\frac{z_i}{\sqrt{i}}\right) + 1$$
$$z = x - o$$

sujeta a:

$$g_1(x) = \sum_{i=1}^{D} [z_i^2 - 100cos(\Pi z_i) + 10] \leq 0$$
$$g_2(x) = \prod_{i=1}^{D} z_i \leq 0$$
$$h_1(x) = \sum_{i=1}^{D} (z_i sin(\sqrt{|z_i|})) = 0$$
$$h_2(x) = \sum_{i=1}^{D} (-z_i sin(\sqrt{|z_i|})) = 0$$
$$x \in [-10, 10]^D$$

C17

Minimizar:

$$f(x) = \sum_{i=1}^{D-1} (z_i - z_{i+1})^2$$

$$z = x - o$$

sujeta a:

$$g_1(x) = \prod_{i=1}^{D} z_i \leq 0$$

$$g_2(x) = \sum_{i=1}^{D} z_i \leq 0$$

$$h_1(x) = \sum_{i=1}^{D} (z_i sin(4\sqrt{z_i})) = 0$$

$$x \in [-10, 10]^D$$

C18

Minimizar:

$$f(x) = \sum_{i=1}^{D-1} (z_i - z_{i+1})^2$$

$$z = x - o$$

sujeta a:

$$g(x) = \frac{1}{D} \sum_{i=1}^{D} (-z_i sin(\sqrt{z_i})) \leq 0$$

$$h(x) = \frac{1}{D} \sum_{i=1}^{D} (z_i sin(\sqrt{z_i})) = 0$$

$$x \in [-50, 50]^D$$

Apéndice C

Publicaciones científicas en extenso.