Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Computación

Diseño de un algoritmo basado en cúmulos de partículas para problemas de alta dimensionalidad

TESIS

Que presenta

Gustavo Pérez Briones

Para obtener el grado de

Maestría en Ciencias

en Computación

Director de la Tesis

Dr. Carlos Artemio Coello Coello

Ciudad de México **Abril, 2025**

Resumen

El algoritmo de optimización mediante cúmulos de partículas (*Particle Swarm Optimization* o PSO) es una metaheurística cuyo principio de funcionamiento se inspira en el comportamiento de ciertos animales que se mueven en grupo, tales como las aves o los peces. Estos animales son capaces de resolver, de forma conjunta, problemas complejos como el buscar fuentes de alimento. Al realizar esta tarea de forma cooperativa, se incrementa el área de búsqueda cubierta y con ello la probabilidad de éxito. Esto contrasta con realizar una búsqueda individual que tomaría mucho tiempo y que, posiblemente, terminaría con un final negativo. Desde la primera versión del algoritmo de PSO publicada en 1995, hasta la fecha, se continúan desarrollando mejoras al algoritmo. Esto se debe, principalmente, al surgimiento de problemas de optimización cada vez más complejos. Uno de ellos es precisamente la optimización global de alta dimensionalidad (100 o más variables), que es un dominio en el que la mayoría de las metaheurísticas utilizadas para optimización suelen tener un rendimiento pobre. Si bien el principio de funcionamiento del PSO original ayuda en gran medida a que el algoritmo sea robusto, a medida que aumenta la dimensionalidad del problema, el tamaño del espacio de búsqueda crece exponencialmente, lo cual incrementa considerablemente la dificultad de encontrar el óptimo global. A este fenómeno se le denomina "maldición de la dimensionalidad" e implica un mayor consumo de los recursos necesarios para encontrar una solución aceptable, ya que resulta más difícil y costoso (computacionalmente hablando) recorrer el espacio de búsqueda. En específico, en este trabajo nos concentramos en mejorar una de las versiones de PSO creadas para resolver problemas de optimización global de alta dimensionalidad. El algoritmo propuesto utiliza una especie de rastro que guardará información sobre cómo se comportan las partículas en determinado momento en el medio en el que se encuentran (espacio de búsqueda). Esto tiene el propósito de ir adaptando en tiempo real el comportamiento social y cognitivo de cada partícula y hacer más eficiente la exploración y la explotación de soluciones prometedoras. Para validar el algoritmo propuesto se utilizó el conjunto de problemas del *2013 IEEE Congress on Evolutionary Computation* (CEC'2013), el cual contiene problemas de optimización global de gran escala. El conjunto consta de quince problemas que involucran diversos desafíos que se presentan en problemas del mundo real, tales como el que haya múltiples óptimos locales o la no separabilidad entre variables. Este conjunto constituye un buen punto de referencia para probar el rendimiento de nuestro algoritmo frente a otros del estado del arte. Los resultados reportados en esta tesis indican que el algoritmo propuesto proporciona mejores resultados que los algoritmos con respecto a los que fue comparado, además de mantener un funcionamiento relativamente simple, lo cual facilita su implementación y uso.

Abstract

Particle Swarm Optimization (PSO) is a metaheuristic which is inspired on the behavior of some animals that move in groups, such as birds and fish. Such animales are capable of solving, collectively, complex problems such as finding food sources. When performing this task in a cooperative way, the search area covered gets increased and this also increases the probability of success. This contrasts with performing individual searches, which would take a lot of time and would probably have a negative ending. Since the original version of PSO published in 1995 to date, a variety of improved versions of the algorithm have been developed. This has been motivated by the rise of problems of increased complexity. Large scale global optimization (100 or more decision variables) is precisely one of such complex problems. Since this is a domain in which most of the existing optimization metaheuristics have a poor performance.

Although the working principle of the original PSO algorithm contributes to its robustness, as we increase the dimensionality of a problem, the size of the search space grows exponentially, which considerably increases the difficulty of finding the global optimum. This phenomenon is known as "the curse of dimensionality" and implies a greater consumption of the resources required to find an acceptable solution, since it becomes more difficult and costly (computationally speaking) to traverse the search space. Specifically, in this work, we focus on improving PSO versions designed to solve large scale global optimization problems.

The proposed algorithm uses some sort of trail that stores information about the behavior of the particles in a given moment in the medium in which they are located (the search space). This aims to adapt, in real time, the social and cognitive behavior of each particle, in order to make more efficient the exploration and the exploitation of promising solutions. In order to validate the proposed algorithm, we adopted the test problems from the *2013 IEEE Congress on Evolutionary Computation* (CEC'2013), which consist of large scale global optimization problems. This set contains fifteen problems that involve several challenges that arise in real-world problems such as the presence of local optima, or non-separability of the decision variables. This set constitutes a good reference to test the performance of our proposed algorithm with respect to others from the state of the art. The results reported in this thesis indicate that the proposed algorithm provides better results than those of the algorithms with respect to which it was compared, while keeping a relatively simple behavior, which facilitates its implementation and use.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Antecentes

Desde la primera versión del algoritmo de optimización por cúmulos de partículas (PSO por sus siglas en inglés) publicada en 1995 [2], distintos autores han buscado mejorar la eficiencia y eficacia de este algoritmo. Para ello, se ha hecho uso de diversas técnicas como por ejemplo: la modificación de parámetros del modelo (ya sea de forma manual o dinámica), combinar el PSO con otras técnicas como algoritmos genéticos, etc. Todo esto para mejorar la exploración y la explotación del cúmulo en el espacio de búsqueda y así obtener la mejor solución posible.
Más recientemente, se han aplicado técnicas de aprendizaje sobre cúmulos de partículas [3] para hacer frente a problemas con alta dimensionalidad. Una de estas estrategias consiste en modificar una partícula elegida bajo algún criterio. Esta partícula tomaría todas o algunas características de una partícula con mejor aptitud, es decir, aprendería de otra partícula cuál es la mejor posición por adoptarse. De forma general, todas estas técnicas buscan hacer eficiente la relación entre la búsqueda de posibles soluciones y la convergencia al mejor resultado posible y con ello mitigar un poco el problema provocado por el gran número de variables en un problema de alta dimensionalidad.

Algunos algoritmos como CSO [4], SLPSO [5] y MSLPSO[3], utilizan el aprendizaje en individuos del cúmulo para encontrar mejores aproximaciones al óptimo global. En una primera etapa se evita la exploración excesiva del espacio de búsqueda aprendiendo directamente de los mejores individuos a través de competencias o identificando cuáles aprenden mejor y que a su vez puedan enseñar a otros. Una segunda etapa evita perder diversidad en las posibles soluciones a través de mecanismos que hagan que se aprenda de un cierto grupo de individuos sin que éstos sean necesariamente los mejores. La investigación aquí propuesta busca mejorar la eficiencia y eficacia del algoritmo de PSO en problemas de alta dimensionalidad. Para esto, tomamos como base el algoritmo del optimizador de cúmulos competitivo [4], en donde los individuos compiten por pares y el individuo perdedor actualiza su posición tomando como referencia la del ganador.

Los resultados esperados podrían tener un impacto significativo en diversas áreas como la optimización de redes neuronales, el diseño de sistemas complejos, la planificación de rutas y la gestión de recursos.

1.2. Estructura de la tesis

Esta tesis está organizada en seis capítulos. El primero de ellos es esta breve introducción en la que hablamos de forma general del algoritmo de PSO, la motivación que nos lleva a realizar este trabajo y los objetivos que deseamos lograr al finalizarlo.

En el capítulo dos presentamos conceptos básicos sobre optimización y algunas nociones de cómputo evolutivo que pueden ser aplicadas al algoritmo de PSO. Además profundizamos en el principio y funcionamiento de dicho algoritmo con el fin de comprender mejor sus propiedades y la importancia que tiene su uso en la optimización de problemas.

El tercer capítulo lo dedicamos a hablar acerca de la alta dimensionalidad y de como ésta afecta al algoritmo de PSO. Por otro lado, mencionamos algunas estrategias comúnmente utilizadas para mitigar este efecto y describimos algunos de los algoritmos del estado del arte especialmente diseñados para intentar atacar esta característica que se presenta al incrementar las variables de un problema propuesto.

En el cuatro capítulo nos adentramos en la optimización global a gran escala, la cual intenta realizar búsquedas mucho más eficientes sin incrementar los recursos que se ocupan para resolver un problema. También hacemos mención de algunos eventos que ayudan a la popularización y estandarización tanto de los problemas de prueba como de la evaluación de la calidad en dichos algoritmos. Con relación a los conjuntos de problemas de prueba, describimos los que fueron especialmente diseñados para probar el desempeño de los algoritmos de optimización global de gran escala.

En el capítulo cinco comenzamos con el diseño de nuestro algoritmo. Para ello iniciamos con la evaluación de tres algoritmos del estado del arte basados en PSO para optimización global a gran escala. Utilizamos cada uno de los conjuntos de problemas de pruebas especializados en optimización global a gran escala en cada algoritmo con el fin de elegir el que presentara el mejor comportamiento y así tener un buen punto de partida para mejorarlo aún más. Una vez elegido el algoritmo de referencia se implementó nuestra estrategia de mejora y se realizó la comparación de resultados usando varios problemas de prueba y algoritmos del estado del arte.

Por último, en el capítulo seis se presentan las conclusiones y las posibles líneas de investigación futura.

1.3. Planteamiento del problema

El algoritmo de optimización mediante cúmulos de partículas es computacionalmente sencillo y eficiente de implementar. Sin embargo, a medida que el número de dimensiones aumenta, su eficiencia se degrada rápidamente porque el algoritmo de PSO no cuenta con operador de cruza (como el algoritmo genético) ni mecanismos que preserven la diversidad. Por ello, es relevante desarrollar versiones del algoritmo de PSO que sean competitivas en problemas de alta dimensionalidad.

Aunque ya existen versiones de PSO que trabajan con alta dimensionalidad, algúnas solo pueden lidiar con un máximo de cien dimensiones o no se cuenta con su código fuente. A la fecha, se sigue trabajando en nuevas versiones del PSO para aumentar la eficiencia del algoritmo y mejorar sus resultados.

1.4. Objetivos

1.4.1. General

- Diseñar e implementar un algoritmo basado en optimización por cúmulos de partículas mono-objetivo que resuelva problemas de alta dimensionalidad.

1.4.2. Particulares

- Implementar el algoritmo de PSO y modificarlo para poder resolver problemas de alta dimensionalidad (hasta 1000 variables).
- Validar el algoritmo propuesto usando varios conjuntos de prueba estándar del área de optimización global de alta dimensionalidad y comparar resultados con respecto a algoritmos basados en PSO del estado del arte.
- Presentar un análisis estadístico de los resultados obtenidos.

Capítulo 2

Conceptos básicos

Los conceptos que se presentan en este capítulo tienen como fin brindar una mejor comprensión de la importancia de este trabajo en su campo. En este capítulo se presentan diversos conceptos básicos que responden a preguntas tales como: ¿qué es la optimización?, ¿qué tipos y métodos para optimizar existen? y ¿en qué casos se utilizan dichos métodos para encontrar la mejor solución a un problema planteado?. En la segunda parte del capítulo se describen las características del algoritmo de Optimización mediante Cúmulos de Partículas (*Particle Swarm Optimization* o PSO). Dicho algoritmo es la base para el diseño de nuestra propuesta, la cual mejora las soluciones a problemas de alta dimensionalidad, en específico para el conjunto de problemas del CEC-LSGO 2013.

2.1. Optimización

La optimización es un proceso que se utiliza para encontrar la mejor solución posible en todo un espacio de posibles soluciones. Para esto se modela el problema en términos de una función objetivo y se utiliza un algoritmo de búsqueda para minimizar o maximizar la función modelada [6]. Para la optimización de la función objetivo se ajustan las variables de decisión que pueden ser continuas o discretas; además, la función a resolver puede o no tener restricciones.

2.2. Tipos de optimización

Al intentar optimizar un problema, éste puede ser analizado desde distintas perspectivas. Por ejemplo, se puede buscar minimizar algún costo, maximizar la eficiencia de un proceso o producto, tomar la mejor decisión de entre un grupo de opciones, etc. Las características particulares de cada problema implican tomar un enfoque más especializado para lograr la solución más efectiva. De tal forma, existen distintas características a tomar en cuenta en un problema para clasificarlo en algún tipo de optimización según su naturaleza. Esto se muestra en el diagrama de la figura 2.1.

Por ejemplo, la naturaleza del algoritmo en el que se basa este trabajo, es la exploración de espacios de búsqueda continuos. En específico, adoptamos al algoritmo de optimización mediante cúmulos de partículas (PSO por sus siglas en inglés), el cual también es una técnica de optimización estocástica ya que se basa en el uso de números generados aleatoriamente para realizar la búsqueda. Pero veamos primero más detalles de los diferentes tipos de optimización descritos en la figura 2.1.

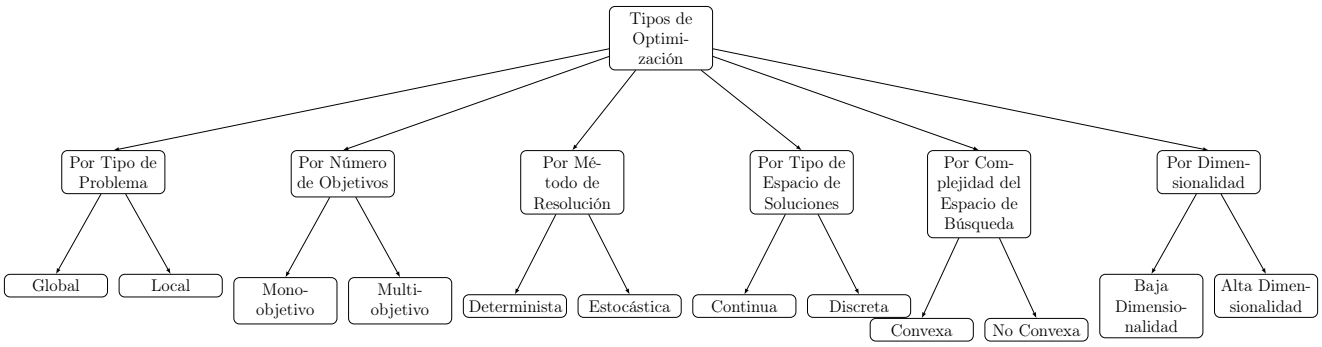

Figura 2.1: Diagrama de características a tomar en cuenta en los diferentes tipos de optimización

2.2.1. Optimización global

Este tipo de optimización busca encontrar la mejor solución en todo el espacio de soluciones del problema. Para ello, es necesario evaluar la función objetivo. Dependiendo del tipo de problema, se busca el mínimo o máximo valor $f_g(x)$ en el espacio de soluciones. Este valor es llamado óptimo global. También se busca evitar quedar atrapado en un óptimo local (éste es un punto en el que se tiene una $f_l(x)$ como solución, y en el cual, todas las soluciones vecinas son peores que la $f_l(x)$ dada, pero, siempre se cumple que $f_g(x) < f_l(x)$ para el caso de mínimos y $f_g(x) > f_l(x)$ para cuando se busca un máximo).

2.2.2. Optimización mono-objetivo

En este tipo de optimización se busca encontrar la mejor solución posible de un problema modelado con una sola función objetivo. Adicionalmente, pueden o no existir restricciones que deban cumplirse para que una solución se considere válida.

2.2.3. Optimización continua

En este tipo de problemas, las variables de decisión de la función objetivo pueden tomar cualquier valor dentro de un intervalo continuo (un número real). Evidentemente, en este caso el número de posibles soluciones se vuelve infinito.

2.2.4. Optimización estocástica

Este tipo de optimización utiliza aleatoriedad para intentar mejorar la exploración en un espacio de búsqueda extenso. Esto produce que en cada ejecución se pueda encontrar una posible mejor solución diferente. Sin embargo, esta misma característica requiere mayor poder de cómputo para realizar dicha exploración en el menor tiempo posible.

2.3. Métodos de optimización

Como podemos ver, un problemas de optimización puede contar con varias características como las ya mencionadas anteriormente. De la misma forma, existen diferentes técnicas para resolver

de la mejor manera posible dicho problema. Como se menciona en [7], al enfrentarnos a espacios de búsqueda muy grandes y en donde los algoritmos más eficientes que existen para resolver el problema requieren tiempo exponencial, resulta obvio que las técnicas clásicas de búsqueda y optimización son insuficientes, por lo que el uso de las metaheurísticas es una opción viable.
A continuación describimos a más detalle los métodos de optimización considerados en este estudio.

2.3.1. Heurísticas

Son estrategias comúnmente utilizadas para realizar una búsqueda en un problema de optimización. El uso de esta estrategia no pretende obtener una solución exacta, si no más bien, busca obtener una solución suficientemente buena en un tiempo razonablemente corto. Más específicamente, podemos definir una heurística de la forma siguiente:

"Una heurística es una técnica que busca soluciones buenas (es decir, cercanas al óptimo) a un costo computacional razonable, sin poder garantizar ni la viabilidad ni la optimalidad, o incluso en muchos casos, sin poder indicar qué tan cerca está una solución particular del óptimo" [8].

Cabe mencionar que, al irse desarrollando esta área, han surgido nuevas características asociadas a las heurísticas que no se ven reflejadas en su definición, tales como las estrategias de aprendizaje adaptativo [8].

2.3.2. Metaheurística

El principio de una metaheurística es el mismo que el de una heurística, con la diferencia de que en este caso se aplican varias heurísticas, las cuales definen una estrategia de búsqueda general para resolver problemas de optimización.

El término fue propuesto por Fred Glover en 1986 y ganó popularidad dentro de la comunidad científica a partir de 1997. Su definición es la siguiente:

"Una metaheurística se refiere a una estrategia principal que guía y modifica otras heurísticas para producir soluciones más allá de aquellas que normalmente se generan en la búsqueda de un óptimo local. Las heurísticas guiadas por tal meta-estrategia pueden ser procedimientos de alto nivel o no ser más que una descripción de los movimientos disponibles para transformar una solución en otra, junto con una regla de evaluación asociada" [6].

En el diagrama de la figura 2.2 mostramos una segunda clasificación dado el método utilizado para resolver un problema de optimización. En dicho diagrama se remarca el tipo de método empleado en esta tesis.

A continuación hablaremos brevemente del cómputo evolutivo. Esta área de la computación nos ofrece una amplia variedad de herramientas que pueden usarse como métodos de optimización directa (por ejemplo, en la literatura encontramos una versión de PSO que hace uso de mutación [9]) para aproximar el óptimo a un problema con un espacio de búsqueda complejo (p.ej., accidentado o altamente multimodal).

2.4. Nociones de cómputo evolutivo

El cómputo evolutivo es un área de la computación que se inspira en los principios de la evolución biológica para simularlos y utilizarlos como una herramienta para el aprendizaje y la optimización.

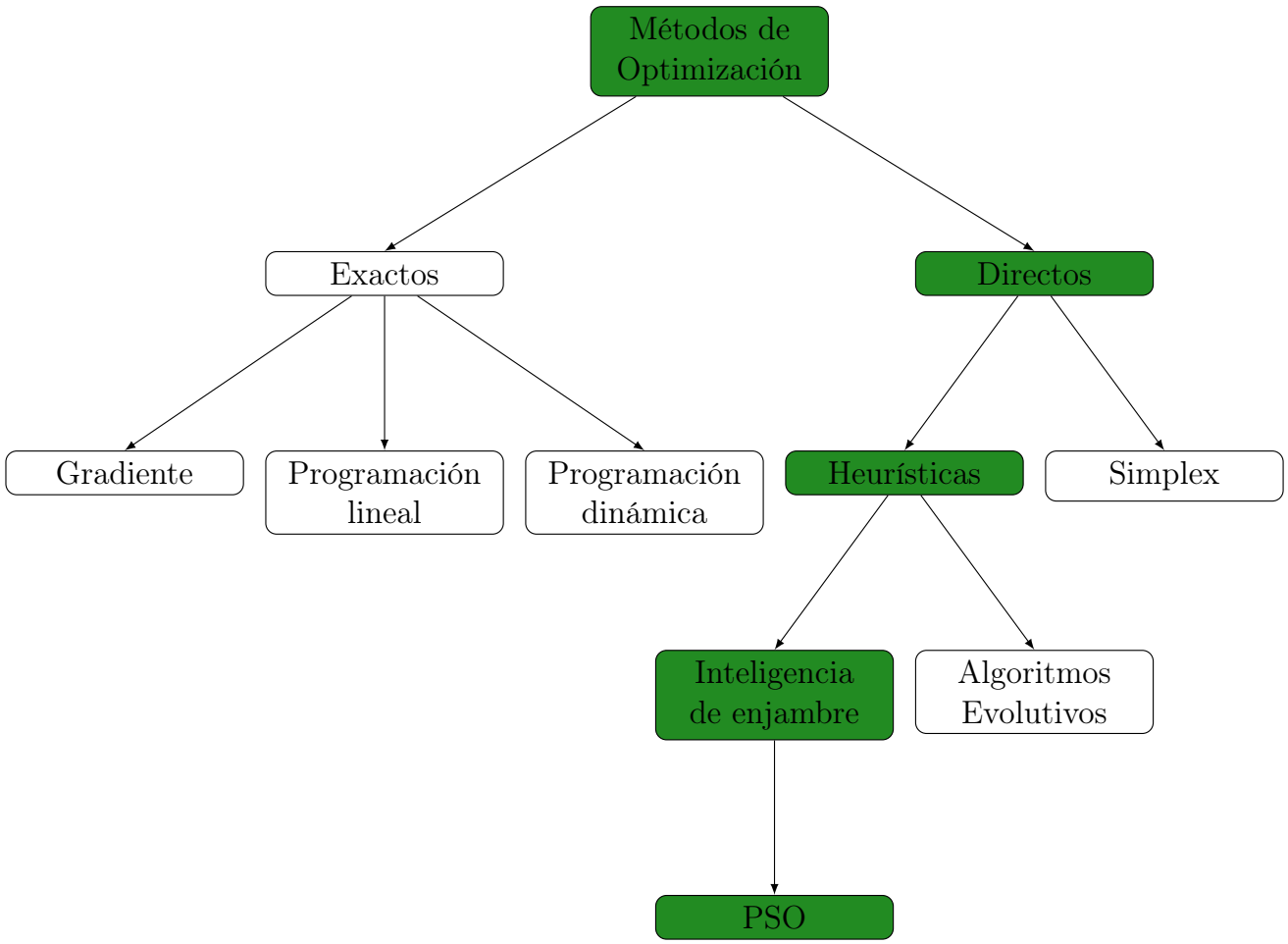

Figura 2.2: Diagrama de métodos de optimización

Los algoritmos evolutivos son especialmente útiles en dominios donde las soluciones óptimas no se pueden encontrar utilizando métodos deterministas o analíticos. De manera general, existen tres paradigmas principales que se utilizan en la computación evolutiva, los cuales son: las estrategias evolutivas, la programación evolutiva y los algoritmos genéticos [10]. Primero se explicará cada paradigma y luego describiremos los mecanismos principales que utilizan los algoritmos genéticos y que hacen posible su funcionamiento.

2.4.1. Programación evolutiva

Esta técnica fue propuesta por Lawrence J. Fogel [11] y se usó inicialmente para hacer evolucionar autómatas de estados finitos para que fueran capaces de predecir las secuencias futuras de símbolos que recibirían. Fogel usó una función de "pago" que indicaba qué tan bueno era un cierto autómata para predecir un símbolo, y usó un operador inspirado en la mutación para efectuar cambios en las transiciones y en los estados de los autómatas que tenderían a hacerlos más aptos para predecir secuencias de símbolos.
Esta técnica no consideraba el uso de un operador de recombinación sexual ya que su fin era modelar el proceso evolutivo a nivel de las especies y no a nivel de los individuos.
La programación evolutiva se aplicó originalmente a problemas de predicción, control automático,

identificación de sistemas y teoría de juegos, entre otros [12].

2.4.2. Estrategias evolutivas

Ingo Rechenberg desarrolló un método que le ayudara a resolver problemas de optimización en mecánica de fluidos. Particularmente, quería optimizar la geometría de un tubo, la minimización del arrastre de una placa de unión y la optimización estructural de una boquilla intermitente de dos fases.

Debido a la imposibilidad de describir y resolver estos problemas de optimización analíticamente o usando métodos tradicionales como el del gradiente [13], Ingo Rechenberg desarrolló un método de ajustes discretos aleatorios inspirado en el mecanismo de mutación que ocurre en la naturaleza [10].

2.4.3. Algoritmos genéticos

A principios de la década de los sesentas, John H. Holland, inspirado por los estudios realizados en aquella época con autómatas celulares [14] y redes neuronales [15], se percató de que el uso de reglas simples podría generar comportamientos flexibles, y visualizó la posibilidad de estudiar la evolución de comportamientos en un sistema complejo. Esto lo llevó a desarrollar una nueva técnica que denominó "planes reproductivos y adptativos", la cual utilizó para aprendizaje de máquina. Con el tiempo, esta técnica sería conocida como el algoritmo genético.

Un algoritmo genético requiere los componentes siguientes:

- Una representación de las soluciones potenciales del problema
- Una forma de crear una población inicial
- Una función de evaluación
- Operadores genéticos
- Valores para los diferentes parámetros que utiliza el algoritmo genético

El operador principal en los algoritmos genéticos es la cruza. La mutación es un operador secundario que, sin embargo, es necesario para mantener el espacio de búsqueda completamente conectado.

Hacemos énfasis especialmente en el operador de mutación ya que éste garantiza que sea posible explorar todo el espacio de búsqueda, y es posible adaptarlo en otros algoritmos, tales como el PSO [9].

2.4.4. Operador de cruza

La cruza es un operador que simula el proceso de reproducción sexual. La cruza crea descendientes con características de ambos padres y fomenta la diversidad de una población [10]. Las tres técnicas básicas de cruza para representación binaria son:

- Cruza de un punto

- Cruza de dos puntos
- Cruza uniforme

La cruza suele aplicarse con una frecuencia que va del 60 % al 100 %.

2.4.5. Cruza usando números reales

En su versión original, el algoritmo genético utilizó una codificación binaria (es decir, todas las variables del problema se convierten a números binarios). Con el tiempo, se usaron otros tipos de representación, tales como los números reales. Cuando utilizamos números reales en nuestra representación, es deseable definir operadores de cruza más acordes que puedan "romper" un cierto valor real, de manera análoga a como la cruza ordinaria "rompe" segmentos de cromosoma al usarse representación binaria. Las técnicas más usadas para lograrlo, son las siguientes [10]:

- Intermedia
- Aritmética simple
- Aritmética total
- Simulated Binary Crossover
- Cruza basada en dirección

2.4.6. Mutación para representación binaria

La mutación es un operador que se utiliza con menos frecuencia que la cruza, Se suelen recomendar porcentajes de mutación entre 0.001 y 0.01 para representaciones binarias [10].

Pese a que el operador de mutación se suele usar con menos frecuencia que el de cruza, se ha sugerido que el usar porcentajes altos de mutación al inicio de la búsqueda y luego decrementarlos exponencialmente favorece el desempeño global del algoritmo genético [16].

2.4.7. Mutación para representación real

Si se utilizan números reales para representar las variables del problema, se debe tomar en cuenta esto para diseñar operadores de mutación que sean análogos a los usados con la representación binaria. El operador de mutación juega un papel clave en la variabilidad de la población de un algoritmo evolutivo [10]. Algunas técnicas de mutación para representación real son:

- No Uniforme
- De Límite
- Uniforme
- Mutación basada en parámetros

2.5. Optimización mediante cúmulos de partículas

2.5.1. Fundamentos socio-cognitivos del PSO

Los sistemas naturales como los enjambres de abejas, las colonias de hormigas y los bancos de peces, muestran comportamientos colectivos emergentes que permiten que el grupo pueda resolver problemas de manera eficiente sin la necesidad de un control centralizado. Esto se conoce como inteligencia de enjambre [17]

Los algoritmos de inteligencia de enjambre utilizan como base el proceso de adaptación cultural. Éste comprende dos componentes: uno de alto nivel que se refiere a la capacidad de los individuos para formar patrones y resolver problemas a nivel de grupo (por ejemplo, la creación de normas culturales o comportamientos colectivos que surgen de la interacción entre individuos). El segundo componente, de bajo nivel, se asocia a los comportamientos individuales que son universales, probablemente innatos de cada individuo y que de igual forma influyen en el grupo. Estos últimos consisten en tres principios [18]:

- Evaluar
- Comparar
- Imitar

A continuación describiremos brevemente cada uno de ellos.

2.5.1.1. Evaluar

Quizás la característica de comportamiento más común en los organismos vivos es la tendencia a evaluar los estímulos que perciben. El aprendizaje no ocurre a menos que un organismo pueda evaluar las características del entorno que le son favorables, con respecto a las que no lo son. En el PSO esta característica se cumple evaluando una función que describe el problema a resolver.

2.5.1.2. Comparar

En 1954, Leon Festinger postuló una teoría social [19] en la que propone que los individuos tienden a evaluarse a sí mismos, comparándose con los demás. Algunas de las predicciones generadas por dicha teoría no han sido confirmadas por investigación empírica, ya que las comparaciones sociales son más complejas de lo que Festinger predijo (por ejemplo, Fisteger predice que los individuos siempre se comparan con individuos similares, pero otras investigaciones dicen que los individuos también se comparan con otros muy distintos). Dichas predicciones han servido como base en la inteligencia de enjambre. En dichos modelos, un agente evalúa su aptitud y se compara con todo el cúmulo para establecer un líder que dirija la trayectoria del cúmulo hacia la mejor solución posible dentro de un espacio de búsqueda.

2.5.1.3. Imitar

Konrad Lorenz [20] menciona que muy pocos animales son capaces de una imitación real. De hecho, la mayoría de los animales no imitan en el sentido estricto del término, sino que, a menudo, aprenden a través de otros mecanismos, como el condicionamiento o la observación. Sin embargo,

no logran el nivel de imitación compleja que se observa en los humanos y algunas aves. La imitación como la entendemos es principalmente humana [21]. En el PSO, el mejor individuo guía la búsqueda, es decir los otros individuos tratan de imitar su trayectoria.

2.5.2. Optimización mediante cúmulos de partículas

El algoritmo de optimización mediante cúmulos de partículas (PSO por sus siglas en inglés) fue propuesto en 1995 por James Kennedy y Russell Eberhart [22] y está inspirado en el comportamiento social de ciertos grupos de animales. Este comportamiento es simulado a través de un modelo en el que un grupo de agentes, en vez de intentar resolver un problema en sí, solo siguen reglas simples planteadas al inicio del algoritmo. A su vez, las reglas no dicen nada acerca de la existencia de un problema, sino que éste se resuelve a través de la influencia social recíproca, con base en el modelo de cultura adaptativa [17]. De tal forma, los agentes son capaces de mejorar características como la aptitud.

El algoritmo comienza inicializando de forma aleatoria (con distribución uniforme) la velocidad y posición de n agentes (a los cuales llamamos partículas). La posición de una partícula i en el espacio de búsqueda representa una posible solución al problema y la velocidad de una partícula i determina cómo cambiará su posición. Luego, se define un número de iteraciones máximo con una condición de parada establecida por el usuario. En cada iteración se evaluará la función objetivo para cada partícula y se actualizarán las posiciones y velocidades para cada una de ellas. Para simplificar lo anterior, podemos ver la figura 2.3 que muestra una representación del movimiento y de la posición de una partícula i en un plano bidimensional que tiende a moverse hacia el mejor global y hacia el mejor personal.

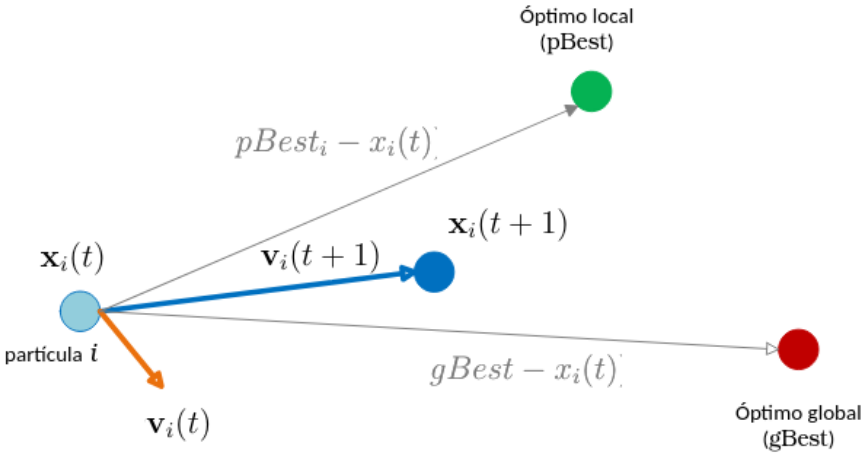

Figura 2.3: Representación esquemática del movimiento de una partícula i (en el algoritmo de PSO) que se desplaza hacia el mejor global y el mejor local

Básicamente, el algoritmo de PSO va ajustando las trayectorias de cada partícula para explorar el espacio de búsqueda en función de la posición de $pBest$ y $gBest$. El movimiento de la partícula

i en el cúmulo consta de dos componentes principales: uno que atrae a la partícula hacia la mejor posición del cúmulo $gBest$ y un segundo componente que la atrae a su mejor ubicación histórica $pBest$. Al mismo tiempo, tiende a moverse de forma aleatoria [18] como lo muestra la ecuación (2.1).

$$\mathbf{v}_i(t+1) = \omega \mathbf{v}_i(t) + c_1 r_1(\mathbf{pBest}_i - \mathbf{x}_i(t)) + c_2 r_2(\mathbf{gBest} - \mathbf{x}_i(t)) \tag{2.1}$$

donde:

- ω es el factor de inercia. Un valor distinto de cero hace que la partícula no altere su dirección de movimiento, Es una especie de desplazamiento o sumador lineal aplicado a la velocidad de la partícula. Cuando su valor es grande, tiende a mover a la partícula más lejos, con el objetivo de explorar regiones alrededor de la partícula (es común aplicarlo al comienzo). Cuando el valor es pequeño, la partícula busca mejores soluciones en un área más cercana al valor anterior de ésta (es común aplicarlo al final).

- c_1 es el coeficiente de aprendizaje personal. Este parámetro se relaciona con el comportamiento individual. Cuanto mayor sea este parámetro, la influencia de la mejor posición personal será más grande. Esto favorece la exploración individual, evitando la agrupación prematura del cúmulo.

- c_2 es el coeficiente de aprendizaje global. Este parámetro se relaciona con el comportamiento social. Cuanto mayor sea este parámetro, la influencia de la mejor posición global será más grande. Esto favorece la exploración colectiva, ayudando a la exploración del cúmulo.

- r_1 y r_2 son números aleatorios (agregan aleatoriedad al comportamiento de las partículas) con una distribución uniforme, en el intervalo [0,1].

- **pBest** es la mejor posición conocida por la partícula i (mejor posición personal).

- **gBest** es la mejor posición conocida por el cúmulo (mejor posición global).

Para identificar mejor el cúmulo, podemos asignar el símbolo vectorial algebraico $\mathbf{x_i}$ para representar la posición de cualquier cantidad de partículas i. Además, cada vector puede tener cualquier dimensión d. A la velocidad le asignamos el símbolo vectorial $\mathbf{v_i}$.

Una vez que tenemos la velocidad de la partícula i en el tiempo $t+1$, obtenemos su nueva posición. La actualización de la posición de una partícula i en el tiempo $t+1$ está dada por la ecuación (2.2).

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \tag{2.2}$$

El algoritmo de PSO se puede aplicar en espacios de búsqueda continuos [22] y discretos [23]. Sin embargo, es más común utilizarlo para resolver problemas de optimización en espacios de búsqueda continuos, es decir, en problemas en los cuales todas las variables son números reales.

En este trabajo utilizaremos la versión continua del algoritmo de PSO. La motivación para ello es la precisión que nos puede llegar a proporcionar en las soluciones, además de que es sencillo implementarlo en un lenguaje de alto nivel como C. Por otra parte, la motivación principal para utilizar este método de optimización es su simplicidad con relación a otros métodos como los algoritmos genéticos. Esto es debido a que el principio de funcionamiento del algoritmo base (ver el algoritmo 1) no requiere operadores como la cruza o la mutación, por lo que podría utilizarse en aplicaciones en las que los recursos de cómputo disponibles son muy limitados.

Algoritmo 1 Optimización mediante cúmulos de partículas

1: Inicializar posiciones $\mathbf{x}_i$ y velocidades $\mathbf{v}_i$ de las partículas aleatoriamente (distribución uniforme) en el espacio de búsqueda para cada partícula i.
2: Inicializar la mejor posición conocida de cada partícula (mejor personal) $pBest_i = \mathbf{x}$
3: Inicializar la mejor posición global $gBest$ como la mejor posición entre todas las partículas.
4: **Mientras** no se cumpla el criterio de parada **hacer**
5: **Para cada** cada partícula i **hacer**
6: Actualizar la velocidad de la partícula i:

$$\mathbf{v}_i(t+1) = \omega \mathbf{v}_i(t) + c_1 r_1 (pBest_i - \mathbf{x}_i(t)) + c_2 r_2 (gBest - \mathbf{x}_i(t))$$

7: Actualizar la posición de la partícula i:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1)$$

8: Evaluar la nueva posición $\mathbf{x}_i(t+1)$.
9: **Si** $f(\mathbf{x}_i(t+1)) < f(pBest_i)$ **entonces**
10: Actualizar la mejor posición personal:

$$pBest_i = \mathbf{x}_i(t+1)$$

11: **Fin Si**
12: **Si** $f(\mathbf{x}_i(t+1)) < f(gBest)$ **entonces**
13: Actualizar la mejor posición global:

$$gBest = \mathbf{x}_i(t+1)$$

14: **Fin Si**
15: **Fin Para cada**
16: **Fin Mientras**

Cabe aclarar que por su naturaleza, el PSO en su versión original no permite garantizar convergencia al óptimo global, sino únicamente a la mejor partícula en el cúmulo [2].

Trabajos más recientes han podido demostrar convergencia global de algunas variantes del PSO bajo ciertas condiciones iniciales y para ciertas clases de problemas (ver por ejemplo [24][25][26][27]).

Además, aunque en este trabajo no utilizaremos la versión discreta de PSO, hacemos una breve mención del funcionamiento de ambas versiones.

2.5.2.1. Funcionamiento del modelo discreto del PSO

El principio de funcionamiento en la versión discreta es el mismo que en la versión continua descrita en la sección anterior. En [23] se describe una reformulación del algoritmo para operar con variables discretas (en específico, números binarios). Pero ahora las trayectorias se interpretan como cambios en la probabilidad de que una coordenada tome un valor de cero o uno.

Esta versión puede ser utilizada en problemas donde se requieren soluciones en espacios discretos. De igual forma, es común reformular problemas de punto flotante en términos binarios y resolverlos en un espacio numérico discreto.

En un espacio binario se puede considerar que una partícula se mueve al cambiar diversos números de bits. Por lo tanto, la velocidad de la partícula en general puede describirse como el número de bits cambiados por iteración o la distancia de Hamming entre la partícula en el tiempo t y en $t+1$. Por ejemplo, una partícula con cero bits invertidos no se mueve, mientras que al invertir todas sus coordenadas binarias, una partícula se mueve a la posición más lejana.

La velocidad se define en términos de cambios en las probabilidades de que un bit esté en un estado u otro. Esto implica que una partícula se mueve en un espacio de estados restringido a 0 y 1 para cada dimensión d. Entonces, cada v_{id} representa la probabilidad de que el bit x_{id} tome el valor de 1. Por ejemplo, si $v_{id} = 0.20$, entonces hay un 20 % de probabilidad de que x_{id} sea 1 y un 80 % de probabilidad de que sea 0.

El término $(pBest_{id} - x_{id})$ puede tener valores -1, 0 ó 1 y se utiliza para ponderar el cambio en la probabilidad v_{id} en $t+1$. Entonces, $pBest$ y x_{id} son enteros en $0, 1$ y v_{id} al ser una probabilidad, debe estar restringida al intervalo $[0, 1]$. Para lograr esta última modificación, se puede utilizar una transformación (sigmoidal) $s(v_{id})$.

Para obtener la posición se utiliza la siguiente condición:

```
1  if (rand() < s(V_id)) //rand() es pseudo aleatorio en [0.0,1.0] con distribucion
       uniforme.
2  then X_id = 1;
3  else
4  X_id = 0;
```

Código 2.1: Criterio de decisión para actualizar la posición X_{id} en el PSO discreto

Con relación a la velocidad máxima que puede presentar el PSO discreto, una V_{max} alta (por ejemplo 10.0) hace que sea menos probable probar nuevos vectores ya que reduce el rango de respuesta de $s(v_{id})$. Por lo tanto, parte de la función de v_{max} es establecer un límite para la exploración adicional una vez que la población converge. Cabe señalar también que, mientras que un valor alto de v_{max} en la versión con valores continuos aumenta el rango explorado por una partícula, en la versión binaria ocurre lo contrario: un v_{max} más pequeño permite una mayor tasa de mutación.

2.5.2.2. Funcionamiento del modelo continuo del PSO

En esta versión de PSO se utilizan números reales R^d donde d es el número de dimensiones de un vector, al que llamamos partícula y que puede ser representada como un punto en un espacio multidimensional. En una población de puntos, nótese que al estar en un espacio de soluciones continuo, los individuos pueden ser más parecidos entre sí (tienen mayor precisión). Por otra parte, el cambio en la posición de cada individuo a lo largo del tiempo t se ve afectado por la velocidad de éste y la posición que tenga en un tiempo anterior $t-1$. Dependiendo de cómo sea la comunicación entre los integrantes del cúmulo, puede haber una rápida convergencia o una tendencia a la exploración del espacio de búsqueda por parte del grupo. No obstante, los individuos tenderán a moverse unos hacia otros (ver figura 2.4), para influenciarse mutuamente, a medida que los individuos buscan acuerdos con sus vecinos.

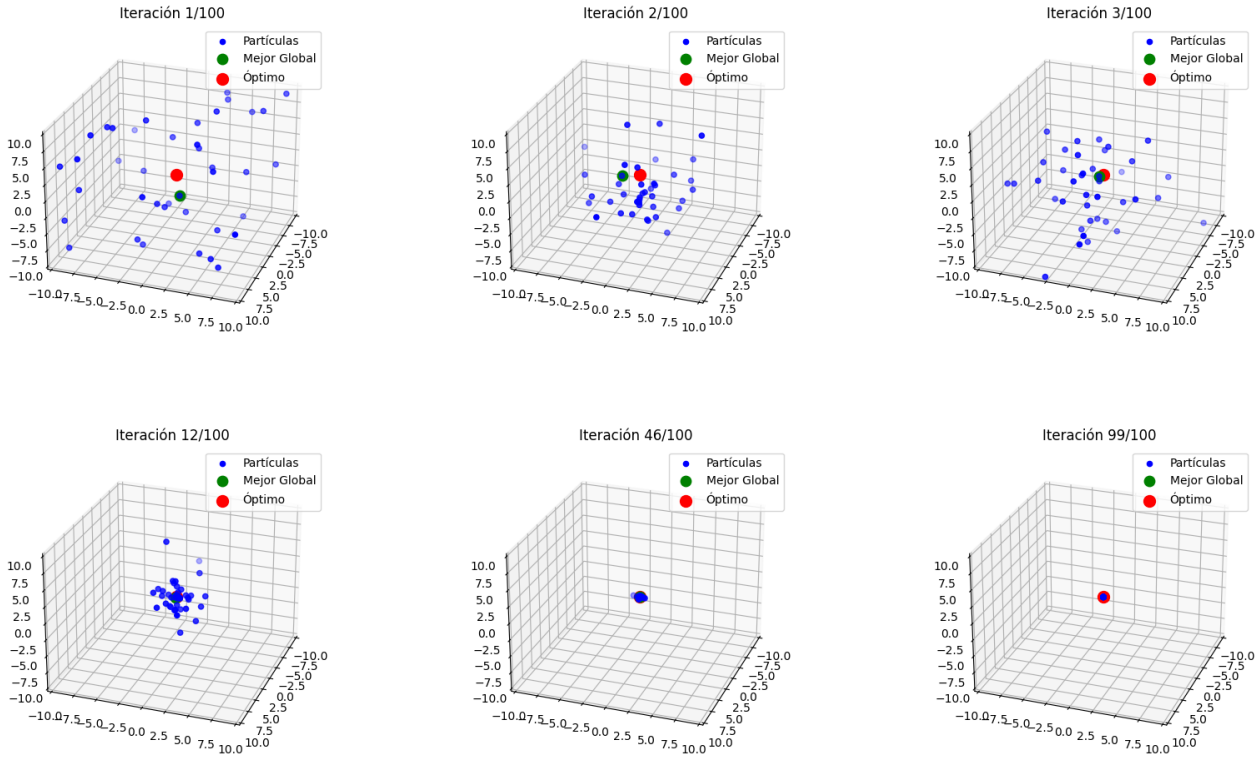

Figura 2.4: Funcionamiento del algoritmo de PSO utilizando 40 individuos y tres dimensiones en la función esfera usando 100 iteraciones

La velocidad es un vector de números reales que se suma a las coordenadas de posición para mover la partícula a una posición $x_i(t+1)$. La actualización de la posición de una partícula i en el tiempo $t+1$ está dada por la ecuación (2.2) .

Como se puede deducir, cada individuo se verá influenciado por su propio comportamiento anterior y por los éxitos de sus vecinos, quienes, no necesariamente son individuos cercanos a él, en el espacio de los parámetros. Sus vecinos son aquellos que están cerca en un espacio topológico (hablaremos acerca de las topologías en la siguiente sección), o estructura establecida para que los individuos puedan socializar o compartir conocimiento, sin importar la distancia que exista entre ellos.

La dirección del movimiento (bajo el concepto de locomoción de Lewin [17]) está en función de la posición y velocidad actual, así como por ubicación del éxito previo más grande del individuo y la mejor posición encontrada por cualquier miembro del vecindario, la cual podemos definir de forma general como la función:

$$x_i(t+1) = f(x_i(t), v_i(t), lBest_i, gBest) \tag{2.3}$$

PSO utiliza el modelo *gbest* que conecta conceptualmente a todos los miembros de la población entre sí. El efecto de esto es que cada partícula está influenciada por el mejor rendimiento de cualquier miembro de toda la población. El segundo modelo, llamado *pbest*, crea un vecindario para cada individuo que comprende a sí mismo y a sus k vecinos más cercanos en la población. Por ejemplo, si $k = 2$, entonces cada individuo i estará influenciado por el mejor rendimiento entre un grupo formado por las partículas $i-1$, i e $i+1$. Las diferentes topologías de vecindario pueden resultar en diferentes tipos de efectos. La velocidad se obtiene de la diferencia entre la mejor posición previa del individuo y su posición actual, así como de la diferencia entre la mejor posición del vecindario y la posición actual del individuo (ver línea 6 del algoritmo 1 en la que se utiliza la ecuación (2.1)).

Una parte importante a tomar en cuenta en el algoritmo de PSO es controlar las trayectorias que pueden tomar las partículas a lo largo de las iteraciones. Si no se realiza esto, pueden expandirse en ciclos cada vez más amplios hasta llegar a salirse del espacio de búsqueda válido. Para evitar esto se puede aplicar algún método de constricción [24]. Cabe aclarar que la velocidad máxima (V_{max}) que debe alcanzar una partícula está acotada a fin de mantener su movimiento dentro de un rango útil [17]. Además, el valor del parámetro depende de cierto conocimiento del problema para evitar quedar atrapado en un óptimo local o al acercarse a un valor prometedor reducirlo para dar pasos más pequeños.

2.5.3. Topologías para PSO

Como ya hemos mencionado anteriormente, un grupo se ve afectado por la estructura de la red social que lo contiene [17]. En la investigación sobre enjambres de partículas, normalmente se utilizan redes en donde se da una interacción de los individuos con sus vecinos inmediatos y de todos los individuos con el mejor desempeño dentro de la población. Sin embargo, se puede utilizar cualquier otra red que ayude al algoritmo a encontrar mejores resultados. En [28] se indica que una topología adecuada puede mitigar la convergencia prematura y mejorar la eficiencia del algoritmo [29].

Algunas de las topologías más comúnmente utilizadas [17][28][30] [31] se describen a continuación.

2.5.3.1. Topología totalmente conectada

En este tipo de interacción, cada partícula tiene acceso a la mejor solución global encontrada por todo el cúmulo. Esta topología puede conducir a una convergencia prematura en problemas complejos, como los multimodales.

2.5.3.2. Topología de anillo

En este tipo de interacción, las partículas solo interactúan con un subconjunto de vecinos cercanos. Es adecuada para problemas con múltiples óptimos.

2.5.3.3. Topología en estrella

En este tipo de interacción, todas las partículas se conectan a una partícula que actúa como líder central.

2.5.3.4. Topología en malla

En este tipo de interacción, las partículas solo interactúan con vecinos directos, formando una especie de malla.

En la figura 2.5 se representan los tipos de topologías más comunes.

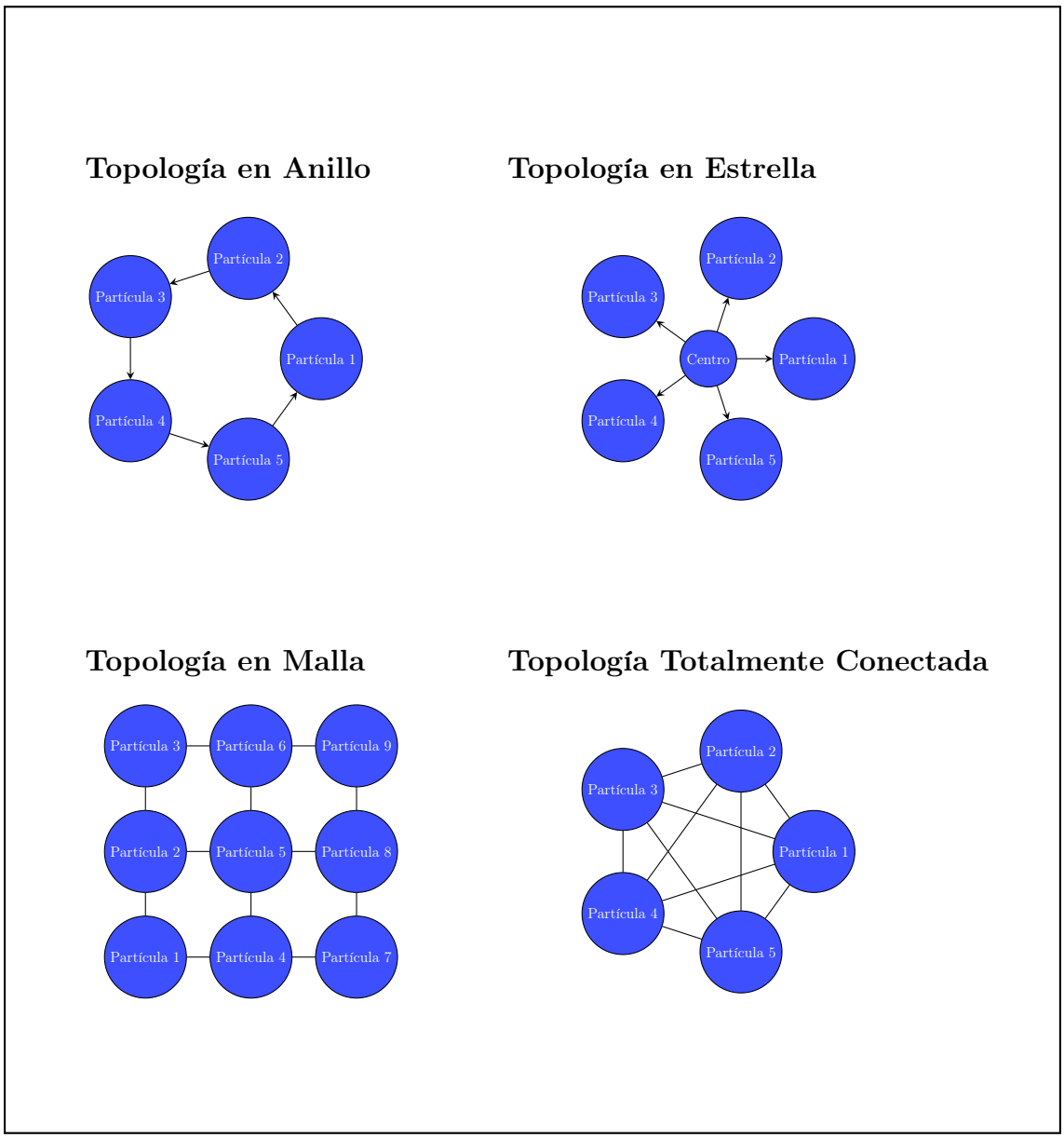

Figura 2.5: Diferentes topologías de partículas para el PSO

2.5.4. Ventajas del PSO

- Simplicidad de su implementación: El algoritmo de PSO es fácil de entender y programar. Tiene una estructura simple que permite adaptarlo rápidamente a diferentes problemas sin necesidad de recurrir a implementaciones complejas.

- Es un método de búsqueda directa: A diferencia de otros métodos de optimización, PSO no necesita calcular derivadas de las funciones, lo que lo hace apto para problemas no lineales o discontinuos o cuando las derivadas no están disponibles.

- Capacidad de Evitar Mínimos Locales: Gracias a su enfoque basado en múltiples partículas que exploran simultáneamente diferentes partes del espacio de búsqueda, PSO tiene una

mayor capacidad para evitar quedarse atrapado en mínimos locales.

- Fácilmente paralelizable: Las partículas se mueven de manera independiente, lo que facilita la implementación paralela de PSO. Esto es ideal para aprovechar arquitecturas de cómputo distribuidas o multi núcleo, lo que permite acelerar el proceso de optimización.
- Adaptabilidad: El algoritmo de PSO se puede ajustar fácilmente para adaptarse a diferentes tipos de problemas y dominios. Se pueden modificar los parámetros y las funciones de evaluación para mejorar su rendimiento en distintas aplicaciones.

2.5.5. Desventajas de PSO

- Convergencia prematura: En problemas con paisajes de búsqueda complejos, PSO puede converger prematuramente a una solución subóptima si todas las partículas se acercan rápidamente a un mismo punto sin explorar el espacio de búsqueda de forma adecuada.
- Dependencia de parámetros: El rendimiento de PSO es altamente sensible a la configuración de sus parámetros, tales como los coeficientes de aceleración, la inercia y el límite de velocidad. Encontrar la combinación adecuada de estos parámetros puede ser complicado.
- Escalabilidad en problemas de alta dimensionalidad: PSO funciona bien en problemas de pequeña a mediana escala, pero su rendimiento tiende a disminuir en problemas de alta dimensionalidad, donde el tamaño del espacio que debe explorarse es inmenso y la probabilidad de encontrar una buena solución disminuye considerablemente.
- Oscilación de partículas: Si no se controlan adecuadamente, las partículas pueden oscilar o moverse indefinidamente sin converger a un punto óptimo, especialmente en problemas donde las soluciones óptimas están cercanas pero la inercia y la aceleración están mal ajustadas.
- Dificultad para problemas con restricciones complejas: PSO puede ser menos eficiente en aquellos problemas donde las restricciones son difíciles de manejar.

Capítulo 3

Alta dimensionalidad

La alta dimensionalidad es un fenómeno que se presenta cuando un problema matemático tiene un número elevado de variables de decisión. Del mismo modo, esta situación se presenta en áreas en las que se manejan grandes cantidades de datos [32] o se busca una solución a problemas con un gran número de características, por ejemplo, en aprendizaje automático, procesamiento de señales, análisis de datos, optimización, etc.
La alta dimensionalidad se puede interpretar como un espacio euclidiano $\mathbb{R}^d$ representado como $x_1, x_2, x_3, ..., x_D$ (un vector $\mathbf{x}$), donde el número de variables D es muy grande (por ejemplo, mayor a 100).
Vale la pena comentar, que en alta dimensionalidad, las propiedades geométricas del espacio cambian de forma radical. Por ejemplo, en un espacio de alta dimensionalidad cambia la forma en la que se distribuyen las distancias a medida que D aumenta. En ese caso, la mayoría de los puntos (aleatorios, con distribución uniforme e independientes) tienden a estar a la misma distancia con respecto de un punto de referencia cualquiera (ver figura 3.1).

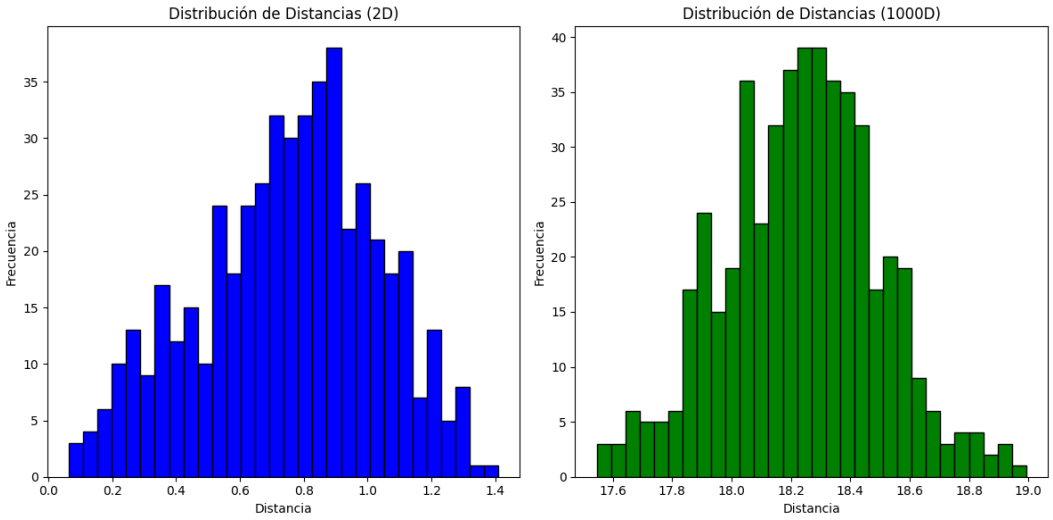

Figura 3.1: El histograma 1000D muestra que la mayoría de las distancias están agrupadas cerca de un valor medio, mientras que en 2D (histograma de la derecha) la distribución de distancias muestra mayor variabilidad. Para este ejemplo se generaron 1000 puntos aleatorios con una distribución uniforme en el espacio y se calcularon las distancias euclidianas de cada punto respecto al origen para cada caso.

Otro aspecto a considerar es que en alta dimensionalidad, la densidad de los puntos baja drásticamente. Es decir, la distancia entre los puntos aumenta (se dispersan) ya que hay más espacio por cubrir (ver figura 3.2).

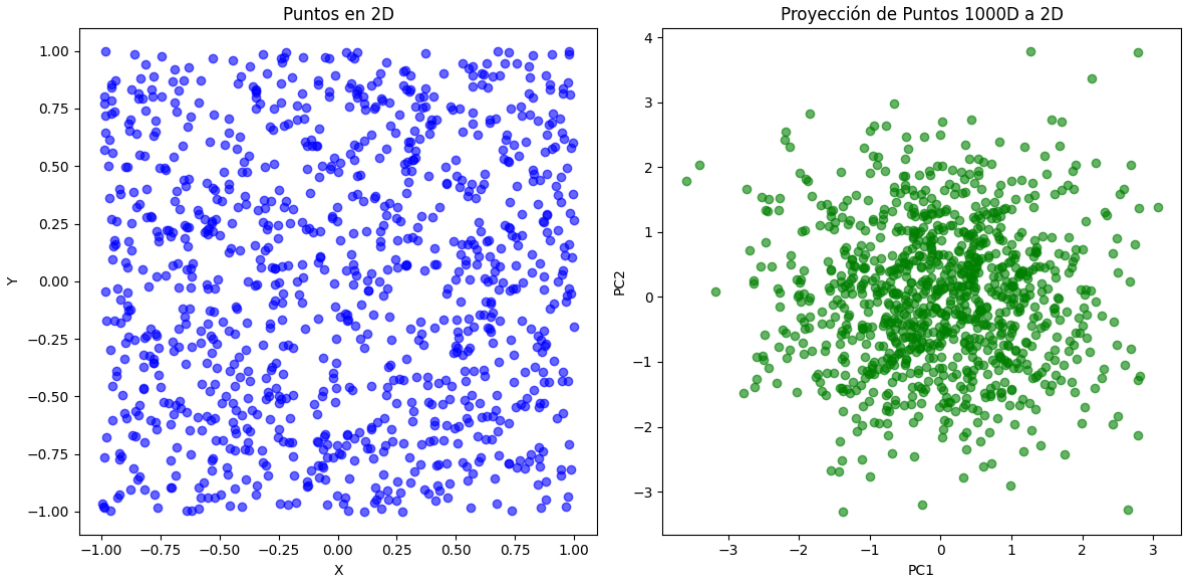

Figura 3.2: Se muestran las gráficas de 1000 puntos generados aleatoriamente con distribución uniforme para 2D y 1000D. Para el caso de 1000D se utilizó la técnica de análisis de componentes principales (*Principal Component Analysis*) para visualizarlos en 2D.

En las siguientes secciones explicaremos los efectos de la alta dimensionalidad en los algoritmos de optimización estocásticos, en especifico, para el algoritmo de PSO. Además, mencionaremos algunas estrategias que intentan hacer frente a los problemas asociados a este hecho. Cabe aclarar que solo algunos de estos métodos serán utilizados en este trabajo de tesis.

3.1. Maldición de la dimensionalidad

En 1957, Richard Bellman [33] abordó la complejidad de los problemas de decisión y optimización donde el número de posibles decisiones es extremadamente grande. En 1961 publicó un artículo donde describe "la maldición de la dimensionalidad", en el cual, describe el problema que causa el aumento exponencial del volumen asociado con la adición de dimensiones extra a un espacio euclidiano.
Por ejemplo, en R^D (donde D es la dimensionalidad), el volumen de un cubo con lado 2 es 2^D más grande que el volumen del cubo unidad [34], a pesar que los lados de los cubos solo difieren por un factor de 2, como se puede ver en la figura 3.3.

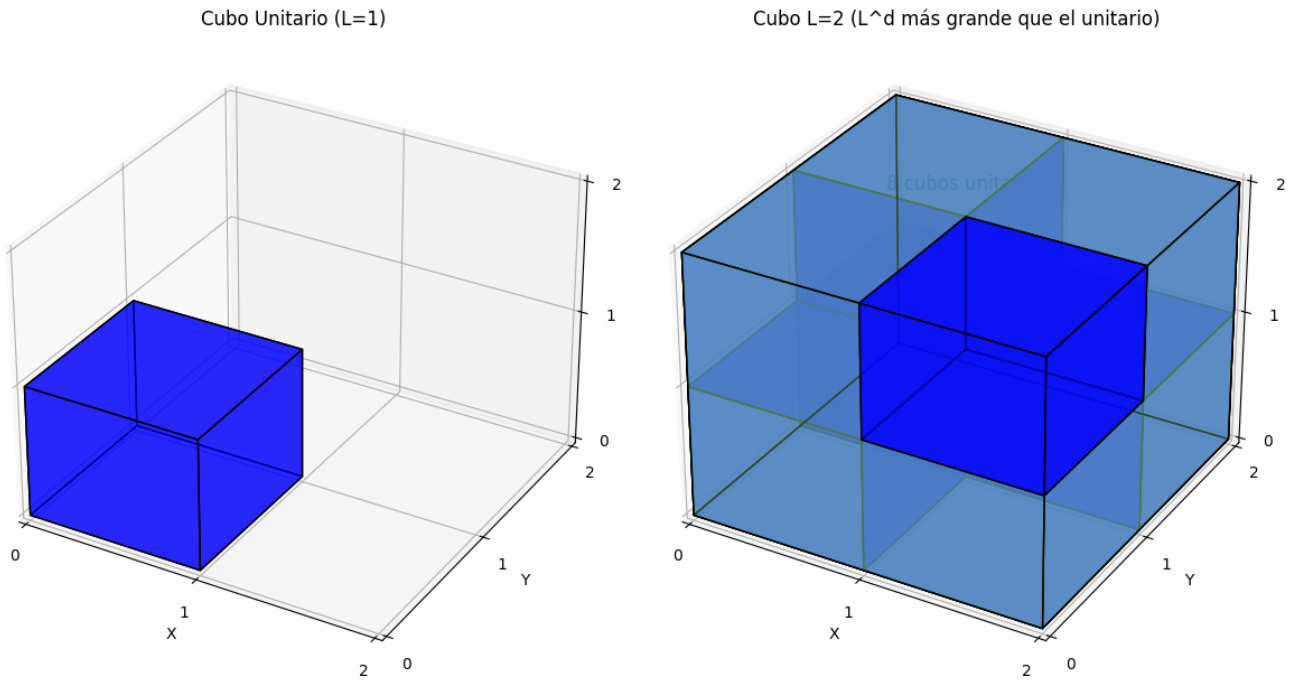

Figura 3.3: A la derecha se muestra un cubo unitario. Al aumentar su longitud solo una unidad, su volumen se incrementa ocho veces (2^3 veces) como lo muestra la segunda figura

3.1.1. Maldición de la alta dimensionalidad en PSO

PSO es un algoritmo fácil de entender y sencillo de implementar. Sin embargo cuando se trabaja con altas dimensiones (arriba de 100), se presenta la llamada "maldición de la dimensinalidad", ya que conforme aumentan las dimensiones del problema a resolver, el espacio de búsqueda crece exponencialmente y los datos se vuelven más dispersos (ver figura 3.2). Esto provoca que se vuelva complicado explorar de forma eficiente dicho espacio (en la figura 3.1 podemos ver que las distancias en altas dimensiones son mayores que en bajas, lo que significa un aumento en la dispersión). En consecuencia, aumenta el riesgo de caer en mínimos locales, sin mencionar el incremento en la complejidad computacional, debido al aumento del número de evaluaciones de la función objetivo realizadas para buscar una buena solución.

3.2. Estrategias en PSO para manejar alta dimensionalidad

Existen distintas estrategias que se pueden utilizar para intentar mitigar los efectos negativos de la alta dimensionalidad en el algoritmo de PSO, a fin de mejorar su rendimiento. Las principales se describen a continuación.

3.2.1. Reducción de dimensionalidad por análisis de componentes principales

El análisis de componentes principales (*Principal Component Analysis*) es un método estadístico que se utiliza para reducir la dimensionalidad de un conjunto de datos. Esto lo hace eliminando las variables redundantes o irrelevantes y manteniendo la mayor cantidad de información posible.

Básicamente, funciona encontrando componentes que maximizan la varianza, y proyectando los datos originales sobre estos componentes [35].

Los objetivos que se busca resolver al aplicar este método son [36]:

- Extraer la información más importante de los datos.
- Comprimir el tamaño del conjunto de datos manteniendo solo esta información importante.
- Simplificar la descripción del conjunto de datos.
- Analizar la estructura de las variables.

Por ejemplo, en la figura 3.2 se utilizó PCA (por sus siglas en inglés) para proyectar 1000 puntos aleatorios en 2D para visualizarlos y poder comparar algunos comportamientos como la reducción de la densidad de puntos contra un espacio en 2D.

Este método no es aplicable en esta tesis, ya que se trata de una técnica lineal y la mayoría de los problemas que vamos a resolver son "no lineales". Además, se tendrían que realizar cálculos de covarianza de datos y eso resultaría altamente costoso.

3.2.2. Adaptación de parámetros

Esta técnica consiste en ajustar de forma inicial o dinámicamente los parámetros del PSO (por ejemplo, la velocidad o la inercia) para mejorar el equilibrio entre exploración y explotación en un espacio de alta dimensionalidad [4]. Es decir, el ajuste controlado de estos parámetros puede disminuir la velocidad de las partículas e intentar encontrar una mejor solución cuando se encuentren en una zona prometedora y por el contrario, incrementarla en la etapa de exploración. Además, se puede limitar la velocidad para evitar que las partículas salgan del espacio de búsqueda válido.

Con relación a los coeficientes de aprendizaje c_1 y c_2 también se pueden modificar para incrementar la dominancia cognitiva o social del individuo.

Así mismo, se puede modificar el valor de la inercia (ω), para que las partículas exploren al principio (con mayor inercia) y exploten cuando se acercan a una solución prometedora.

Por ejemplo, en [4] se mencionan diferentes técnicas para adaptar los parámetros de control del PSO, que van desde el peso de inercia ω propuesto por Shi y Eberhart [37], pasando por una implementación de un sistema difuso para adaptar dinámicamente ω [16], hasta un mecanismo de control más reciente de múltiples parámetros para cambiar adaptativamente ω, c_1 y c_2 en [38]. Para el presente trabajo, se realiza un ajuste en tiempo real de estos parámetros, tomando como referencia el comportamiento de las evaluaciones de la función de aptitud con relación al espacio de búsqueda.

3.2.3. Uso de topologías

Esta técnica tiene como principio el hecho de que la comunicación dentro de un grupo se ve afectada por la estructura de la red social [17]. En [2], por ejemplo, se utilizan estructuras sociales simples, como la interacción de los individuos con sus vecinos inmediatos y la interacción de todos los individuos con el individuo de mejor desempeño en la población. Sin embargo, se podría utilizar cierto número de integrantes u otro tipo de estructuras sociales, que afecten el intercambio de información de forma positiva para alcanzar un mejor consenso que convenga a todos.

Como vimos en el capítulo anterior, existen diferentes topologías como la de anillo o estrella, las cuales, pueden ayudar a explorar de forma más eficiente un espacio de búsqueda [39].

También se puede modificar la topología cuando se presente una circunstancia determinada, para aprovechar la propiedad de ésta. Por ejemplo, en lugar de usar la estructura de cúmulo global (donde todas las partículas interactúan entre sí) siempre, se puede optar por una topología local en la que las partículas solo interactúan con sus vecinos más cercanos para evitar la exploración innecesaria en una iteración, o incluso se puede hacer formando subcúmulos de partículas. Esto último se realiza en [40], en donde distintas partes del espacio de búsqueda son optimizadas de forma independiente por diferentes subcúmulos, para luego reagruparse y configurar una nueva vecindad con base en la información compartida de cada subcúmulo.

En [41], se desarrolló un PSO completamente informado, donde la actualización de cada partícula se basa en las posiciones de varios vecinos. También tenemos el Comprehensive Learning PSO (CLPSO) introducido en [42], en el cual las partículas actualizan cada dimensión aprendiendo de diferentes posiciones locales óptimas.

3.2.4. Métodos híbridos

El algoritmo de PSO puede combinarse con otros algoritmos de optimización para mejorar su rendimiento en alta dimensionalidad. Por ejemplo, se puede combinar con algoritmos genéticos o solo con algunos operadores como el de mutación para ayudar a mantener la diversidad en el cúmulo, evitando que las partículas converjan demasiado rápido a soluciones subóptimas.

Se puede utilizar un buscador local combinado con PSO (como el Recocido Simulado o un algoritmo basado en el gradiente de la función) para mejorar la explotación [43]

Otra técnica consiste en descomponer el problema en partes más pequeñas, para optimizar cada parte por separado para luego combinar los resultados. Un ejemplo de este tipo de esquema es la coevolución cooperativa, en la cual se divide el espacio de búsqueda en varios grupos y se aplica PSO a cada grupo de manera cooperativa. Es decir, cada subcúmulo optimiza solo un subconjunto de las variables, y las interacciones entre los subcúmulos permiten encontrar soluciones globales [44].

Para esta tesis, no utilizamos esta técnica, ya que no mezclamos la versión base que tomamos como referencia para nuestra mejora [4] con ninguna otra versión de PSO o con algún operador de algoritmos genéticos. Solo intentamos mejorar el equilibrio entre diversidad y convergencia a través de la información de la posible forma del espacio de búsqueda que la partícula aporta al explorarlo.

3.2.5. Estrategias de aprendizaje

Se puede hacer uso de aprendizaje adquirido en una etapa previa de optimización en un espacio de menor dimensión que sirve como un preprocesamiento que mejora la eficiencia del algoritmo al transferir el conocimiento sobre los patrones de búsqueda exitosos a la fase de alta dimensionalidad [45]. Se debe tomar en cuenta que esta estrategia es útil cuando se tiene acceso a varios problemas similares; por ejemplo, el mismo problema en distintas dimensionalidades. Además, el éxito de la estrategia depende de la calidad del aprendizaje transmitido, por lo que incluso puede perjudicar a la solución principal.

Otra estrategia consiste en utilizar múltiples estrategias de búsqueda adaptativas que ayuden a mejorar la exploración y explotación del espacio de soluciones que se adapten a las necesidades de éste en cada momento [3].

En nuestro trabajo, la única estrategia de aprendizaje que se utiliza es la propuesta en el algoritmo de referencia que se adoptó como base para el algoritmo desarrollado en esta tesis [4].

3.2.6. Aleatoriedad controlada

Este método consiste en introducir elementos de aleatoriedad de forma controlada en algunos parámetros del PSO, permitiendo que las partículas exploren diferentes regiones del espacio de búsqueda sin tener que recorrer todas las dimensiones de manera exhaustiva. Esto mejora la exploración en espacios de búsqueda muy grandes (este método ya es utilizado en el algoritmo de PSO original, en donde se introduce aleatoriedad para los componentes cognitivo y social) [2] [46].

En general, existe una gran variedad de estrategias o métodos que pueden mitigar los efectos negativos de la alta dimensionalidad en el PSO. Todas las estrategias tienen sus limitaciones según las características del problema a resolver. Sin embargo, no hay restricciones de uso, siempre y cuando preserven los principios socio-cognitivos del algoritmo.

3.3. Algoritmos para resolver problemas de optimización a gran escala

Como hemos mencionado, la popularización de los algoritmos de gran escala se ha visto reflejada en el desarrollo de algoritmos que incorporan distintas técnicas que buscan encontrar la mejor solución posible. A continuación se mencionan algunos ejemplos de estos algoritmos.

3.3.1. Optimizador de Cúmulos Competitivos para Optimización a Gran Escala

El algoritmo CSO (*A Competitive Swarm Optimizer for Large Scale Optimization*) [4] fue propuesto en el 2015, por Ran Cheng y Yaochu Jin. Los autores se inspiraron en la versión original de PSO de Kennedy y Eberhart [2], con la diferencia de que en este algoritmo no se utiliza el mejor personal ($pBest$) ni el mejor global ($gBest$) para actualizar el movimiento de las partículas. En lugar de esto, se introduce un mecanismo de competencia por pares, con el cual, la partícula perdedora actualiza su velocidad y posición aprendiendo de la ganadora.

A continuación se explica más detalladamente cómo funciona el algoritmo:

- Se inicializa de forma aleatoria (como en el algoritmo original de PSO) un cúmulo m de n partículas. Luego, se actualiza de manera iterativa. Cada partícula tiene una posición de D dimensiones $\mathbf{x}_i(t) = (x_{i,1}(t), x_{i,2}(t), \ldots, x_{i,n}(t))$ y un vector de velocidad de D dimensiones, $\mathbf{v}_i(t) = (v_{i,1}(t), v_{i,2}(t), \ldots, v_{i,n}(t))$.

- En cada generación (t), el cúmulo se revuelve y se asignan parejas (se asume que el tamaño del cúmulo m es un número par). Luego se realiza una competencia entre las dos partículas que forman el par.

- Como resultado de cada competencia, la partícula con una mejor aptitud, denominada "ganadora", será pasada directamente a la siguiente generación del cúmulo $(t + 1)$

- La partícula que pierde la competencia, denominada "perdedora", actualizará su posición y velocidad aprendiendo de la ganadora.

- Después de aprender de la ganadora, la partícula perdedora también será pasada a la siguiente generación del cúmulo $(t + 1)$

- El algoritmo termina cuando se llegue a un máximo de evaluaciones ($maxFE$) de la función de aptitud

Cada partícula participará en una competencia solo una vez, es decir, para un cúmulo m ocurren $m/2$ competencias y la velocidad y posición de las $m/2$ partículas serán actualizadas. La figura 3.4 ilustra este funcionamiento.

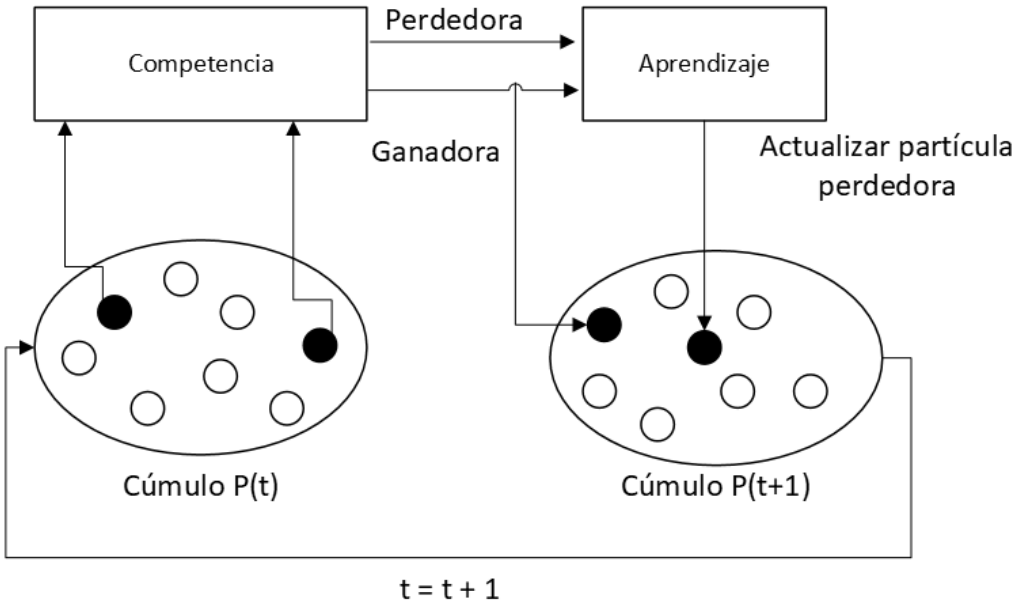

Figura 3.4: La competencia inherente al algoritmo fomenta la exploración en las primeras etapas y la explotación más refinada en etapas posteriores del algoritmo CSO.

La velocidad del perdedor se actualizará utilizando la estrategia de aprendizaje de la ecuación (3.1)

$$\mathbf{v}_{l,k}(t+1) = R_1(k,t)\mathbf{v}_{l,k}(t) + R_2(k,t)(\mathbf{x}_{w,k}(t) - \mathbf{x}_{l,k}(t)) + \phi R_3(k,t)(\bar{x}_k(t) - \mathbf{x}_{l,k}(t)) \tag{3.1}$$

donde:

- $\mathbf{x}_{w,k}(t)$ Es la posición del ganador en la k-ésima ronda de competencia en la generación t
- $\mathbf{x}_{l,k}(t)$ Es la posición del perdedor en la k-ésima ronda de competencia en la generación t
- $\mathbf{v}_{w,k}(t)$ Es la velocidad del ganador en la k-ésima ronda de competencia en la generación t
- $\mathbf{v}_{l,k}(t)$ Es la velocidad del perdedor en la k-ésima ronda de competencia en la generación t
- $R_1(k,t)$, $R_2(k,t)$, y $R_3(k,t)$ Son factores aleatorios $\in [0,1]^d$
- ϕ Es un parámetro de aprendizaje que controla la influencia de $\bar{x}_k(t)$
- $\bar{x}_k(t)$ Es el valor medio de la posición de las partículas relevantes. El control de vecindario puede ayudar a mejorar el rendimiento del PSO en funciones multimodales al mantener un mayor grado de diversidad en el cúmulo [47]. Se puede adoptar una versión global y una versión local:

- $\bar{x}_k^g(t)$ Denota la posición media global de todas las partículas en el cúmulo

- $\bar{x}_{l,k}^l(t)$ Significa la posición media local de las partículas en un vecindario predefinido de la partícula l.

La posición del perdedor se puede actualizar ahora con la nueva velocidad, utilizando la ecuación (3.2),

$$\mathbf{x}_{l,k}(t+1) = \mathbf{x}_{l,k}(t) + \mathbf{v}_{l,k}(t+1). \tag{3.2}$$

Para el parámetro $\bar{x}_k(t)$ se suele introducir el control de vecindario para aumentar la diversidad del cúmulo, lo que potencialmente mejora el rendimiento de búsqueda del CSO (se adopta como configuración predeterminada $\bar{x}_k(t)$).

La primera parte de la ecuación (3.1) asegura la estabilidad del proceso de búsqueda y es similar al término de inercia ω en el PSO original. También puede interpretarse como que $\omega = 1$ y se agrega un vector aleatorio $R_1(t)$.

La segunda parte es la componente cognitiva. En esta parte de la ecuación se da el aprendizaje de la partícula perdedora con respecto a la ganadora en lugar de hacerlo de la mejor personal $pBest$.

La tercera parte es el componente social en donde la partícula perdedora aprende de la posición media del cúmulo actual $\bar{x}_k(t)$ en lugar del mejor global $gBest$.

De forma general, podemos decir que el algoritmo del CSO (ver algoritmo 2) es sencillo y no muy distinto a la versión original del PSO (ver algoritmo 1).

Algoritmo 2 Algoritmo del CSO

Inicializar una población m de n partículas con posiciones aleatorias $\mathbf{x_i}$ y velocidades $\mathbf{v_i}$ en el espacio de búsqueda.
Evaluar la aptitud de cada partícula $f(\mathbf{x}_i)$ con base en la función objetivo.
Mientras no se cumpla el criterio de parada **hacer**
    Emparejar las partículas aleatoriamente en $m/2$ pares.
    Para cada cada par $(\mathbf{x_i}, \mathbf{x_j})$ **hacer**
        Si $f(\mathbf{x}_i) < f(\mathbf{x}_j)$ **entonces** ▷ Para problemas de minimización
            Marcar $\mathbf{x}_i$ como el ganador y $\mathbf{x}_j$ como el perdedor.
        else
            Marcar $\mathbf{x}_j$ como el ganador y $\mathbf{x}_i$ como el perdedor.
        Fin Si
        Reincorporar la partícula ganadora al cúmulo
        Actualizar la velocidad del perdedor usando la ecuación (3.1)
        Actualizar la posición del perdedor usando la ecuación (3.2)
        Borrar o sobreescribir $\mathbf{x_i}$ y $\mathbf{x_j}$;
    Fin Para cada
    Reincorporar la partícula actualizada al cúmulo
Fin Mientras
Devolver la mejor solución encontrada $\mathbf{x}^* = \arg\min f(\mathbf{x}_i)$.

3.3.2. Un algoritmo de optimización mediante cúmulos de partículas con aprendizaje social para optimización escalable

El SL-PSO (*A social learning particle swarm optimization algorithm for scalable optimization*) fue desarrollado por Ran Cheng y Yaochu Jin en 2014 [5]. En esta versión los autores utilizan mecanismos de aprendizaje social en un PSO. Cada partícula aprende de cualquier otra partícula que sea mejor (denominada "demostradora") en el cúmulo actual. Además, se simplifica la configuración de parámetros, ya que se adopta un método de control de parámetros dependiente de la dimensionalidad.

A continuación se detalla la forma en que funciona este algoritmo (ver figura 3.5):

- Se inicializa un cúmulo $P(t)$ de m partículas (m es el tamaño del cúmulo y t es el índice de la generación). Para cada partícula i en $P(t)$, se tiene al igual que en el PSO original, un vector $x_i(t)$ inicializado aleatoriamente.
- A cada partícula se le asigna un valor de aptitud calculado a partir de la función objetivo $f(X)$.
- El cúmulo se ordena de forma ascendente (según la aptitud de cada individuo). Posterior-

mente, cada partícula (excepto aquella con el mejor valor de aptitud) corregirá sus comportamientos aprendiendo de aquellas partículas (demostradoras) que tengan mejores valores de aptitud.

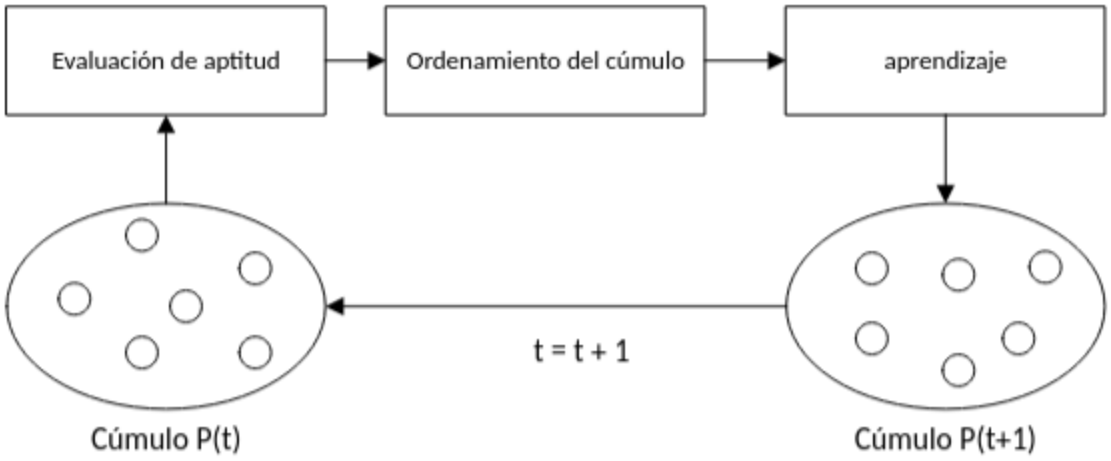

Figura 3.5: El cúmulo se ordenará según la aptitud de las partículas y todas, excepto la mejor, se actualizarán aprendiendo de cualquier partícula cuya aptitud sea mejor. El aprendizaje individual es un proceso de ensayo y error, mientras que el aprendizaje social aprovecha mecanismos como la imitación, el refuerzo y el condicionamiento

Este algoritmo tiene como base la imitación. Sus autores la describen como un proceso en el cual un imitador copia parte de un comportamiento de un demostrador a través de la observación. Por tanto, los componentes más importantes en SL-PSO son el ordenamiento del cúmulo y el aprendizaje de comportamientos. También se debe tomar en cuenta lo siguiente:

- En cada generación, una partícula podría servir como demostrador para diferentes imitadores más de una vez.
- En un cúmulo ordenado, para cualquier imitador (partícula i, donde $1 \leq i < m$), su demostrador puede ser cualquier partícula k que cumpla con $i < k \leq m$.
- Por ejemplo, para la partícula 1, las partículas 2, 3, ..., m pueden ser sus demostradores, mientras que para la partícula $(m-1)$, solo la partícula m puede ser su demostrador. Como resultado, la partícula 1 (la peor) nunca puede ser un demostrador y la partícula m (la mejor) nunca será un imitador. Es decir, la mejor partícula en el cúmulo actual no se actualizará.

Un imitador aprenderá los comportamientos de diferentes demostradores utilizando la ecuación (3.3).

$$X_{i,j}(t+1) = \begin{cases} X_{i,j}(t) + \Delta X_{i,j}(t+1), & \text{si } p_i(t) \leq P_i^L, \\ X_{i,j}(t), & \text{en caso contrario,} \end{cases} \tag{3.3}$$

donde:

- P_i^L Es una probabilidad de aprendizaje para cada partícula i. En [5] se asume que cuanto mayor sea la dimensionalidad del problema, más difícil será resolverlo y, por lo tanto, menos probable será que una partícula esté dispuesta a aprender de otras. Por ende, se recomienda una relación inversamente proporcional entre la probabilidad de aprendizaje y la dimensionalidad del problema como lo muestra la ecuación (3.4).

$$P_i^L = \left(1 - \frac{i-1}{m}\right)^{\alpha \cdot \log\left(\lceil \frac{d}{m} \rceil\right)} \tag{3.4}$$

  donde:

  - m es el tamaño del cúmulo. Se define como una función de la dimensionalidad de búsqueda como lo muestra la ecuación (3.5).

$$m = M + \frac{j \cdot d}{k} \tag{3.5}$$

    donde:

    - M es el tamaño base del cúmulo, necesario para que el algoritmo SL-PSO funcione adecuadamente. En [5] utilizan $M = 100$, (un tamaño pequeño de cúmulo suele ser suficiente para problemas de optimización unimodales, mientras que un tamaño mayor es necesario para problemas de optimización multimodales, a fin de permitir una exploración más intensiva [48, 49]).

  - $1 - \frac{i-1}{m}$ indica que la probabilidad de aprendizaje es inversamente proporcional al índice de la partícula i en un cúmulo ordenado. Es decir, cuanto mayor sea la aptitud de una partícula, menor será su probabilidad de aprendizaje.
  - $\alpha \cdot \log\left(\lceil \frac{d}{m} \rceil\right)$ indica que la probabilidad de aprendizaje es inversamente proporcional a la dimensionalidad de búsqueda (se mantendría una mejor diversidad del enjambre para problemas a gran escala debido a la tasa de aprendizaje reducida). $\alpha \cdot \log(\cdot)$ se utiliza para suavizar la influencia de $\frac{d}{m}$. Empíricamente [5], recomiendan que el coeficiente $\alpha < 1$. En particular, utilizaron $\alpha = 0.5$.

- p_i es una probabilidad generada aleatoriamente.
- $X_{i,j}(t)$ es la j-ésima dimensión del vector de comportamiento de la partícula i en la generación t, con $i \in \{1, 2, 3, \ldots, m\}$ y $j \in \{1, 2, 3, \ldots, d\}$.

- $\Delta X_{i,j}(t+1)$ es la corrección del comportamiento en $t+1$. Ver la ecuación (3.6).

$$\Delta X_{i,j}(t+1) = r_1(t) \cdot \Delta X_{i,j}(t) + r_2(t) \cdot I_{i,j}(t) + r_3(t) \cdot \epsilon \cdot C_{i,j}(t) \tag{3.6}$$

donde:

- $\Delta X_{i,j}(t)$ actúa de forma similar al componente de inercia del PSO original.
- $I_{i,j}(t)$ es el componente de imitación. Es equivalente a la parte cognitiva del PSO original, solo que en lugar de aprender de *pbest*, la partícula i aprende de cualquiera de sus demostradoras como lo muestra la ecuación (3.7). Específicamente, el j-ésimo elemento en el vector de comportamiento de la partícula i, $X_{i,j}(t)$, imita a $X_{k,j}(t)$, que es el j-ésimo elemento en el vector de comportamiento de la partícula k (demostradora de la partícula i). Nótese que $i < k \leq m$, y k se genera independientemente para cada elemento j. Ver la figura 3.6.

$$I_{i,j}(t) = X_{k,j}(t) - X_{i,j}(t) \tag{3.7}$$

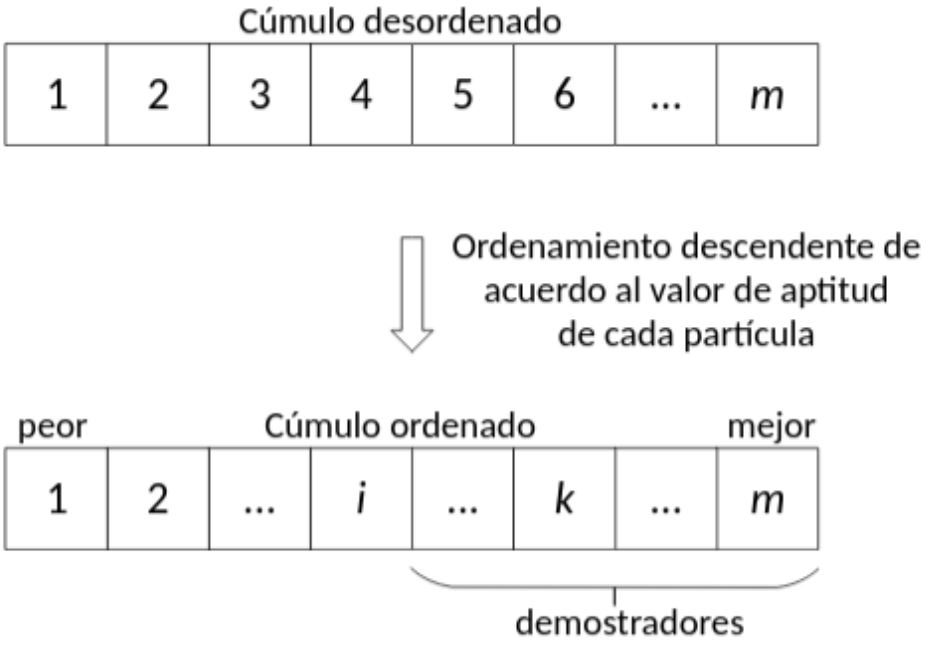

Figura 3.6: El cúmulo se ordena según los valores de aptitud (lo hacemos en orden descendente ya que se está minimizando). Después, cada partícula (excepto la mejor) aprende de sus demostradores, los cuales tienen mejores valores de aptitud.

- $C_{i,j}(t)$ es el componente de influencia social (ver ecuación (3.9)). La partícula i no aprende de *gbest*. En realidad, lo hace del comportamiento colectivo de todo el cúmulo, es decir, del comportamiento promedio de todas las partículas en el cúmulo actual,

denotado por la ecuación (3.8).

$$\bar{X}_j(t) = \frac{1}{m}\sum_{i=1}^{m} X_i(t) \tag{3.8}$$

además,

$$C_{i,j}(t) = \bar{X}_j(t) - X_{i,j}(t) \tag{3.9}$$

- ϵ es el *factor de influencia social*, (ver ecuación (3.10)) y es proporcional a la dimensionalidad del problema.

$$\epsilon = \beta \times \frac{n}{M} \tag{3.10}$$

donde:

  - β =0.01

- (ω, c_1 y c_2) son reemplazados por los coeficientes $r_1(t)$, $r_2(t)$ y $r_3(t)$, que se generarán aleatoriamente dentro del intervalo $[0, 1]$.

,

El algoritmo 3, describe de forma general el funcionamiento del SL-PSO.

Algoritmo 3 Algoritmo del SL-PSO

$t = 0$
$M = 100$, $\alpha = 0.5$, $\beta = 0.01$
$m = M + \frac{j \cdot d}{10}$
$\epsilon = \beta \times \frac{d}{M}$
Para cada $i = 1$ **to** m **hacer**
    inicializar de forma aleatoria X_i en el cúmulo
    $P_i^L = \left(1 - \frac{i-1}{m}\right)^{\alpha \cdot \log\left(\lceil \frac{d}{m} \rceil\right)}$
Fin Para cada
/*comienza ciclo como en la figura 3.5*/
Mientras no es satisfecha la condición de parada **hacer**
    Para cada $i = 1$ **to** m **hacer**
        $F_i = f(X_i(t))$ /*$f(X)$ es la función objetivo*/
    Fin Para cada
    actualizar la mejor solución X^*
 aprendizaje:*/
    ordenar el cúmulo de acuerdo al valor de la aptitud en F
    Para cada $i = 1$ **to** $m - 1$ **hacer**
corregir el comportamiento de $X_i(t)$
        $p_i(t) = \text{randr}(0, 1)$ /*randr(a, b) genera un número real aleatorio entre a y b*/
        Si $p_i(t) \leq P_L^i$ **entonces**
            Para cada $j = 1$ **to** n **hacer**
                $k \leftarrow \text{rand}(i + 1, m)$ /*rand(A, B) genera un número entero aleatorio entre A y B*/
                $\Delta X_{i,j}(t + 1) = r_1(t) \cdot \Delta X_{i,j}(t) + r_2(t) \cdot I_{i,j}(t) + r_3(t) \cdot \epsilon \cdot C_{i,j}(t)$
                $X_{i,j}(t + 1) = X_{i,j}(t) + \Delta X_{i,j}(t + 1)$
            Fin Para cada
        Fin Si
    Fin Para cada
    $t = t + 1$
Fin Mientras
Salida: X^*

3.3.3. Optimización mediante cúmulos de partículas con aprendizaje de múltiples estrategias para problemas de optimización a gran escala

El algoritmo MSL-PSO (*Multiple-strategy learning particle swarm optimization for large-scale optimization problems*), fue desarrollado en 2018 por Hao Wang [3]. En dicho trabajo, se verifica la efectividad del algoritmo en la resolución de problemas de optimización a gran escala para el conjunto de problemas del CEC´2008 utilizando 100, 500 y 1000 dimensiones y problemas del CEC´2010 con 1000 dimensiones en comparación a algoritmos como el SL-PSO y CSO. En el

MSL-PSO se adopta la idea del aprendizaje social del SLPSO [5] para actualizar la posición de cada individuo en la población. Sin embargo, se agregan dos etapas con diferentes estrategias de aprendizaje.

La primera etapa se utiliza para mejorar la capacidad de exploración. Cada individuo explora aprendiendo de los demostradores que tienen mejor desempeño. Para esto, el cúmulo se ordenará del peor al mejor individuo y cada uno explorará diferentes posiciones aprendiendo de sus demostradores y de la posición media de la población actual. Todas las mejores posiciones de entre todas las exploradas por su individuo correspondiente, serán parte de una nueva población temporal. La nueva población temporal se ordenará en orden descendente según las aptitudes y será utilizada por cada individuo para encontrar sus demostradores, basándose en el rango de la mejor solución explorada en la población temporal y el rango del individuo en la población actual. Se espera que la segunda etapa equilibre la convergencia y la diversidad de la población, para poder actualizar la velocidad y la posición de cada individuo

A continuación describimos a mayor detalle el funcionamiento del algoritmo:

- Se inicializa una población Pop y se obtiene la aptitud inicial de cada individuo.
- Luego se establece una condición de paro del algoritmo. Mientras no se cumpla esta condición, el proceso se seguirá repitiendo.
- Todos los individuos se ordenarán según su aptitud en orden descendente. Es decir, los valores más altos (peores) quedarán al principio, mientras que los menores quedarán al final de la lista (esto es así porque se está minimizando el problema).
- Cada individuo explorará K_{max} posiciones aprendiendo de sus demostradores y de la posición media de la población actual siguiendo la idea del SL-PSO [5]. Para esto se utiliza el algoritmo 5, el cual hace la exploración de 1 a K_{max} posiciones y a su vez recorre las D dimensiones de cada posición. Se selecciona aleatoriamente un demostrador para actualizar la velocidad en la d-ésima dimensión utilizando la ecuación (3.11). Al finalizar el recorrido de las D dimensiones se generará la k-ésima posición candidata utilizando la ecuación (3.12). Cada una de estas K_{max} posiciones tienen poca probabilidad de ser iguales. Esto agrega diversidad y la oportunidad de encontrar una mejor solución.

$$vv_{id}^{k} = r_1^k \cdot v_{id}(t) + r_2^k \cdot (x_{jd}(t) - x_{id}(t)) + \phi \cdot r_3^k \cdot (\bar{x}_d(t) - x_{id}(t)) \tag{3.11}$$

  donde:

  - K_{max} es el número máximo de pruebas para cada individuo ($k = 1, 2, \ldots, K_{max}$).
  - vv_{id}^{k} es la velocidad actualizada (con aprendizaje) de la partícula i en la d-ésima dimensión, utilizando el demostrador k-ésimo ($\mathbf{VV}_i^k = (vv_{i1}^k, vv_{i2}^k, \ldots, vv_{iD}^k)$).
  - $v_{id}(t)$ es la velocidad del individuo i en el tiempo t ($\mathbf{V}_i(t) = (v_{i1}, v_{i2}, \ldots, v_{iD})$).

- es la media o posición promedio del cúmulo en el tiempo t, como se indica en la ecuación (3.8) del SL-PSO.
- t, r_1^k, r_2^k y r_3^k son números aleatorios generados uniformemente en el rango [0, 1] en la k-ésima iteración.
- ϕ es la probabilidad de aprendizaje social utilizada para definir el grado de aprendizaje a partir de la posición media de la población.

$$xx_{id}^k = x_{id}(t) + vv_{id}^k \tag{3.12}$$

- xx_{id}^k es la posición actualizada (con aprendizaje) de la partícula i en la d-ésima dimensión, utilizando el demostrador k-ésimo ($\mathbf{XX}_i^k = (vv_{i1}^k, vv_{i2}^k, \ldots, XX_{iD}^k)$).
- $x_{id}(t)$ es la posición del individuo i en el tiempo t ($\mathbf{X}_i(t) = (x_{i1}, x_{i2}, \ldots, x_{iD})$).

Al finalizar el ciclo de K_{max} se evalúa la aptitud de cada una de las posiciones candidatas $\mathbf{XX}_i^k$ y se elige la partícula con la mejor aptitud. Ésta será guardada y denotada con x_c para su posición y v_c para su velocidad.

- Al finalizar el recorrido de NP tendremos una nueva población $NPop$ compuesta de las mejores posiciones probadas de cada individuo en Pop, que ahora se ordenarán de forma descendente.

- Luego se realiza la segunda etapa. En ésta, los demostradores de cada individuo se seleccionan de dos subconjuntos de $NPop$ ($NPop = x_{c1}, x_{c2}, \ldots, x_{nNP}$) para actualizar la velocidad y posición de cada individuo en Pop como lo muestra la ecuación (3.13) y (3.14).

$$v_{id}(t+1) = r_1 \cdot vc_{id}(t) + r_2 \cdot (xc_{jd} - xc_{id}(t)) + \phi \cdot r_3 \cdot (xc_{kd} - xc_{id}(t)) \tag{3.13}$$

donde:

- v_{ci} es la velocidad de la mejor posición explorada para el individuo i ($v_{ci} = (v_{ci1}, v_{ci2}, \ldots, v_{ciD})$).
- j y k representan a los dos demostradores en $NPop$ de los cuales el individuo i aprende en la dimensión d.
- ϕ es la probabilidad de aprendizaje social.

$$x_{id}(t+1) = xc_{id} + v_{id}(t+1) \tag{3.14}$$

donde:

- x_{ci}es la posición de la mejor posición explorada para el individuo i ($x_{ci} = (x_{ci1}, x_{ci2}, \ldots, x_{ciD})$).

En la figura 3.7 se puede observar un ejemplo de cómo se pueden seleccionar dos demostradores en la segunda etapa.

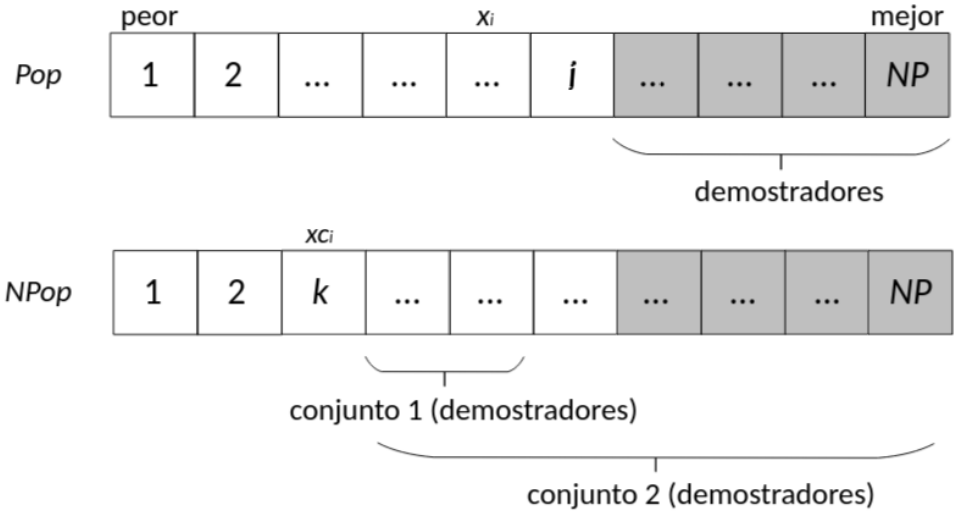

Figura 3.7: Un individuo i elige los demostradores para aprender. Para Pop elegirá del conjunto **demostradores**, mientras que para $Npop$ el individuo puede elegir del **conjunto 1** y del **conjunto 2**. El rango de xc_i en $NPop$ es k, y el rango de $x_i(t)$ en Pop es j. Si $k < j$, entonces los demostradores entre k y j se utilizarán para guiar al individuo a explotar una mejor solución. De lo contrario, cuando el rango de la mejor posición explorada de un individuo i en $NPop$ es mejor que el del individuo i en Pop, significa que la mejor posición explorada tiene un mejor rendimiento entre $NPop$ que el individuo i en Pop.

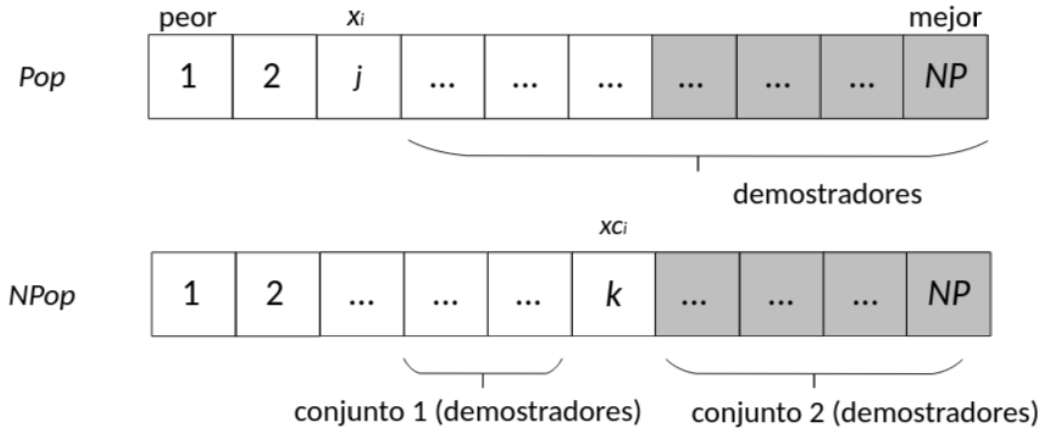

Figura 3.8: Para evitar una convergencia prematura, también aprendemos de algunos perdedores de la mejor solución explorada. Si $j < k$, los perdedores entre j y k serán seleccionados como uno de los demostradores. El otro demostrador se selecciona del **conjunto 2**, que está compuesto por todos los individuos que tienen mejor aptitud que la mejor posición explorada

El algoritmo 4 describe de forma general el funcionamiento del MSL-PSO. El algoritmo 5 des-

cribe la primera etapa de aprendizaje que proporciona la diversidad y el algoritmo 6 describe la segunda etapa para lograr el equilibrio entre diversidad y convergencia.

Algoritmo 4 Optimización de Cúmulo de Partículas con Aprendizaje de Múltiples Estrategias (MSL-PSO)

Entrada: Tamaño de la población NP, número máximo de evaluaciones de aptitud MAX_FES
Inicializar una población Pop
Evaluar la aptitud de cada individuo en Pop, $fes = NP$
Mientras $fes \leq MAX\_FES$ **hacer**
    Ordenar la población en orden descendente
    Para cada $i = 1$ to NP **hacer**
        Sondear K_{max} posiciones usando la técnica de aprendizaje social propuesta en [5] para el individuo i (ver Algoritmo 5)
        Evaluar estas K_{max} posiciones y mantener la mejor solución entre estas K_{max} soluciones
    Fin Para cada
    Ordenar la nueva población $NPop$, compuesta por la mejor posición sondeada de cada individuo en Pop, en orden descendente
    Para cada $i = 1$ to NP **hacer**
        Encontrar dos subconjuntos en la nueva población $NPop$ para el aprendizaje social del individuo i, y actualizar la población Pop (ver Algoritmo 6)
    Fin Para cada
    $fes = fes + (K_{max} + 1) \times NP$
Fin Mientras
Salida: La mejor solución y su valor de aptitud

Algoritmo 5 Sondeo de Posición

Entrada: Individuo i
Para cada $k = 1$ a K_{max} **hacer**
    Para cada $d = 1$ a D **hacer**
        Seleccionar aleatoriamente un individuo de sus demostradores
        Actualizar la velocidad en la dimensión d usando la ecuación (3.11)
    Fin Para cada
    Generar la posición candidata k-ésima usando la ecuación (3.12)
Fin Para cada
Evaluar los valores de aptitud de estas posiciones candidatas
$\mathbf{x}_{ci} = \arg\min\{f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_{K_{max}})\}$
Salida: La mejor posición sondeada $\mathbf{x}_{ci}$

Algoritmo 6 Actualización de posición

Entrada: La mejor posición probada del individuo i ($\mathbf{x}_{ci}$);

1: Actualizar la velocidad y posición utilizando las eqs. (3.13) y (3.14), respectivamente;
2: Evaluar el valor de aptitud (*fitness*) del individuo i;
3: **Si** $f(\mathbf{x}_i(t+1)) < f(g_{best})$ **entonces**
4: $\quad g_{best} = \mathbf{x}_i(t+1)$;
5: $\quad f(g_{best}) = f(\mathbf{x}_i(t+1))$;
6: **Fin Si**
7: **Salida:** la posición del individuo i en la generación $t+1$;

Capítulo 4

Optimización global de gran escala

Los adelantos tecnológicos de los últimos años han dejado al descubierto la necesidad de encontrar mejores soluciones a problemas de optimización más complejos. En este capítulo trataremos con el tipo de problemas que cuentan con un gran número de variables. Si bien el aumento en la capacidad de procesamiento de cómputo ha sido continuo, la "maldición de la dimensionalidad" continúa siendo una limitante al momento de buscar los mejores resultados. Por esta razón, a principios de este siglo se popularizó la optimización global de gran escala (*Large-Scale Global Optimization*, LSGO por sus siglas en inglés), la cual, a través de mecanismos novedosos, realiza una búsqueda mucho más eficiente para explorar la mayor cantidad de soluciones posibles, sin que esto impacte significativamente en la cantidad de tiempo requerida [50]. Eventos como el *IEEE Congress on Evolutionary Computation (CEC)*, la *Genetic and Evolutionary Computation Conference (GECCO)* y *Parallel Problem Solving from Nature (PPSN)*, entre otros, han ayudado a popularizar aún más la optimización global de gran escala.

Un primer objetivo de los eventos antes mencionados fue crear una referencia para la evaluación de la calidad en los algoritmos para optimización a gran escala, ya que un problema cumún que se presentaba, era que el trabajo existente estaba limitado a los problemas de prueba utilizados en estudios individuales. Sin embargo, esta desventaja se ha resuelto al proponer pruebas estándar diseñadas específicamente para probar el rendimiento de algoritmos de optimización de gran escala. Por ejemplo, el *2008 IEEE Congress on Evolutionary Computation (CEC'2008)* [51] fue el primer congreso donde se propuso un conjunto de problemas de referencia diseñados especialmente para optimización global de gran escala.

4.1. Definiciones utilizadas en los conjuntos de problemas para optimización de alta dimensionalidad

Antes de presentar los principales conjuntos de problemas especializados para probar los algoritmos basados en cúmulos de partículas para optimización de alta dimensionalidad, proporcionamos algunos conceptos relacionados con las características generales de las funciones utilizadas para tal objetivo.

Podemos expresar un problema de ecuaciones no lineales simultáneas como lo muestra la ecuación (4.1) donde, $\mathbb{R}^n$ denota el espacio euclidiano de n dimensiones. La ecuación (4.1) es la forma estándar de representar un sistema de n ecuaciones no lineales con n incógnitas, con la convención de que el lado derecho de cada ecuación es cero [52], y donde $\mathbf{x}_*$ es la variable con la que se alcanza el valor óptimo $\mathbf{F}(\mathbf{x}_*)$.

$$\text{Dado: } F : \mathbb{R}^n \longrightarrow \mathbb{R}^n, \quad \text{encontrar: } \mathbf{x}_* \in \mathbb{R}^n, \quad \text{para el cual } \mathbf{F}(\mathbf{x}_*) = 0 \in \mathbb{R}^n \tag{4.1}$$

Definición 1. Una función $f(\mathbf{x})$ es separable si y solo si:

$$\arg \min_{(x_1,\cdots,x_D)} f(x_1, \cdots, x_D) = \left(\arg \min_{x_i} f(x_1, \cdots), \cdots \arg \min_{x_D} f(\cdots, x_D) \right),$$

Es decir, una función de D variables es separable si puede reescribirse como una suma de D funciones de una sola variable. Si una función $f(x)$ es separable, sus parámetros x_i se llaman independientes [53].

Definición 2. Una función $f(x)$ es parcialmente separable con m subcomponentes independientes si y solo si:

$$\arg \min_{x} f(x) = \left(\arg \min_{x_1} f(x_1, \dots), \dots, \arg \min_{x_m} f(\dots, x_m) \right),$$

donde $\mathbf{x} = (x_1, \dots, x_D)^\top$ es un vector de decisión de D dimensiones, y $x_1, \dots, x_m$ son subvectores disjuntos de $\mathbf{x}$, con $2 \leq m \leq D$.

Como un caso especial de la definición 2, una función es completamente separable si los subvectores $x_1, \dots, x_m$ son unidimensionales (es decir, $m = D$).

Definición 3. Una función $f(\mathbf{x})$ es completamente no separable si cada par de sus variables de decisión interactúan entre sí.

Definición 4. Una función es parcialmente aditiva separable si tiene la siguiente forma general:

$$f(\mathbf{x}) = \sum_{i=1}^{m} f_i(\mathbf{x_i}),$$

donde x_i son vectores de decisión mutuamente excluyentes de f_i, $\mathbf{x} = (x_1, \ldots, x_D)^\top$ es un vector de decisión global de D dimensiones, y m es el número de subcomponentes independientes.

4.2. Conjuntos de problemas para optimización de alta dimensionalidad

4.2.1. Conjunto de problemas de prueba del CEC'2008

Tang et al. [54] diseñaron el conjunto de problemas del CEC'2008. Estos problemas fueron creados especificamente para probar algoritmos de optimización global para espacios de alta dimensionalidad, como parte de una sesión especial y una competencia que se centró en optimización a gran escala (de 100, 500 y 1000 dimensiones). Los autores mencionan dos posibles causas del deterioro del rendimiento de los algoritmos cuando se incrementa la dimensionalidad del espacio de búsqueda. La primera es el aumento de la complejidad del problema cuando aumenta su tamaño y la otra es que el espacio de soluciones del problema aumenta exponencialmente cuando se incrementa la dimensionalidad.

Los problemas de prueba son básicamente funciones que se basan en los propuestos para la competencia de optimización global realizada durante el CEC'2005 y que pueden ser unimodales o multimodales y que a su vez, pueden ser separables o no separables buscando parecerse a problemas del mundo real. En la tabla 4.1 se listan los problemas utilizados.

Tabla 4.1: Para realizar cada prueba se asignó un número fijo de evaluaciones de 5000 × *dimensionalidad*. El rendimiento del algoritmo lo indica el valor de la función objetivo al finalizar dichas evaluaciones. Cada prueba se ejecutó 25 veces para obtener una media del rendimiento. (puede verse a mayor detalle cada función del CEC'2008 en el apéndice A)

	Nombre de la función	**Tipo**	**Separabilidad**
f_1	Función esfera desplazada	Unimodal	Separable
f_2	Problema de Schwefel desplazado	Unimodal	No separable
f_3	Función de Rosenbrock desplazada	Multi-modal	No separable
f_4	Función de Rastrigin desplazada	Multi-modal	Separable
f_5	Función de Griewank desplazada	Multi-modal	No separable
f_6	Función de Ackley desplazada	Multi-modal	Separable
f_7	Función FastFractal "DoubleDip"	Multi-modal	No separable

El conjunto de problemas diseñado para el CEC'2008 probó la eficiencia de distintas metaheurísticas tales como evolución diferencial, PSO y los algoritmos genéticos [55]. Con relación a las variantes de PSO para optimización global de gran escala, los que reportaron los mejores resultados fueron: *Efficient Population Utilization Strategy for Particle Swarm Optimizer* (EPUS-PSO) [56] y *Dynamic multi-swarm particle swarm optimizer with local search for Large Scale Global Optimization* (DMS-PSO, por sus siglas en inglés) [57]. Así mismo, el conjunto de problemas del CEC'2008 fue utilizado en el artículo donde se propuso el algoritmo de CSO [4] el cual, utilizamos como base para aplicar nuestra mejora. En la tabla 4.2 mostramos los resultados reportados con estos tres algoritmos antes mencionados.

Tabla 4.2: Comparación del rendimiento de los algoritmos basados en PSO que concursaron en el CEC'2008 (EPUS-PSO y DMS-PSO) y el algoritmo base que utilizamos para nuestra mejora (CSO).

Algoritmo	$\mathbf{f_1}$	$\mathbf{f_2}$	$\mathbf{f_3}$	$\mathbf{f_4}$	$\mathbf{f_5}$	$\mathbf{f_6}$	$\mathbf{f_7}$
EPUS-PSO	5.53E+02	4.66E+01	8.37E+05	7.58E+03	5.89E+00	1.89E+01	-6.62E+03
DMS-PSO	0.00E+00	9.15E+01	8.98E+09	3.84E+03	0.00E+00	7.66E+00	-7.51E+03
CSO	1.09E-21	4.15E+01	1.01E+03	6.89E+02	2.26E-16	1.21E-12	-3.83E+06

En la clasificación final, el primer lugar lo obtuvo el *Multiple Trajectory Search for Multiobjetive Optimization* (MTS, por sus siglas en inglés) [58], mientras que el DMS-PSO tuvo el lugar 5 y el EPUS-PSO tuvo el lugar 8 [59].

4.2.2. Conjunto de problemas de prueba del CEC'2010

En el conjunto de problemas de prueba del CEC'2010, presentado por Ke Tang, Xiaodong Li, P. N. Suganthan, Zhenyu Yang y Thomas Weise [59] se menciona otra posible causa del deterioro que sufren los métodos de optimización a medida que aumenta la dimensionalidad del espacio de búsqueda. Ésta se refiere al cambio de las características de una función cuando aumenta la escala. Por ejemplo, la función de Rosenbrock (ver figura 4.1) es unimodal en dos dimensiones, pero se convierte en una función multimodal cuando el número de dimensiones aumenta (Hansen y Deb descubrieron que la función de Rosenbrock no es una función unimodal en dimensiones superiores [1]). Esto provoca que el método de optimización pueda dejar de funcionar a medida que aumenta la dimensionalidad.

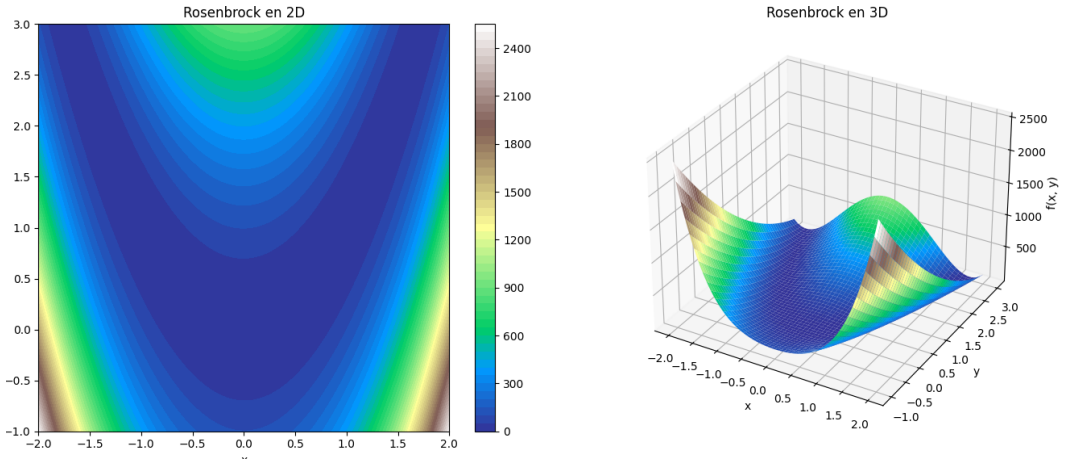

Figura 4.1: La función de Rosenbrock es una función muy utilizada en optimización numérica y se caracteriza por su valle parabólico estrecho, donde se encuentra su óptimo global [1].

Este conjunto de problemas de prueba fue diseñado con base en el concepto de separabilidad y no separabilidad (en este contexto, lo comparan con la *epistasis* en biología).

El conjunto de problemas de prueba tiene cuatro tipos de problemas de alta dimensionalidad:

- Funciones separables
- Funciones parcialmente separables, con un pequeño número de variables dependientes, y las restantes independientes
- Funciones parcialmente separables con múltiples subcomponentes independientes, cada uno de los cuales es m no separable
- Funciones completamente no separables

Para generar funciones con distintos grados de separabilidad, los autores proponen dividir las variables en grupos. Después, para cada grupo deciden si se mantienen independientes o se provoca una interacción entre las variables por medio de alguna técnica de rotación de coordenadas [60] y finalmente se aplica una función de aptitud de las mostradas en la tabla 4.3 a cada grupo de variables.

Tabla 4.3: Funciones base utilizadas en el conjunto de problemas de prueba del CEC´2010. Solo la primera función es separable y es utilizada a modo de demostración. Ver el apéndice B para mayor información.

	Función base
1	Función esférica
2	Función elíptica rotada
3	Problema 1.2 de Schwefel
4	Función de Rosenbrock
5	Función de Rastrigin rotada
6	Función de Ackley rotada

Con relación al conjunto de problemas de prueba, éste tiene 20 funciones (ver tabla 4.4) para ser utilizadas con $D = 1000$. Se propone un parámetro m para controlar el número de variables en cada grupo y definir el grado de separabilidad. Se establece $m = 50$ pero este parámetro es modificable. Además, esta versión es una mejora de los problemas de prueba utilizados para la competencia de optimización global a gran escala del CEC'2008 que no tenia funciones parcialmente no separables que ayudaran para saber más del comportamiento del algoritmo en estos escenarios.

Tabla 4.4: Funciones de prueba utilizadas en el CEC'2010. Puede verse a mayor detalle cada función en el apéndice B

	Nombre de la función	**Tipo**	**Separabilidad**
f_1	Función elíptica desplazada	Unimodal	Separable
f_2	Función de Rastrigin desplazada	Multi-modal	Separable
f_3	Función de Ackley desplazada	Multi-modal	Separable
f_4	Función elíptica desplazada y m-rotada de grupo simple	Unimodal	m no separable
f_5	Función de Rastrigin desplazada y m-rotada de grupo simple	Multi-modal	m no separable
f_6	Función de Ackley desplazada y m-rotada de grupo simple	Multi-modal	m no separable
f_7	Problema de Schwefel 1.2 desplazado y m-dimensional de grupo simple	Unimodal	m no separable
f_8	Función de Rosenbrock desplazada y m-dimensional de grupo simple	Multi-modal	m no separable
f_9	Función elíptica desplazada y m-rotada grupo $\frac{D}{2m}$	Unimodal	m no separable
f_{10}	Función de Rastrigin desplazada y m-rotada grupo $\frac{D}{2m}$	Multi-modal	m no separable
f_{11}	Función de Ackley desplazada y m-rotada grupo $\frac{D}{2m}$	Multi-modal	m no separable
f_{12}	Problema de Schwefel 1.2 desplazado y m-dimensional grupo $\frac{D}{2m}$	Unimodal	m no separable
f_{13}	Función de Rosenbrock desplazada y m-dimensional grupo $\frac{D}{2m}$	Multi-modal	m no separable
f_{14}	Función elíptica desplazada y m-rotada grupo $\frac{D}{m}$	Unimodal	m no separable
f_{15}	Función de Rastrigin desplazada y m-rotada grupo $\frac{D}{m}$	Multi-modal	m no separable
f_{16}	Función de Ackley desplazada y m-rotada grupo $\frac{D}{m}$	Multi-modal	m no separable
f_{17}	Problema de Schwefel 1.2 desplazado y m-dimensional grupo $\frac{D}{m}$	Unimodal	m no separable
f_{18}	Función de Rosenbrock desplazada y m-dimensional grupo $\frac{D}{m}$	Multi-modal	m no separable
f_{19}	Problema de Schwefel 1.2 desplazado	Unimodal	No separable
f_{20}	Función de Rosenbrock desplazada	Multi-modal	No separable

El conjunto de problemas diseñado para el CEC'2010 probó la eficiencia de distintas metaheurísticas tales como evolución diferencial, PSO y algoritmos genéticos [61]. Con relación a las variantes

de PSO para optimización global de gran escala, fue probado el *Dynamic Multi-Swarm Particle Swarm Optimizer with Subregional Harmony Search* (DMS-PSO-SHS) [62]. Así mismo, el conjunto de problemas del CEC'2010 fue utilizado en el algoritmo de MSLPSO [3] el cual, forma parte de un grupo de algoritmos que tomamos como referencia para elegir uno como base para nuestra mejora. En la tabla 4.5 mostramos los resultados reportados con los algoritmos antes mencionados.

Tabla 4.5: Comparación del rendimiento de los algoritmos en las funciones de la 1 a la 20 del conjunto de problemas de prueba del CEC'2010.

Función	**DMS-PSO-SHS**	**MSLPSO**
f_1	2.6144E-19	8.31E-19
f_2	7.1637E+01	7.92E+02
f_3	1.2825E-12	1.45E-13
f_4	2.0411E+11	5.30E+11
f_5	6.1023E+07	5.98E+06
f_6	5.8392E-05	9.07E-08
f_7	1.3440E+03	9.12E-02
f_8	1.0250E+07	8.16E+06
f_9	7.3404E+06	1.23E+07
f_{10}	5.2594E+03	6.55E+03
f_{11}	3.4766E+01	5.83E-12
f_{12}	6.0203E+02	1.06E+04
f_{13}	1.0087E+03	4.72E+02
f_{14}	1.6726E+07	1.38E+07
f_{15}	4.0071E+03	7.06E+02
f_{16}	6.4927E+01	7.11E-12
f_{17}	1.1444E+03	5.56E+04
f_{18}	2.0402E+03	1.27E+03
f_{19}	1.1031E+06	8.01E+06
f_{20}	2.8414E+02	9.37E+03

En la clasificación final del concurso, el primer lugar lo obtuvo el *Memetic algorithm based on local search chains for large scale continuous global optimization* (MA-SW-Chains por sus siglas en inglés) [63], mientras que el DMS-PSO-SHS tuvo el tercer lugar [62].

4.2.3. Conjunto de problemas de prueba del CEC'2013

En el conjunto de prueba del CEC 2013 [59], se propusieron problemas aún más desafiantes para los algoritmos de optimización global a gran escala. Se introdujeron obstáculos adicionales y configuraciones más complejas en los problemas para poner a prueba la capacidad de los algoritmos

para superar barreras y explorar eficientemente todo el espacio de búsqueda. El conjunto de prueba del CEC 2013 hizo énfasis en problemas de 1000 dimensiones.

Se incluyeron funciones en las cuales se requería que los algoritmos utilizaran diferentes estrategias de búsqueda en distintas fases del proceso de optimización, lo que permitió poner a prueba la capacidad de los algoritmos para adaptarse y cambiar de estrategia según fuera necesario. Además, se incluyen problemas de alta dimensionalidad mencionados tanto en el conjunto de problemas del CEC'2008 como en el conjunto del CEC'2010:

- El espacio de búsqueda crece exponencialmente a medida que el número de variables de decisión crece
- Las propiedades del espacio de búsqueda pueden cambiar a medida que el número de dimensiones crece
- La evaluación de los problemas de gran escala es muy costosa
- En algunos problemas la interacción entre las variables hace que no puedan optimizarse de manera independiente para obtener el óptimo global de una función objetivo, es decir, pueden ser no separables.

Para hacer el conjunto de problemas más cercano a problemas reales se han introducido las siguientes características:

- Tamaños no uniformes de subcomponentes: Las funciones con subcomponentes uniformes, no son representativas de muchos problemas del mundo real. Para representar mejor esta característica, las funciones en el conjunto de problemas contienen subcomponentes de diferentes tamaños.
- Desequilibrio en la contribución de los subcomponentes: En muchos problemas del mundo real, es probable que los subcomponentes de una función objetivo sean diferentes en su naturaleza, y por lo tanto, su contribución al valor global de la función objetivo puede variar. Al introducir tamaños de subcomponentes no uniformes, la contribución de los diferentes subcomponentes será automáticamente diferente, siempre que tengan tamaños distintos. Sin embargo, la contribución de un subcomponente puede ser amplificada o atenuada al multiplicar un coeficiente con el valor de cada función subcomponente.
- Funciones con subcomponentes superpuestos: En el conjunto de problemas del CEC'2010 los subcomponentes son subconjuntos disjuntos de las variables de decisión por lo que es teóricamente posible descomponer un problema de gran escala en un agrupamiento ideal de las variables de decisión. Sin embargo, cuando existe algún grado de superposición entre los subcomponentes, no habrá un agrupamiento óptimo único de las variables de decisión. Esto representa un desafío para los algoritmos de descomposición, ya que deben detectar

la superposición y diseñar una estrategia adecuada para optimizar dichos subcomponentes parcialmente interdependientes.

- Nuevas transformaciones a las funciones base: Las funciones del CEC'2010 son muy regulares y simétricas, por lo que se les aplicaron algunas transformaciones no lineales con el objetivo de romper la simetría e introducir ciertas irregularidades en el paisaje de aptitud [13]. Las transformaciones aplicadas no alteran las propiedades de separabilidad y modalidad de las funciones. Las tres transformaciones aplicadas son:

  - Mal condicionamiento: El mal condicionamiento se refiere al cuadrado de la relación entre la dirección más grande y la más pequeña de las líneas de contorno [13]. En el caso de un elipsoide, si éste se estira en la dirección de uno de sus ejes más que en los demás, se dice que la función está mal condicionada.
  - Ruptura de simetría: La mayoría de las funciones de referencia tienen patrones regulares. Es deseable introducir cierto grado de irregularidad aplicando alguna transformación.
  - Irregularidades: Algunos operadores que generan variaciones genéticas, especialmente aquellos basados en una distribución gaussiana, son simétricos, y si las funciones también son simétricas, existe un sesgo a favor de los operadores simétricos. Para eliminar dicho sesgo, es deseable una transformación que rompa la simetría.

Se han definido cuatro categorías principales de problemas a gran escala (ver tabla 4.6):

- Funciones completamente separables:
  - f_1 : función elíptica
  - f_2 : función Rastrigin
  - f_3 : función Ackley
- Dos tipos de funciones parcialmente separables:
  - Funciones parcialmente separables con un conjunto de subcomponentes no separables y un subcomponente completamente separable.
    - f_4 : función elíptica
    - f_5 : función Rastrigin
    - f_6 : función Ackley
    - f_7 : problema de Schwefel 1.2
  - Funciones parcialmente separables con solo un conjunto de subcomponentes no separables y sin subcomponentes completamente separables.
    - f_8 : función elíptica
    - f_9 : función Rastrigin
    - f_{10} : función Ackley

    - f_{11} : problema de Schwefel 1.2

- Funciones con subcomponentes superpuestos:
  - f_{12} : función de Rosenbrock
  - Funciones superpuestas con subcomponentes conformes: en este tipo de funciones, las variables de decisión compartidas entre dos subcomponentes tienen el mismo valor óptimo con respecto a ambas funciones de subcomponentes. En otras palabras, la optimización de un subcomponente puede mejorar el valor del otro subcomponente debido a la optimización de las variables de decisión compartidas.
    - f_{13} : función de Schwefel con subcomponentes superpuestos conformes
  - Funciones superpuestas con subcomponentes conflictivos: en este tipo de funciones, las variables de decisión compartidas tienen un valor óptimo diferente con respecto a cada una de las funciones de subcomponentes. Esto significa que la optimización de un subcomponente puede tener un efecto perjudicial en el otro subcomponente superpuesto debido a la naturaleza conflictiva de las variables de decisión compartidas.
    - f_{14} : función de Schwefel con subcomponentes superpuestos conflictivos
- Funciones completamente no separables
  - f_{15} : problema de Schwefel 1.2

Tabla 4.6: Funciones base utilizadas en el CEC'2013. Pueden verse más detalles de cada función en el Apéndice C

	Nombre de la función	**Tipo**	**Separabilidad**
f_1	Función elíptica	Unimodal	Completamnte separable
f_2	Función de Rastrigin	Unimodal	Completamnte separable
f_3	Función de Ackley	Unimodal	Completamnte separable
f_4	Función elíptica	Unimodal	Separable con subcomponentes separables
f_5	Función de Rastrigin	Multi-modal	Separable con subcomponentes separables
f_6	Función de Ackley	Multi-modal	Separable con subcomponentes separables
f_7	Problema de Schwefel 1.2	Multi-modal	Separable con subcomponentes separables
f_8	Función elíptica	Multi-modal	Separable con subcomponentes no separables
f_9	Función de Rastrigin	Multi-modal	Separable con subcomponentes no separables
f_{10}	Función de Ackley	Multi-modal	Separable con subcomponentes no separables
f_{11}	Problema de Schwefel 1.2	Multi-modal	Separable con subcomponentes no separables
f_{12}	Función de Rosenbrock	Multi-modal	Funciones Superpuestas
f_{13}	Función de Schwefel con subcomponentes superpuestos sin conflicto	Multi-modal	Funciones Superpuestas
f_{14}	Función de Schwefel con subcomponentes superpuestos con conflicto	Multi-modal	Funciones Superpuestas
f_{15}	Problema de Schwefel 1.2	Multi-modal	Completamente no separable

El conjunto de problemas diseñado para el concurso del CEC'2013 probó la eficiencia del algoritmo para optimización global de gran escala denominado *Large scale global optimization: Experi-*

mental results with MOS-based hybrid algorithms [64] [3]. En la tabla 4.7 mostramos los resultados reportados para el algoritmo antes mencionado.

En la clasificación final del concurso, el primer lugar lo obtuvo el *Large scale global optimization: Experimental results with MOS-based hybrid algorithms* [64].

El conjunto de problemas del CEC'13 también fue utilizado para la competencia sobre Optimización Global a Gran Escala del CEC'15. En esta ocasión se probó la eficiencia de distintas metaheurísticas tales como evolución diferencial, PSO y algoritmos genéticos [65]. Para la *sección de optimización global a gran escala* al parecer no compitieron algoritmos basados en cúmulos de partículas. Sin embargo, se probaron algoritmos como el *Smoothing and Auxiliary Functions Based Cooperative Coevolution for Global Optimization* (SACC) [66], el *Large Scale Global Optimization: Experimental Results with MOS-based Hybrid Algorithms* (MOS) [64], el *Scaling Up Covariance Matrix Adaptation Evolution Strategy using Cooperative Coevolution* (CC-CMA-ES) [67]. En la tabla 4.7 mostramos los resultados reportados con estos algoritmos antes mencionados.

Tabla 4.7: Comparación del rendimiento de los algoritmos en las funciones de la 1 a la 15 del conjunto de problemas de prueba del CEC'2013 en el concurso del CEC'2015.

Función	**SACC**	**MOS**	**CC-CMA-ES**
f_1	2.73E-24	0.00E+00	5.77E- 09
f_2	7.06E+02	8.32E+02	1.33E+ 03
f_3	1.11E+00	9.17E-13	1.51E-13
f_4	4.56E+10	1.74E+08	2.19E+09
f_5	7.74E+06	6.94E+06	7.28E+14
f_6	2.47E+05	1.48E+05	5.83E+05
f_7	8.98E+07	1.62E+04	7.44E+06
f_8	1.20E+15	8.00E+12	3.88E+14
f_9	5.98E+08	3.83E+08	3.71E+08
f_{10}	2.95E+07	9.02E+05	7.55E+05
f_{11}	2.78E+09	5.22E+07	1.59E+08
f_{12}	8.73E+02	2.47E+02	1.27E+03
f_{13}	1.78E+09	3.40E+06	6.69E+08
f_{14}	1.75E+10	2.56E+07	7.10E+07
f_{15}	2.01E+06	2.35E+06	3.03E+07

En el la clasificación final del concurso, el primer lugar lo obtuvo el *Large scale global optimization: Experimental results with MOS-based hybrid algorithms* [64].

4.2.4. Conjuntos de Problemas del CEC para problemas de optimización global de gran escala (LSGO)

A medida que avanza el desarrollo de los métodos de optimización global a gran escala, ha sido necesario agregar distintos comportamientos que se presentan en altas dimensiones. Esto ha ayudado a mejorar los problemas de prueba propuestos en el sentido de que se asemejan más a problemas del mundo real. Desde la publicación del primer conjunto de problemas de prueba para altas dimensiones (del CEC'2008) éstos se han ido actualizando constantemente. Sin embargo, existe la necesidad de contar con más divulgación, código fuente disponible, repositorios públicos nuevos, etc. que faciliten la generación de problemas de gran escala para poner a prueba los algoritmos que son desarrollados constantemente. En la tabla 4.8 podemos ver los conjuntos de problemas utilizados en las distintas ediciones del CEC especialmente diseñados para LSGO.

Tabla 4.8: Características de los conjuntos de prueba con problemas de gran escala utilizados en diferentes ediciones del IEEE CEC

Edición	**Nombre del Conjunto de Problemas utilizado**	**Dimensionalidad**	**Características Principales**
2008	CEC'2008 Competition on Large Scale Global Optimization	100, 500 y 1000	Siete funciones que deben utilizarse con 100, 500 y 1000 dimensiones (21 problemas de minimización)
2010	Benchmark Functions for the CEC'2010 Special Session and Competition on Large-Scale Global Optimization	1000 variables	20 problemas. Se utilizaron funciones totalmente separables, no separables y parcialmente separables
2012	CEC2010 Benchmark for Constrained Optimization	1000 variables	Optimización con restricciones lineales y no lineales, aplicados en problemas de gran escala.
2013	CEC2013 Large-Scale Benchmark Functions	1000 variables	Mejora del conjunto del CEC2010, con funciones más difíciles, no separables y complejas en alta dimensionalidad.
2015	CEC2013 Large-Scale Benchmark Functions	1000 variables	Se utilizó el mismo conjunto de problemas del CEC´2013.

En ediciones más recientes como por ejemplo en el concurso del CEC'2021 [68], encontramos el algoritmo *A Modified APSODEE for Large Scale Optimization* [69] el cual, presenta mejores resultados que sus antecesores. Aunque no se encontró registro de algún concurso sobre LSGO, los resultados son competitivos y los presentamos en la tabla 4.9.

Tabla 4.9: Resultados de la media en cada una de las funciones del conjunto de pruebas del CEC'2013 que se utilizó en el concurso del CEC'2021.

Función	**MAPSODEE**
f_1	1.53E-21
f_2	5.67E+02
f_3	2.16E+01
f_4	7.85E+08
f_5	6.18E+05
f_6	1.06E+06
f_7	4.93E+04
f_8	3.05E+13
f_9	3.63E+07
f_{10}	9.39E+07
f_{11}	1.80E+07
f_{12}	9.95E+02
f_{13}	6.38E+06
f_{14}	1.50E+07
f_{15}	2.36E+06

Capítulo 5

Diseño de un algoritmo de PSO para alta dimensionalidad

El algoritmo diseñado en la presente tesis tiene como finalidad mejorar los resultados hasta ahora obtenidos por otras versiones basadas en PSO para problemas de gran escala (es decir, para problemas de 100 o más variables). Por otro lado, también se busca tener en lo posible un algoritmo sencillo. Una forma en que este objetivo puede llevarse a cabo es, tomar como base un algoritmo del estado del arte que muestre resultados competitivos y modificarlo o aplicarle nuevas estrategias de búsqueda que le ayuden a obtener mejores soluciones (nosotros intentaremos manipular la exploración y explotación del cúmulo de partículas según el tipo de espacio de búsqueda en el que se crea ésta). Una vez realizada la mejora, usaremos el conjunto de problemas del CEC'2013 para verificarla.

5.1. Un diseño mejorado, con base en el algoritmo CSO

Para la elección del CSO (ver algoritmo 2) como base de nuestro trabajo, lo primero que se hizo fue comparar el rendimiento de varios algoritmos del estado del arte y luego, seleccionamos aquel que consideramos que era la mejor opción. El criterio para la elección del algoritmo deba cumplir con tres aspectos importantes:

- Proporcionar resultados competitivos.
- Implementación sencilla con relación a la mejora que proporcionaba.
- Funcionar en alta dimensionalidad.

Para la comparación se utilizó el CSO [4], el SL-PSO [5] y el MSL-PSO [3], pues todos ellos son representativos del estado del arte en el área. En la tabla 5.1 se muestra un resumen de las características de estos algoritmos.

Tabla 5.1: Comparación de los algoritmos CSO, SL-PSO y MSL-PSO.

Algoritmo	**Año y Autor**	**Características principales**	**Ventajas**	**Desventajas**	**Conjunto de problemas y pruebas con 1000 dimensiones**
CSO [4]	2013, Cheng y Shi	- Se basa en competencias entre partículas - Introduce la idea de líder y seguidor - Principio de funcionamiento sencillo	- Mejor rendimiento en problemas de alta dimensionalidad - Buena exploración y explotación	- Sensible a la inicialización - Puede requerir ajuste manual de parámetros	- Conjunto de problemas del CEC'2008 - Pruebas en funciones: F1-F7
SL-PSO [5]	2015, Zheng et al.	- Integra aprendizaje social para mejorar el comportamiento del cúmulo - Cada partícula adapta su comportamiento basándose en experiencias compartidas	- Mejora la convergencia en problemas escalables - Reduce la probabilidad de convergencia prematura	- Mayor complejidad computacional debido al cálculo del aprendizaje social	- Conjunto de problemas del CEC'2008 - Pruebas en funciones: F1-F7
MSL-PSO [3]	2017, Cheng et al.	- Combina estrategias múltiples en el aprendizaje - Uso eficiente de diferentes métodos de aprendizaje en paralelo	- Flexible y adaptable a diversas configuraciones - Resuelve problemas grandes con mayor eficacia	- Complejo de implementar debido a la gestión de múltiples estrategias	- Conjunto de problemas del CEC'2008 y del CEC'2010 - F1-F7 y F1-F20

Después, se obtuvieron los programas de cada algoritmo propuesto (el código fuente del algoritmo MSL-PSO no se pudo obtener, por tal motivo, se desarrolló una implementación propia con base en [3]).

Una vez se tuvieron los programas (en C), se ejecutaron para poder hacer una comparativa de resultados. Primero se ejecutaron los programas para el conjunto de problemas CEC'2008 y se compararon los resultados con lo reportado por los autores de cada artículo. Con esto, se verificó

que el funcionamiento del código fuera el correcto, lo cual se puede apreciar en la tabla 5.2. Con respecto a la comparación de resultados de las tres versiones propuestas, pudimos verificar que para el conjunto de problemas CEC'2008, se presenta una ligera ventaja del algoritmo CSO (gana en tres problemas), frente al SL-PSO (gana en dos problemas) y el MSL-PSO (gana solo en un problema) como puede verse también en la tabla 5.2. Podemos decir que las estrategias utilizadas por los diferentes algoritmos tienen un efecto similar en las características del conjunto de problemas de gran escala del CEC'2008.

Tabla 5.2: Comparación de la media(μ) de los algoritmos de PSO de alta dimensionalidad utilizando el conjunto de problemas del CEC'2008. Se marcaron con color verde los mejores resultados obtenidos de los programas proporcionados por cada autor (con excepción del MSL-PSO, el cual se implementó ya que no se disponía del código fuente por parte del autor).

Problema (1000_d)	**Óptimo Global**		**CSO (Resultados de Artículo [4])**	**CSO (Implementación) CEC2008)**	**SL-PSO (Resultados de Artículo [5])**	**SL-PSO (Implementación) CEC2008)**	**(MSLPSO) (Resultados de Artículo [3])**	**(MSLPSO) (Implementación) (CEC'2008)**
F_1	0	μ	1.09E-21	1.07E-21	7.10E-23	7.11E-23	9.33E-25	8.12E-19
		σ^2		9.30E-46		3.94E-48		1.73E-36
		σ		3.05E-23		1.98E-24		1.31E-18
F_2	0	μ	4.15E+01	3.87E+01	8.87E+01	1.65E+02	1.61E+01	8.80E+03
		σ^2		1.29607		8.14472		5.03E+04
		σ		1.13845		2.8539		224.300
F_3	0	μ	1.01E+03	9.94E+02	1.04E+03	2.59E+03	9.75E+02	2.20E+01
		σ^2		481.466		10216300		2.12E-05
		σ		21.9423		3196.29		0.004610
F_4	0	μ	6.89E+02	7.07E+02	5.89E+02	9.82E+03	5.50E+02	1.31E+10
		σ^2		1421.7		13288.7		1.22E+19
		σ		37.7054		115.277		3.48E+09
F_5	0	μ	2.26E-16	2.22E-16	4.44E-16	4.66E-16	1.10E-16	8.78E+05
		σ^2		6.78E-42		1.97E-33		1.43E+10
		σ		2.60E-21		4.44E-17		120000
F_6	0	μ	1.21E-12	1.19E-12	3.44E-13	3.44E-13	1.11E-14	1.07E+06
		σ^2		4.38E-28		2.63E-29		1.81E+06
		σ		2.09E-14		5.13E-15		1345.00
F_7	0	μ	-3.83E+06	-3.83E+06	-1.30E+04	-1.66E+06	-3.00E+04	-1.12E+08
		σ^2		2.33E+09		4.33E+09		3.15E+16
		σ		48309.5		65841.6		1.77E+08

Una segunda prueba consistió en comparar los resultados obtenidos de la ejecución de cada algoritmo, esta vez, utilizando el conjunto de problemas del CEC'2010. Pudimos observar que la implementación de MSL-PSO ya no siguió los resultados reportados por sus autores [3]. Aún así, a modo de referencia, se decidió continuar con el código fuente disponible y reportar los resultados obtenidos. Otra observación es que comenzamos por obtener mejores resultados con el algoritmo CSO que con el SL-PSO, como lo muestra la tabla 5.3.

Tabla 5.3: Comparación de la media(μ) de los algoritmos de PSO de alta dimensionalidad (CSO, SLPSO y MSLPSO). En verde se marca la mejor solución a cada problema para el conjunto de problemas del CEC'2010. Como se puede observar, el algoritmo CSO obtiene los mejores resultados.

Problema (1000_d)	**óptimo global**		**CSO (código base) CEC'2010**	**SL-PSO (código base) CEC'´2010**	**MSLPSO (código base) CEC'2010**
F_1	0	μ	4.29E-17	2.20E-18	1.14796e-19
		σ^2	9.39E-36	3.50E-39	1.43212e-40
		σ	3.06E-18	5.92E-20	1.19671e-20
F_2	0	μ	7.52E+03	9.73E+03	1.37E+04
		σ^2	35703.4	7976.23	4858.23
		σ	188.953	89.3097	69.701
F_3	0	μ	2.40E-09	3.64E-13	2.12E+01
		σ^2	3.92E-20	1.32E-29	0.0016986
		σ	1.98E-10	3.64E-15	0.0412141
F_4	0	μ	8.90E+11	6.79E+11	6.79E+14
		σ^2	3.72E+22	1.46E+22	6.12E+27
		σ	1.93E+11	1.21E+11	7.82E+13
F_5	0	μ	8.75E+06	1.18E+07	7.99E+07
		σ^2	2.53E+12	9.98E+12	2.87E+14
		σ	1.59E+06	3.16E+06	1.69E+07
F_6	0	μ	9.00E-07	2.16E+01	1.32E+07
		σ^2	8.70E-16	2.68E-05	6.24E+12
		σ	2.95E-08	0.00518325	2.49E+06
F_7	0	μ	1.84E+04	1.21E+04	1.38E+11
		σ^2	18625900	41117500	3.35E+20
		σ	4315.77	6412.29	1.83E+10

Problema (1000_d)	**óptimo global**		**CSO (código base) CEC'2010**	**SL-PSO (código base) CEC'´2010**	**MSLPSO (código base) CEC'2010**
F_8	0	μ	3.86E+07	2.78E+07	6.02E+07
		σ^2	3.95E+09	3.98E+10	1.34E+15
		σ	62917	199600	3.66E+07
F_9	0	μ	1.52E+11	2.94E+07	2.79E+07
		σ^2	1.34E+10	7.04E+11	2.16E+12
		σ	15796	839301	1.46E+06
F_{10}	0	μ	1.07E+04	1.06E+04	1.44E+04
		σ^2	2201.13	5091.92	3689.5
		σ	46.9162	71.3577	60.7413
F_{11}	0	μ	7.42E+01	2.36E+02	2.37E+02
		σ^2	757.825	0.0248536	0.015176
		σ	27.5286	0.15765	0.123191
F_{12}	0	μ	3.98E+05	2.47E+06	1.11E+07
		σ^2	2085220000	3.02E+11	6.363E+10
		σ	45664.2	549806	252251
F_{13}	0	μ	6.33E+02	1.98E+03	5.11E+09
		σ^2	65547	3847010	1.18E+19
		σ	256.021	1961.38	3.43E+09
F_{14}	0	μ	2.40E+08	2.39E+08	1.28E+08
		σ^2	1.77E+14	3.66E+14	2.26E+14
		σ	1.33E+07	1.91E+07	1.50E+07
F_{15}	0	μ	1.08E+04	1.11E+04	9.84E+03
		σ^2	3296.64	5596.12	4.85E+07
		σ	57.4164	74.8072	6965.9

Finalmente, se realizó una tercera comparación utilizando los mismos algoritmos, sin embargo, esta vez para el conjunto de problemas del CEC'2013. En este caso, se pudo observar que comienza a ser más notoria la robustez del algoritmo CSO frente a sus competidores, ya que aunque en general disminuye su rendimiento, el CSO entrega los mejores resultados como lo muestra la tabla 5.4.

Tabla 5.4: Comparación de la media(μ) de los algoritmos de PSO de alta dimensionalidad (CSO, SLPSO y MSLPSO). En verde se marca la mejor solución a cada problema, para el conjunto de problemas del CEC´2013

Problema (1000_d)	**óptimo global**		**CSO (código base) CEC'2013**	**SL-PSO (código base) CEC'2013**	**(MSLPSO) (código base) CEC'2013**
F_1	0	μ	3.61E-17	8.55E-19	2.70E+03
		σ^2	1.06E-36	1.58E-36	3.99E+06
		σ	1.03E-18	1.25E-18	1.99E+03
F_2	0	μ	8.59E+03	8.75E+03	9.65E+03
		σ^2	107574	49596.3	8.77E+05
		σ	327.985	222.702	9.36E+02
F_3	0	μ	2.10E+01	2.16E+01	2.16E+01
		σ^2	1.79E-05	2.10E-05	2.01E-05
		σ	0.0042358	0.00459078	4.48E-03
F_4	0	μ	1.13E+10	1.30E+10	7.49E+10
		σ^2	2.15E+18	1.21103e+19	1.99E+20
		σ	1.46E+09	3.47999e+09	1.41E+10
F_5	0	μ	7.69E+05	8.76E+05	8.95E+06
		σ^2	1.40E+10	1.42184e+10	2.24E+11
		σ	1.40E+10	119241	4.74E+05
F_6	0	μ	1.06E+06	1.06E+06	1.06E+06
		σ^2	821433	1793620	7.05E+05
		σ	906.329	1339.26	8.40E+02
F_7	0	μ	5.21E+06	1.11E+08	1.19E+10
		σ^2	1.40E+12	3.13E+16	1.25E+19
		σ	1.18E+06	1.76E+08	3.54E+09

Problema (1000_d)	**óptimo global**		**CSO (código base) CEC'2013**	**SL-PSO (código base) CEC'2013**	**(MSLPSO) (código base) CEC'2013**
F_8	0	μ	2.70E+14	3.11716e+14	2.57E+15
		σ^2	3.96E+27	3.57986e+28	2.015E+30
		σ	6.29E+13	1.89205e+14	1.41E+15
F_9	0	μ	3.13E+07	9.34E+07	4.74E+08
		σ^2	5.67E+13	2.52064e+15	1.99E+16
		σ	7.53E+06	5.0206e+07	1.41E+08
F_{10}	0	μ	9.41E+07	6.60E+07	9.40E+07
		σ^2	4.27E+10	1.1595e+15	7.16E+10
		σ	206841	3.40514e+07	2.67E+05
F_{11}	0	μ	3.27E+09	1.0495E+14	1.72E+12
		σ^2	6.35E+18	8.36527e+27	1.81E+24
		σ	6.35E+18	9.14618e+13	1.34E+12
F_{12}	0	μ	1.05E+03	1.01E+03	2.45E+05
		σ^2	434.222	1411.85	1.12E+10
		σ	20.838	37.5746	1.05E+05
F_{13}	0	μ	9.80E+08	1.91771e+13	3.44E+11
		σ^2	1.75E+17	1.10233e+27	1.77E+22
		σ	4.19E+08	3.32014e+13	1.33E+11
F_{14}	0	μ	3.09E+09	1.07014e+14	3.20E+12
		σ^2	1.13E+18	7.73263e+27	5.38E+24
		σ	1.06E+09	8.79354e+13	2.32E+12
F_{15}	0	μ	7.56E+07	2.17E+10	4.81E+09
		σ^2	2.27E+13	4.95E+20	1.90E+19
		σ	4.76E+06	2.22E+10	4.36E+09

Al finalizar las comparaciones y observar los resultados (como ya comentamos al principio de este capítulo) elegimos al CSO [4] como la mejor opción para utilizarlo como base para nuestra mejora. Además, otro aspecto positivo de este algoritmo y que apoya la decisión de elegirlo, es que no difiere mucho con respecto a la versión original [18], lo que lo vuelve sencillo y rápido en comparación a versiones más complejas [3]. El CSO, solo agrega un mecanismo de competencia

entre las partículas y actualiza a la partícula perdedora de la competencia, ésta, a su vez, aprende de la ganadora (como ya se mencionó anteriormente). Por lo tanto, se puede ver que el mecanismo utilizado por CSO no añade mayor complejidad en comparación a los recursos que utilizaría su versión original.

5.2. Estrategias utilizadas para mejorar al CSO

Una vez que se eligió el algoritmo base, se le incorporó una estructura de datos (una cola), la cual, tiene el propósito de censar el comportamiento del cúmulo sobre la región del espacio de búsqueda en el que éste se encuentre en el tiempo t_s. Cuando el cúmulo está explorando el espacio de búsqueda puede ser que una partícula encuentre una mejor posición local. Este hecho se almacenará como un "cero" en la cola; de otro modo, se guardará un "uno". Lo anterior se realiza después de actualizar la partícula que perdió la competencia. En cuanto al número de elementos que puede almacenar la estructura para censar el comportamiento del cúmulo, es igual al número total de partículas en el grupo. Esto es así, a modo de tener una ventana de información lo más completa posible, acerca de alguna tendencia que se esté presentando al explorar el espacio de búsqueda. Así mismo, la estructura se mantiene actualizada debido a que en cada iteración se ingresa la nueva lectura y se desecha la más antigua. En la figura 5.1 se muestra el diagrama de la estructura de datos que se utiliza.

Figura 5.1: Diagrama que representa la estructura de datos que guardará '0' cuando se encuentra un mejor local y '1' en caso contrario (esto para cada partícula), lo que crea un rastro de comportamiento de mejores locales en un rango de tiempo.

Ya que el censado es a cada s iteraciones, donde $s > tam_{cola}$ (para permitir que la estructura tenga tiempo de llenarse), la información que proporcione la cola (si encuentra o nó mejores resultados) se va almacenando y actualizando hasta que llega el nuevo tiempo de censar $(t_s + 1)$. En este tiempo $(t_s + 1)$, se tiene una especie de rastro del comportamiento de la posición de las últimas n partículas en ese momento, el cual podemos utilizar como fuente de información para suponer el tipo de espacio de búsqueda en el que se encuentra el cúmulo. A partir de esto, se podrían modificar los parámetros del algoritmo para ajustar su comportamiento social o cognitivo según se requiera para establecer un equilibrio entre la exploración y la explotación del espacio de búsqueda.

Como se ha mencionado, cada s iteraciones, se realiza un conteo de los "ceros" que se encuentran almacenados en la cola. Este conteo nos indica cuántos mínimos están encontrando las partículas de forma local. Si el conteo es alto, nos indicaría que el cúmulo es bastante diverso y está encontrando bastantes posiciones mejores, o que tal vez la función sea altamente multimodal, lo cual nos hace pensar que se puede mantener la exploración o el componente social como en el algoritmo base.

Si el número de "ceros" es bajo, podría ser que se tratara de una función unimodal o que el cúmulo pudiera estar atrapado en un óptimo local. En este caso se hace uso de un operador de varianza, el cual se calcula a partir de las mejores posiciones locales de cada partícula del cúmulo. Si la varianza es alta, entonces suponemos que el cúmulo está atrapado en un óptimo local, por lo que incrementamos el parámetro social $R_3(k,t)$.

Por ahora solo se está utilizando la información que proporciona la cola de datos para favorecer la exploración del cúmulo, pero creemos que un análisis más detallado podría ayudar a equilibrar mejor los componentes social y cognitivo de esta nueva versión para obtener aún mejores resultados. Con esta simple modificación podemos ver una mejora en nuestra versión en comparación con la versión base de CSO. Esto se muestra más adelante en la sección de resultados en la tabla 5.8.

Una segunda mejora se encontró a partir del componente social del CSO. Éste fomenta que las partículas aprendan de la media de las posiciones del cúmulo en lugar de *gbest* como se hace en el PSO original. En el artículo de CSO [4], los autores utilizan la media de todo el cúmulo como una versión global (la influencia de todas las partículas) y comentan que es posible mejorar la diversidad a partir de una media local de las partículas en un vecindario predefinido, (esto mejora el rendimiento en funciones multimodales). Por lo que nosotros elegimos solo las cinco mejores partículas para obtener una media de ellas. Esto de igual forma mejoró al algoritmo como se verá más adelante en la sección de resultados en la tabla 5.10.

5.3. Pseudocódigo del CSO mejorado

A continuación mostramos más detalladamente la nueva versión del algoritmo:

- Se inicializa de forma aleatoria (como en PSO) un cúmulo m de n partículas. Luego, se actualiza de manera iterativa. Cada partícula tiene una posición de D dimensiones $\mathbf{x}_i(t) = (x_{i,1}(t), x_{i,2}(t), \ldots, x_{i,n}(t))$ y un vector de velocidad de D dimensiones, $\mathbf{v}_i(t) = (v_{i,1}(t), v_{i,2}(t), \ldots, v_{i,n}(t))$ como en CSO.
- Se inicializa la cola de datos ($tam_{cola} = m$).
- Obtener la varianza de las mejores posiciones locales del cúmulo.

- Obtener la media de las posiciones de las mejores cinco partículas del cúmulo.
- En cada generación (t), el cúmulo se revuelve y se asignan parejas (se asume que el tamaño del cúmulo m es un número par), luego se realiza una competencia entre las dos partículas que forman el par.
- Como resultado de cada competencia, la partícula con una mejor aptitud, denominada "ganadora", será pasada directamente a la siguiente generación del cúmulo $(t+1)$
- Cada $s = 10000$ (evaluaciones de función (EF)) se realizará un conteo de "ceros" almacenados en la cola de datos. Inicialmente, la estructura está vacía por lo que se usa la condición que $s > tam_{cola}$.
- La partícula que pierde la competencia, denominada "perdedora", actualizará su posición y velocidad aprendiendo de la ganadora. Durante la actualización de la partícula modificaremos los factores aleatorios que tienen que ver con la actualización de la velocidad v_i:
  - R_1 := Factor aleatorio en el primer componente $R_1(k,t)V_{l,k}(t)$. Está relacionado con el factor de inercia ω y da estabilidad al algoritmo.
  - R_2 := Factor aleatorio en el segundo componente $R_2(k,t)(x_{w,k}(t) - x_{l,k}(t))$. Se asocia al aprendizaje cognitivo de la partícula.
  - R_3 := Factor aleatorio en el tercer componente $\phi R_3(k,t)(\bar{x}_k(t) - x_{l,k}(t))$. Se asocia al aprendizaje social de la partícula.

  Lo anterior se hace con relación a la cantidad de "ceros" que se encuentren en el conteo (ver algoritmo 7).
- Después de la actualización, la nueva aptitud de la partícula "perdedora" es comparada con su mejor posición local histórica. Si ésta es menor que la mejor local de la que se tiene registro (es decir, se encontró una mejor aptitud), se pondrá un cero en la estructura de datos. En caso contrario, esta nueva aptitud es peor y se pondrá un 1. Esta operación irá llenando la estructura a medida que pasan las competencias durante un rango de 10000 evaluaciones
- Después de aprender de la ganadora, la partícula perdedora también será pasada a la siguiente generación del cúmulo $(t+1)$
- El algoritmo termina cuando se llegue a un máximo de evaluaciones ($maxFE$) de la función de aptitud

La forma en que se realizan las competencias es igual que en el CSO. Es decir, para un cúmulo m ocurren $m/2$ competencias y la velocidad y posición de las $m/2$ partículas serán actualizadas. La figura 3.4 ilustra este funcionamiento.

La velocidad del perdedor se actualizará utilizando la estrategia de aprendizaje de la ecuación (3.1), como en CSO, donde:

- $\mathbf{x}_{w,k}(t)$ Es la posición del ganador en la k-ésima ronda de competencia en la generación t
- $\mathbf{x}_{l,k}(t)$ Es la posición del perdedor en la k-ésima ronda de competencia en la generación t
- $\mathbf{v}_{w,k}(t)$ Es la velocidad del ganador en la k-ésima ronda de competencia en la generación t
- $\mathbf{v}_{l,k}(t)$ Es la velocidad del perdedor en la k-ésima ronda de competencia en la generación t
- $R_1(k,t)$, $R_2(k,t)$, y $R_3(k,t)$ Son factores aleatorios $\in [0,1]^d$ determinados por la cantidad de "ceros" que contenga la cola de datos, como lo muestra el algoritmo 7.
- ϕ Es un parámetro de aprendizaje que controla la influencia de $\bar{x}_k(t)$
- $\bar{x}_k(t)$ Es el valor medio de la posición de las partículas relevantes. Para nuestra mejora se utilizaron las posiciones de las mejores cinco partículas. Se adoptó una versión local a diferencia de CSO:
  - $\bar{x}^l_{l,k}(t)$ Significa la posición media local de las partículas en un vecindario predefinido de la partícula l, con $(k = 5)$.

La posición del perdedor se puede actualizar ahora con la nueva velocidad, utilizando la misma ecuación que en el CSO (ver ecuación (3.2)).

La primera parte de la ecuación (3.1) asegura la estabilidad del proceso de búsqueda y es similar al término de inercia ω en el PSO original. También puede interpretarse como que $\omega = 1$ y se agrega un vector aleatorio $R_1(t)$.

La segunda parte es la componente cognitiva. En esta parte de la ecuación se da el aprendizaje de la partícula perdedora con respecto a la ganadora en lugar de hacerlo de la mejor personal $pBest$.

La tercera parte es el componente social en donde la partícula perdedora aprende de la posición media del cúmulo actual $\bar{x}_k(t)$ en lugar del mejor global $gBest$.

A continuación se presenta el algoritmo de nuestra versión mejorada:

5.3.1. Algoritmo de CSO mejorado

La nueva versión del CSO se describe en el algoritmo 7.

Algoritmo 7 Competitive Swarm Optimizer for Large Scale Optimization with Data Queue (CSOQ, por sus siglas en inglés)

Inicializar una población m de n partículas $\mathbf{x_i}$ y $\mathbf{v_i}$.
Inicializa cola de datos ($tam_{cola} = m$).
Obtener varianza de las mejores posiciones locales var.
Obtener la media de las mejores cinco partículas del cúmulo.
Evaluar la aptitud de cada partícula $f(\mathbf{x}_i)$ con base en la función objetivo.
Mientras no se cumpla el criterio de parada **hacer**
    Si $FEs\,\%10000 == 0$ **entonces**
        $contadorCeros = contarCeros(q)$
    Fin Si
    Emparejar las partículas aleatoriamente en $m/2$ pares.
    Para cada cada par $(\mathbf{x_i}, \mathbf{x_j})$ **hacer**
        Si $f(\mathbf{x}_i) < f(\mathbf{x}_j)$ **entonces** ▷ Para problemas de minimización
            Marcar $\mathbf{x}_i$ como el ganador y $\mathbf{x}_j$ como el perdedor.
        else
            Marcar $\mathbf{x}_j$ como el ganador y $\mathbf{x}_i$ como el perdedor.
        Fin Si
        Reincorporar la partícula ganadora al cúmulo ▷ Hacer ajuste dinámico
        Si $contadorCeros < ((upper - lower)/4)$ **entonces**
            Si $velocidad_pocentaje == 0$ **entonces**
                $R_3 = 1 + R_3$ ▷ Puede estar en un óptimo local
            else
                $R_2 = 1 + R_2$ ▷ Puede ser una función unimodal;
                $R_1 = 1 + R_1$ ▷ Puede ser una función unimodal;
            Fin Si
        else
            Si $var > 100000000000000$ **entonces**
                $R_3 = 1 + R_3$ ▷ Se fomenta la exploración
            Fin Si
        Fin Si
        Actualizar la velocidad del perdedor usando ecuación (3.1)
        Actualizar la posición del perdedor usando ecuación (3.2)
        Borrar o sobreescribir $\mathbf{x_i}$ y $\mathbf{x_j}$;
    Fin Para cada
    Reincorporar la partícula actualizada al cúmulo
Fin Mientras
Devolver la mejor solución encontrada $\mathbf{x}^* = \arg\min f(\mathbf{x}_i)$.

5.4. Resultados Experimentales

En esta sección mostramos los resultados experimentales de la ejecución de nuestro algoritmo mejorado. Para ello, tomamos como referencia la metodología propuesta en las competencias del Congreso sobre Computación Evolutiva (*Conference on Evolutionary Computation*), organizado por la sociedad de inteligencia Computacional del IEEE (*Institute of Electrical and Electronics Engineers*). Además, comparamos el funcionamiento de nuestra versión con respecto al CSO [4], al SLPSO [5] y al MSLPSO [3]. Como hemos mencionado anteriormente, estos tres algoritmos del estado de arte se enfocan en resolver problemas de optimización con un gran número de variables. Para hacer las comparaciones, utilizamos los conjuntos de problemas de prueba del CEC (por sus siglas en inglés) en sus ediciones 2008, 2010 y 2013, cuya prioridad es evaluar el desempeño de algoritmos de optimización global a gran escala.

5.4.1. Conjunto de problemas de prueba del CEC'2013

Para la evaluación de nuestro algoritmo, hemos adoptado el conjunto de problemas de prueba utilizados para la sesión especial y competencia de optimización global a gran escala organizados en el CEC'2013. Son 15 problemas (ver tabla 5.5) y son una extensión del conjunto de prueba del CEC'2010 cuyo objetivo es representar mejor las características de un mayor número de problemas del mundo real. De igual forma, este conjunto de problemas proporciona flexibilidad para comparar los algoritmos diseñados. En especial, los realizados para la optimización global a gran escala. Como ya se había mencionado, las principales características de esta nueva extensión de problemas son la introducción de desequilibrio entre la contribución de varios subcomponentes (tamaños no uniformes) y funciones superpuestas.

Tabla 5.5: Conjunto de problemas de prueba utilizados para la Sesión Especial y Competencia de optimización global a gran escala organizados en el CEC'2013 (puede verse más a detalle cada función del CEC'2008 en el apéndice C).

Función	Nombre	Rango	Global Óptimo	Propiedades
f_1	Función elíptica	$[-100, 100]^D$	0	Unimodal, separable, desplazada, con irregularidades locales suaves
f_2	Función de Rastrigin desplazada	$[-5, 5]^D$	0	Multimodal, separable, desplazada, irregularidades locales suaves
f_3	Función de Ackley desplazada.	$[-32, 32]^D$	0	Multimodal, separable, desplazada, irregularidades locales suaves
f_4	Función elíptica desplazada y rotada	$[-100, 100]^D$	0	Unimodal, parcialmente separable, desplazada, irregularidades locales suaves
f_5	Función de Rastrigin desplazada y rotada	$[-5, 5]^D$	0	Multimodal, parcialmente separable, desplazada, irregularidades locales suaves;
f_6	Función de Ackley desplazada y rotada	$[-32, 32]^D$	0	Multimodal, parcialmente separable, desplazada, irregularidades locales suaves;
f_7	Función Schwefel desplazada	$[-100, 100]^D$	0	Multimodal, parcialmente separable, desplazada, irregularidades locales suaves
f_8	Función elíptica desplazada y rotada	$[-100, 100]^D$	0	Unimodal, parcialmente separable, desplazada, irregularidades locales suaves
f_9	Función de Rastrigin desplazada y rotada	$[-5, 5]^D$	0	Multimodal, parcialmente separable, desplazada, irregularidades locales suaves
f_{10}	Función de Ackley desplazada y rotada	$[-32, 32]^D$	0	Multimodal, parcialmente separable, desplazada, irregularidades locales suaves
f_{11}	Función Schwefel desplazada	$[-100, 100]^D$	0	Unimodal, parcialmente separable, desplazada, irregularidades locales suaves
f_{12}	Función de Rosenbrock desplazada	$[-100, 100]^D$	0	Multimodal, separable, desplazada, irregularidades locales suaves
f_{13}	Schwefel desplazada con subcomponentes superpuestos	$[-100, 100]^D$	0	Unimodal, no separable, superpuesto, desplazado, irregularidades locales suaves;
f_{14}	Schwefel desplazada con subcomponentes superpuestos en conflicto.	$[-100, 100]^D$	0	Unimodal, no separable, subcomponentes en conflicto, desplazado, irregularidades locales suaves;
f_{15}	Función Schwefel desplazada	$[-100, 100]^D$	0	Unimodal, totalmente no separable, desplazada, irregularidades locales suaves;

5.4.2. Metodología

La metodología empleada para realizar los experimentos al nuevo algoritmo sigue el procedimiento propuesto en el CEC'2013. Esto, con la finalidad de hacer más justa y estandarizada la comparación contra otros algoritmos similares. La prueba consistió en utilizar un conjunto de 15 problemas de minimización (ver tabla 5.5). Para todos los problemas de utilizaron 1000 dimensiones. Además, se realizaron 25 ejecuciones por función y para cada problema se estableció un número máximo de 3×10^6 evaluaciones de la función objetivo. El criterio de finalización para cada problema se presenta cuando Max_FE alcance el valor 3×10^6.

5.4.3. Comparación de resultados

Un primer experimento para evaluar el comportamiento de nuestro algoritmo mejorado, fue ejecutarlo con el conjunto de problemas del CEC'2008. Estos resultados se presentan en la tabla 5.6 y se comparan con respecto a las dos versiones del estado del arte utilizadas como referencia. Se puede observar en la columna de nuestro algoritmo (CSOQ) que solo gana en la función f_4 y f_7. Sin embargo, con excepción de los resultados para f_5 y f_6, los otros resultados son muy similares a su versión base (CSO). Se esperaba que el algoritmo CSOQ fuera mucho mejor que sus contrincantes. Sin embargo, esto no ocurrió y creemos que esto puede deberse a que nuestro algoritmo se comporta mejor para funciones multimodales, ya que da preferencia a la exploración del espacio de búsqueda para intentar no quedar atrapado en óptimos locales. La función f_4 (función de Rastrigin desplazada) es multimodal separable y la función f_7 (Double Deep) es multimodal separable con un espacio de búsqueda pequeño (de $\mathbf{x} \in [-5, 5]^D$ y $\mathbf{x} \in [-1, 1]^D$ respectivamente). Las funciones f_5 (Griewank desplazada) y f_6 (Ackley desplazada) tal vez mostraron peores resultados debido al desplazamiento y a la forma de la función que no presenta muchas irregularidades y se tiene que explorar más sin encontrar resultados buenos.

Tabla 5.6: Comparación de los algoritmos del estado del arte contra nuestro nuevo algoritmo utilizando el conjunto de problemas CEC'2008. En verde se pueden observar los mejores resultados obtenidos del experimento. En gris remarcamos los resultados del algoritmo MSLPSO debido a que estos resultados se usan solo como referencia para nuestro estudio comparativo

Problema (1000_d)	Óptimo Global Ideal		CSO (Resultados de Artículo [4])	CSO (Implementación CEC2008)	SL-PSO (Resultados de Artículo [5])	SL-PSO (Implementación CEC2008)	(MSLPSO) (Resultados de Artículo [3])	(MSLPSO) (Implementación CEC2008)	CSOQ (Implementación) CEC2008)
F_1	0	μ	1.09E-21	1.07E-21	7.10E-23	7.11E-23	9.33E-25	8.12E-19	2.15E-22
		σ^2		9.30E-46		3.94E-48		1.73E-36	8.86E-47
		σ		3.05E-23		1.98E-24		1.31E-18	9.41E-24
F_2	0	μ	4.15E+01	3.87E+01	8.87E+01	1.65E+02	1.61E+01	8.80E+03	8.35E+01
		σ^2		1.29607		8.14472		5.03E+04	2.93
		σ		1.13845		2.8539		224.300	1.71
F_3	0	μ	1.01E+03	9.94E+02	1.04E+03	2.59E+03	9.75E+02	2.20E+01	1.12E+03
		σ^2		481.466		10216300		2.12E-05	6.64+03
		σ		21.9423		3196.29		0.004610	81.51
F_4	0	μ	6.89E+02	7.07E+02	5.89E+02	9.82E+03	5.50E+02	1.31E+10	5.83E+02
		σ^2		1421.7		13288.7		1.22E+19	4.93E+02
		σ		37.7054		115.277		3.48E+09	22.21
F_5	0	μ	2.26E-16	2.22E-16	4.44E-16	4.66E-16	1.10E-16	8.78E+05	4.88E-02
		σ^2		6.78E-42		1.97E-33		1.43E+10	4.39E-03
		σ		2.60E-21		4.44E-17		120000	6.63E-02
F_6	0	μ	1.21E-12	1.19E-12	3.44E-13	3.44E-13	1.11E-14	1.07E+06	5.73E-01
		σ^2		4.38E-28		2.63E-29		1.81E+06	2.03E-01
		σ		2.09E-14		5.13E-15		1345.00	4.51E-01
F_7	0	μ	-3.83E+06	-3.83E+06	-1.30E+04	-1.66E+06	-3.00E+04	1.12E+08	-7.20E+06
		σ^2		2.33E+09		4.33E+09		3.15E+16	2.30E+10
		σ		48309.5		65841.6		1.77E+08	1.51E+05

Se realizó un segundo experimento. En este caso, comparamos los resultados de nuestro algoritmo (utilizando el conjunto de problemas del CEC'2010), contra los otros dos algoritmos del estado del arte utilizando el mismo conjunto de problemas. Los resultados son mostrados en la tabla 5.7. Para este conjunto de problemas, nuestro algoritmo es mejor para más funciones. Particularmente, obtiene mejores resultados para los problemas multimodales f_2 (función de Rastrigin desplazada de $\mathbf{x} \in [-5,5]^D$), f_5 (función de Rastrigin desplazada y rotada de $\mathbf{x} \in [-5,5]^D$), f_{10} (función de Rastrigin desplazada y rotada de grupo $\frac{D}{2m}$, $\mathbf{x} \in [-5,5]^D$) y f_{12} (problema de Schwefel despalzado de $\mathbf{x} \in [-100,100]^D$) y para los unimodales f_{11} (función de Ackley desplazada y rotada de $\mathbf{x} \in [-32,32]^D$) y f_{15} (función de Rastrigin desplazada y rotada de $\mathbf{x} \in [-5,5]^D$). Como se puede observar, la mayoría de los buenos resultados se obtienen para funciones con espacios de búsqueda pequeños ($\mathbf{x} \in [-5,5]^D$), con excepción de la función f_{11} y f_{12} cuyo espacio de búsqueda es más grande, Pero aún así es mejor que las otras versiones. Esto demuestra que el algoritmo CSOQ, es robusto en el conjunto de problemas del CEC'2010.

Tabla 5.7: Comparación de la media(μ) de dos algoritmos del estado del arte, contra el nuestro, utilizando el conjunto de problemas del CEC'2010. En verde se pueden observar los mejores resultados obtenidos del experimento.

Problema (1000_d)	**óptimo global**		**CSO (Implementación CEC'2010)**	**SL-PSO (Implementación CEC'2010)**	**MSLPSO (Implementación CEC'2010)**	**CSOQ (Implementación CEC'2010)**
F_1	0	μ	4.29E-17	2.20E-18	1.14796e-19	1.07E-17
		σ^2	9.39E-36	3.50E-39	1.43212e-40	5.23E-37
		σ	3.06E-18	5.92E-20	1.19671e-20	7.23E-19
F_2	0	μ	7.52E+03	9.73E+03	1.37E+04	5.27E+03
		σ^2	35703.4	7976.23	4858.23	3.85E+02
		σ	188.953	89.3097	69.701	1.96E+01
F_3	0	μ	2.40E-09	3.64E-13	2.12E+01	5.48E-13
		σ^2	3.92E-20	1.32E-29	0.0016986	1.08E-28
		σ	1.98E-10	3.64E-15	0.0412141	1.03E-14
F_4	0	μ	8.90E+11	6.79E+11	6.79E+14	7.23E+11
		σ^2	3.72E+22	1.46E+22	6.12E+27	6.00E+21
		σ	1.93E+11	1.21E+11	7.82E+13	7.74E+10
F_5	0	μ	8.75E+06	1.18E+07	7.99E+07	8.65E+06
		σ^2	2.53E+12	9.98E+12	2.87E+14	2.18E+12
		σ	1.59E+06	3.16E+06	1.69E+07	1.47E+06
F_6	0	μ	9.00E-07	2.16E+01	1.32E+07	1.05E+01
		σ^2	8.70E-16	2.68E-05	6.24E+12	2.41857
		σ	2.95E-08	0.00518325	2.49E+06	1.55518
F_7	0	μ	1.84E+04	1.21E+04	1.38E+11	3.21E+04
		σ^2	18625900	41117500	3.35E+20	1.45E+08
		σ	4315.77	6412.29	1.83E+10	1.20+04
F_8	0	μ	3.86E+07	2.78E+07	6.02E+07	3.52E+07
		σ^2	3.95E+09	3.98E+10	1.34E+15	4.80E+09
		σ	62917	199600	3.66E+07	6.93E+04
F_9	0	μ	1.52E+11	2.94E+07	2.79E+07	1.52E+11
		σ^2	1.34E+10	7.04E+11	2.16E+12	1.34E+10
		σ	15796	839301	1.46E+06	1.15E+05
F_{10}	0	μ	1.07E+04	1.06E+04	1.44E+04	5.05E+02
		σ^2	2201.13	5091.92	3689.5	4.97E+02
		σ	46.9162	71.3577	60.7413	2.23E+01
F_{11}	0	μ	7.42E+01	2.36E+02	2.37E+02	3.42E+01
		σ^2	757.825	0.0248536	0.015176	2.45E+02
		σ	27.5286	0.15765	0.123191	1.56E+01
F_{12}	0	μ	3.98E+05	2.47E+06	1.11E+07	5.64E+04
		σ^2	2085220000	3.02E+11	6.363E+10	2.65E+06
		σ	45664.2	549806	252251	1.62E+03
F_{13}	0	μ	6.33E+02	1.98E+03	5.11E+09	7.76E+02
		σ^2	65547	3847010	1.18E+19	2.28E+05
		σ	256.021	1961.38	3.43E+09	4.78E+02
F_{14}	0	μ	2.40E+08	2.39E+08	1.28E+08	1.47E+08
		σ^2	1.77E+14	3.66E+14	2.26E+14	3.91E+13
		σ	1.33E+07	1.91E+07	1.50E+07	6.25E+06
F_{15}	0	μ	1.08E+04	1.11E+04	9.84E+03	7.43E+02
		σ^2	3296.64	5596.12	4.85E+07	2.19E+03
		σ	57.4164	74.8072	6965.9	4.68E+01

Finalmente, se realizó una tercera comparación de resultados de nuestro algoritmo utilizando el conjunto de problemas del CEC'2013 (para el cual fue diseñado), contra los otros algoritmos del estado del arte. Los resultados se muestran en la tabla 5.8.

Como se puede observar, en esta última comparación, nuestra nueva versión mantiene su buen desempeño e incluso mejora algunos resultados. Por ejemplo, para los problemas f_2 (función Rastrigin desplazada, $\mathbf{x} \in [-5, 5]^D$), f_5 (Rastrigin desplazada y rotada de 7 no separable, 1 separable, $\mathbf{x} \in [-5, 5]^D$), f_{11} (Schwefel desplazada y rotada de 20 no separable, $\mathbf{x} \in [-100, 100]^D$) y f_{15} (Schwefel desplazada, $\mathbf{x} \in [-100, 100]^D$) presenta un buen desempeño en contraste con el decremento en la eficiencia de los otros algoritmos contra los que se compara. Por tanto, el algoritmo propuesto resulta ser robusto y más eficiente frente a las características incorporadas en el conjunto de problemas del CEC'2013, como se aprecia en la tabla 5.8.

Nuestro algoritmo también obtiene ligeramente mejores resultados para los problemas multi-modales f_3 (función Ackley desplazada, $\mathbf{x} \in [-32, 32]^D$) y f_6 (Ackley Desplazada y Rotada de 7 no separable, 1 separable, $\mathbf{x} \in [-32, 32]^D$) y para los unimodales f_4 (elíptica desplazada y rotada de 7 no separable, 1 separable, $\mathbf{x} \in [-100, 100]^D$), f_8 (elíptica desplazada y rotada de 20 no separable, $\mathbf{x} \in [-100, 100]^D$), f_{13} (Schwefel desplazada con subcomponentes superpuestos sin conflicto, $\mathbf{x} \in [-100, 100]^D$). De estos resultados tal vez los más relevantes son f_4, f_8 y f_6 ya que su espacio de búsqueda es grande y además tienen componentes separables y no separables, una característica nueva en los problemas de este conjunto. Así mismo, la función f_{13} que tiene componentes superpuestos.

Tabla 5.8: Comparación de la media(μ) de los resultados obtenidos por nuestro algoritmo CSOQ. En verde se marca la mejor solución a cada problema comparando con respecto a CSO, SLPSO y MSLPSO para el conjunto de problemas del CEC'2013

Problema (1000_d)	**óptimo global**		**CSO (Implementación CEC'2013)**	**SL-PSO (Implementación CEC'2013)**	**(MSLPSO) (Implementación CEC'2013)**	**CSOQ (Implementación CEC'2013)**
F_1	0	μ	3.61E-17	8.55E-19	2.70E+03	1.15E-17
		σ^2	1.06E-36	1.58E-36	3.99E+06	9.80E-37
		σ	1.03E-18	1.25E-18	1.99E+03	9.90E-19
F_2	0	μ	8.59E+03	8.75E+03	9.65E+03	5.89E+02
		σ^2	107574	49596.3	8.77E+05	1.07E+03
		σ	327.985	222.702	9.36E+02	3.28E+01
F_3	0	μ	2.10E+01	2.16E+01	2.16E+01	2.02E+01
		σ^2	1.79E-05	2.10E-05	2.01E-05	1.87E-04
		σ	0.0042358	0.00459078	4.48E-03	1.37E-02
F_4	0	μ	1.13E+10	1.30E+10	7.49E+10	1.01E+10
		σ^2	2.15E+18	1.21103e+19	1.99E+20	2.96E+18
		σ	1.46E+09	3.47999e+09	1.41E+10	1.72E+09
F_5	0	μ	7.69E+05	8.76E+05	8.95E+06	6.65E+05
		σ^2	1.40E+10	1.42184e+10	2.24E+11	7.12E+09
		σ	1.40E+10	119241	4.74E+05	8.44E+04
F_6	0	μ	1.06E+06	1.06E+06	1.06E+06	9.96E+05
		σ^2	821433	1793620	7.05E+05	3.62E+04
		σ	906.329	1339.26	8.40E+02	1.90E+02
F_7	0	μ	5.21E+06	1.11E+08	1.19E+10	1.074E+07
		σ^2	1.40E+12	3.13E+16	1.25E+19	3.92E+12
		σ	1.18E+06	1.76E+08	3.54E+09	1.98E+06
F_8	0	μ	2.70E+14	3.11716e+14	2.57E+15	2.39E+14
		σ^2	3.96E+27	3.57986e+28	2.015E+30	4.63E+27
		σ	6.29E+13	1.89205e+14	1.41E+15	6.81E+13
F_9	0	μ	3.13E+07	9.34E+07	4.74E+08	4.55E+07
		σ^2	5.67E+13	2.52064e+15	1.99E+16	3.86E+13
		σ	7.53E+06	5.0206e+07	1.41E+08	6.22E+06
F_{10}	0	μ	9.41E+07	6.60E+07	9.40E+07	9.05E+07
		σ^2	4.27E+10	1.1595e+15	7.16E+10	9.27E+07
		σ	206841	3.40514e+07	2.67E+05	9.63E+03
F_{11}	0	μ	3.27E+09	1.0495E+14	1.72E+12	3.95E+08
		σ^2	6.35E+18	8.36527e+27	1.81E+24	9.31E+15
		σ	6.35E+18	9.14618e+13	1.34E+12	9.65E+07
F_{12}	0	μ	1.05E+03	1.01E+03	2.45E+05	1.15E+03
		σ^2	434.222	1411.85	1.12E+10	9.29E+03
		σ	20.838	37.5746	1.05E+05	9.64E+01
F_{13}	0	μ	9.80E+08	1.91771e+13	3.44E+11	6.67E+08
		σ^2	1.75E+17	1.10233e+27	1.77E+22	1.88E+16
		σ	4.19E+08	3.32014e+13	1.33E+11	1.37E+08
F_{14}	0	μ	3.09E+09	1.07014e+14	3.20E+12	4.19E+09
		σ^2	1.13E+18	7.73263e+27	5.38E+24	4.93E+18
		σ	1.06E+09	8.79354e+13	2.32E+12	2.22E+09
F_{15}	0	μ	7.56E+07	2.17E+10	4.81E+09	4.05E+06
		σ^2	2.27E+13	4.95E+20	1.90E+19	8.01E+10
		σ	4.76E+06	2.22E+10	4.36E+09	2.83E+05

A continuación mostramos un resumen de los resultados obtenidos

Tabla 5.9: Cantidad de funciones en las que cada algoritmo obtuvo el mejor rendimiento en diferentes conjuntos de problemas de prueba del CEC.

Algoritmo	**Ganó (CEC'2008)**	**Ganó (CEC'2010)**	**Ganó (CEC'2013)**
CSO	2 (f_2, f_5)	2 (f_6, f_{13})	3 (f_7, f_9, f_{14})
SL-PSO	2 (f_1, f_6)	4 (f_3, f_4, f_7, f_8)	3 (f_1, f_{10}, f_{12})
MSLPSO	1 (f_3)	3 (f_1, f_9, f_{14})	0
CSOQ	2 (f_4, f_7)	6 (f_2, f_5, f_{10}-f_{12}, f_{15})	**9** (f_2-f_6, f_8,f_{11}, f_{13}, f_{15})
Total	7	15	15

Como podemos observar en la tabla 4.9, nuestro algoritmo gana en la mayoría de problemas de prueba del CEC'2013 (ganó en nueve de quince). En principio suponemos que el algoritmo trabaja bien en espacios de búsqueda con rangos pequeños, por ejemplo $\mathbf{x} \in [-5, 5]^D$). Esta característica es observable desde que se realizaron las comparaciones con el CEC'2008. Para los siguientes conjuntos de prueba, el algoritmo mejorado (CSOQ) comienza a distinguirse de los otros debido a que las mejoras realizadas funcionan cuando la cola de datos incorporada al CSO detecta un espacio de búsqueda regular (con pocos mínimos). Esto hace que se incremente el componente que fomenta la exploración.

Como habíamos comentado en secciones anteriores, tenemos una segunda versión que mejora un poco más a nuestro algoritmo. Los resultados se muestran en la tabla 5.10. De igual forma, utilizamos el conjunto de problemas del CEC'2013.

Tabla 5.10: Comparación de la media(μ) de los resultados obtenidos por la versión mejorada de nuestro algoritmo llamada CSOQ*. En verde se marca la mejor solución a cada problema comparando con respecto a CSO, CSOQ, SLPSO y MSLPSO para el conjunto de problemas del CEC'2013

Problema (1000_d)	**óptimo global**		**CSO (Implementación CEC'2013)**	**SL-PSO (Implementación CEC'2013)**	**(MSLPSO) (Implementación CEC'2013)**	**CSOQ (Implementación CEC'2013)**	**CSOQ* CEC'2013)**
F_1	0	μ	3.61E-17	8.55E-19	2.70E+03	1.15E-17	3.43E-20
		σ^2	1.06E-36	1.58E-36	3.99E+06	9.80E-37	2.06E-41
		σ	1.03E-18	1.25E-18	1.99E+03	9.90E-19	4.54E-21
F_2	0	μ	8.59E+03	8.75E+03	9.65E+03	5.89E+02	7.97E+02
		σ^2	107574	49596.3	8.77E+05	1.07E+03	1.22E+03
		σ	327.985	222.702	9.36E+02	3.28E+01	3.49E+01
F_3	0	μ	2.10E+01	2.16E+01	2.16E+01	2.02E+01	2.03E+01
		σ^2	1.79E-05	2.10E-05	2.01E-05	1.87E-04	4.39E-04
		σ	0.0042358	0.00459078	4.48E-03	1.37E-02	2.09E-02
F_4	0	μ	1.13E+10	1.30E+10	7.49E+10	1.01E+10	2.59E+09
		σ^2	2.15E+18	1.21103e+19	1.99E+20	2.96E+18	4.68E+17
		σ	1.46E+09	3.47999e+09	1.41E+10	1.72E+09	6.84E+08
F_5	0	μ	7.69E+05	8.76E+05	8.95E+06	6.65E+05	6.29E+05
		σ^2	1.40E+10	1.42184e+10	2.24E+11	7.12E+09	1.10E+10
		σ	1.40E+10	119241	4.74E+05	8.44E+04	1.05E+05
F_6	0	μ	1.06E+06	1.06E+06	1.06E+06	9.96E+05	9.98E+05
		σ^2	821433	1793620	7.05E+05	3.62E+04	6.37E+06
		σ	906.329	1339.26	8.40E+02	1.90E+02	2.52E+03
F_7	0	μ	5.21E+06	1.11E+08	1.19E+10	1.074E+07	2.97E+06
		σ^2	1.40E+12	3.13E+16	1.25E+19	3.92E+12	9.64E+11
		σ	1.18E+06	1.76E+08	3.54E+09	1.98E+06	9.82E+05
F_8	0	μ	2.70E+14	3.11716e+14	2.57E+15	2.39E+14	5.26E+13
		σ^2	3.96E+27	3.57986e+28	2.015E+30	4.63E+27	3.43E+26
		σ	6.29E+13	1.89205e+14	1.41E+15	6.81E+13	1.85E+13
F_9	0	μ	3.13E+07	9.34E+07	4.74E+08	4.55E+07	4.25E+07
		σ^2	5.67E+13	2.52064e+15	1.99E+16	3.86E+13	3.43E+13
		σ	7.53E+06	5.0206e+07	1.41E+08	6.22E+06	5.86E+06
F_{10}	0	μ	9.41E+07	6.60E+07	9.40E+07	9.05E+07	9.06E+07
		σ^2	4.27E+10	1.1595e+15	7.16E+10	9.27E+07	1.90E+09
		σ	206841	3.40514e+07	2.67E+05	9.63E+03	4.36E+04
F_{11}	0	μ	3.27E+09	1.0495E+14	1.72E+12	3.95E+08	1.72E+08
		σ^2	6.35E+18	8.36527e+27	1.81E+24	9.31E+15	2.13E+15
		σ	6.35E+18	9.14618e+13	1.34E+12	9.65E+07	4.62E+07
F_{12}	0	μ	1.05E+03	1.01E+03	2.45E+05	1.15E+03	1.10E+03
		σ^2	434.222	1411.85	1.12E+10	9.29E+03	3.51E+03
		σ	20.838	37.5746	1.05E+05	9.64E+01	5.93E+01
F_{13}	0	μ	9.80E+08	1.91771e+13	3.44E+11	6.67E+08	3.26E+08
		σ^2	1.75E+17	1.10233e+27	1.77E+22	1.88E+16	2.10E+16
		σ	4.19E+08	3.32014e+13	1.33E+11	1.37E+08	1.45E+08
F_{14}	0	μ	3.09E+09	1.07014e+14	3.20E+12	4.19E+09	5.14E+07
		σ^2	1.13E+18	7.73263e+27	5.38E+24	4.93E+18	1.75E+14
		σ	1.06E+09	8.79354e+13	2.32E+12	2.22E+09	1.32E+07
F_{15}	0	μ	7.56E+07	2.17E+10	4.81E+09	4.05E+06	3.85E+06
		σ^2	2.27E+13	4.95E+20	1.90E+19	8.01E+10	3.66E+10
		σ	4.76E+06	2.22E+10	4.36E+09	2.83E+05	1.91E+05

La tabla 5.11 muestra un resumen de los nuevos resultados obtenidos.

Tabla 5.11: Cantidad de funciones en las que cada algoritmo obtuvo el mejor rendimiento en diferentes conjuntos de problemas de prueba del CEC.

Algoritmo	**Ganó (CEC'2008)**	**Ganó (CEC'2010)**	**Ganó (CEC'2013)**
CSO	2 (f_2, f_5)	2 (f_6, f_{13})	1 (f_9)
SL-PSO	2 (f_1, f_6)	4 (f_3, f_4, f_7, f_8)	2 (f_{10}, f_{12})
MSLPSO	1 (f_3)	3 (f_1, f_9, f_{14})	0
CSOQ*	2 (f_4, f_7)	6 (f_2, f_5, f_{10}-f_{12}, f_{15})	**12** (f_1-f_8,f_{11}, f_{13}-f_{15})
Total	7	15	15

Como puede observarse, esta versión de CSOQ solo perdió en las funciones f_9 (Rastrigin desplazada y rotada de 20-no separable, $\mathbf{x} \in [-5, 5]^D$), f_{10} (*Ackley* desplazada y rotada de 20-no separable, $\mathbf{x} \in [-32, 32]^D$) y f_{12} (Rosenbrock desplazada, $\mathbf{x} \in [-100, 100]^D$). En las primeras dos se puede deber a los componentes no separables que añaden complejidad para obtener buenas soluciones. En el caso de f_{12} creímos que podíamos obtener un mejor resultado. Sin embargo, el resultado obtenido no difiere mucho de los obtenidos por las otras versiones.

5.4.4. Análisis Estadístico

Para poder determinar la confiabilidad estadística de los resultados obtenidos en nuestra mejora, utilizaremos el método de bootstrap. Esta técnica estadística es muy utilizada para obtener estimaciones precisas a partir de muestras de $\mathbf{x}$ datos observados (en nuestro caso son 25 valores resultantes de 25 ejecuciones por cada problema de prueba, $n = 25$).

Se define una muestra bootstrap como $\mathbf{x}^* = (x_1^*, x_2^*, ..., x_n^*)$. A partir de $\mathbf{x}$ se generarán B submuestras de tamaño n ($\mathbf{x}^{*\mathbf{1}}, \mathbf{x}^{*\mathbf{2}}, ..., \mathbf{x}^{*\mathbf{B}}$ normalmente $B = 1000$). El número de submuestras es elevado para mitigar el sesgo de los resultados. En nuestro caso, queremos obtener un intervalo de confianza de la media estadística.

Para formar cada $\mathbf{x}^{*\mathbf{B}}$ submuestra (de tamaño n), se elige cada valor de forma aleatoria (con reemplazo) de la muestra de valores reales del experimento ($\mathbf{x}$). La probabilidad de obtener cada valor para formar la submuestra $\mathbf{x}^{*\mathbf{B}}$ a partir de la muestra real es $\frac{1}{n}$.

Para todos los problemas, se tomó un intervalo de confianza de 95 por ciento. En la tabla 5.12 se muestran los intervalos de confianza para cada problema.

Tabla 5.12: Intervalos de confianza para diferentes funciones en 1000 dimensiones.

Función	**1000 dim**
f_1	[3.26E−20, 3.61E−20]
f_2	[7.86E+02, 8.18E+02]
f_3	[2.0319E+01, 2.0335E+01]
f_4	[2.31E+09, 2.86+E09]
f_5	[5.89E+05, 6.71+E05]
f_6	[9.97E+05, 9.99E+05]
f_7	[2.59E+06, 3.37E+06]
f_8	[4.54E+13, 6.00E+13]
f_9	[4.02E+07, 4.48E+07]
f_{10}	[9.05E+07, 9.06E+07]
f_{11}	[1.53E+08, 1.89E+08]
f_{12}	[1.08E+03, 1.13E+03]
f_{13}	[2.71E+08, 3.86E+08]
f_{14}	[4.67E+07, 5.71E+07]
f_{15}	[3.78E+06, 3.93E+06]

Derivado de los resultados que se presentan en la tabla 5.12, presentamos la tabla 5.13 con el análisis de cada función.

Podemos observar que las funciones f_1, f_2, f_3, f_6, f_{10} y f_{12} tienen intervalos muy estrechos, lo que indica alta estabilidad en los resultados y por ende, son más confiables en términos de optimización. Las funciones f_4, f_9 y f_{15} tienen un intervalo moderado, mientras que las funciones f_5, f_7, f_8, f_{11}, f_{13} y f_{14} tienen intervalos más amplios, lo que sugiere mayor variabilidad. Estas funciones tienen la característica de ser no separables, lo que incrementa la dificultad para encontrar una buena solución. Cabe aclarar que estas funciones presentan los mejores resultados del grupo de algoritmos con los cuales fueron comparados (ver tabla 5.10).

Tabla 5.13: Análisis de estabilidad basado en los intervalos de confianza de bootstrap para diferentes funciones utilizando 1000 dimensiones.

Función	**1000 dim**	**Análisis**
f_1	[3.26E-20,3.61E-20]	**Excelente**. El intervalo es extremadamente pequeño, muy estable.
f_2	[7.86E+02, 8.18E+02]	**Bueno**. El intervalo es pequeño, estabilidad alta.
f_3	[2.0319E+01, 2.0335E+01]	**Bueno**. El intervalo es estrecho, estabilidad alta.
f_4	[2.31E+09, 2.86E+09]	**Bueno**. Alta variabilidad, sin embargo muestra los mejores resultados contra otros algoritmos.
f_5	[5.89E+05, 6.71E+05]	**Preocupante**. Alta variabilidad, sin embargo muestra los mejores resultados contra otros algoritmos.
f_6	[9.97E+05, 9.99E+05]	**Bueno**. Variación mínima, buena estabilidad.
f_7	[2.59E+06, 3.37E+06]	**Preocupante**. Alta variabilidad, sin embargo muestra los mejores resultados contra otros algoritmos.
f_8	[4.54E+13, 6.00E+13]	**Preocupante**. Muy alta variabilidad,sin embargo muestra los mejores resultados contra otros algoritmos.
f_9	[4.02E+07, 4.48E+07]	**Bueno**. Variación moderada, buena estabilidad.
f_{10}	[9.05E+07, 9.06E+07]	**Bueno**. El intervalo es estrecho, estabilidad alta.
f_{11}	[1.53E+08, 1.89E+08]	**Preocupante**. Alta variabilidad, sin embargo muestra los mejores resultados contra otros algoritmos.
f_{12}	[1.08E+03, 1.13E+03]	**Bueno**. El intervalo es estrecho, estabilidad alta.
f_{13}	[2.71E+08, 3.86E+08]	**Preocupante**. Alta variabilidad, sin embargo muestra los mejores resultados contra otros algoritmos.
f_{14}	[4.67E+07, 5.71E+07]	**Preocupante**. Muy alta variabilidad, sin embargo muestra los mejores resultados contra otros algoritmos.
f_{15}	[3.78E+06, 3.93E+06]	**Bueno**. Variación moderada, buena estabilidad.

Capítulo 6

Conclusiones y trabajo futuro

Si bien, existe una gran variedad de metaheurísticas que podemos utilizar para resolver un problema de optimización, y aun si nos limitamos solo al PSO, vimos que hasta la fecha siguen surgiendo nuevas versiones, debemos tomar en cuenta que la elección de un método para optimizar, está estrechamente relacionado con el tipo de problema que se quiera resolver. En esta tesis, nuestro interés fue resolver problemas de optimización global de alta dimensionalidad. El algoritmo propuesto, llamado CSOQ, funciona bien y obtiene los mejores resultados de entre un grupo de algoritmos representativos del estado del arte en el área (el conjunto de problemas del CEC'2013 el cual consiste de problemas de optimización global a gran escala). El algoritmo CSOQ consiste básicamente en dos mejoras al CSO. En la primera, se incorporó una especie de rastro que registra el tipo de espacio de búsqueda en el cual está el cúmulo y a partir de éste se guía la búsqueda. La segunda, es la elección de los individuos en los cuales se aplicará el aprendizaje social, en particular se eligen los mejores cinco partículas. Con estas dos mejoras se logró superar a los algoritmos con respecto a los cuales fue comparado. La mayor ventaja de nuestro algoritmo es que mantiene su principio de funcionamiento simple, lo cual implica que el consumo de recursos sea bajo. Además, la nueva estrategia que se agregó, influye en tiempo real para que el algoritmo vaya adaptando sus parámetros con relación al posible comportamiento que detecta el rastro de partículas que puede tener el espacio de búsqueda.

6.1. Trabajo a futuro

A pesar de que nuestro algoritmo mejoró los resultados de un grupo de algoritmos tomados del estado del arte, creemos que se puede mejorar la forma en la que se obtiene la información que permite definir mejor el espacio de búsqueda sobre el cual se encuentra el cúmulo. Además, el rastro en sí podría aportar más información para tomar acciones en ese momento y guiar la búsqueda conforme sea requerido. De igual forma, debido al bajo consumo de recursos, muy parecidos a los de la versión original del PSO, se puede utilizar en dispositivo pequeños o implementar en hardware.

Apéndice A

Definiciones de los Problemas de Prueba del CEC'2008

A.1. Funciones unimodales:

f1: Función esfera desplazada

$$f_1(\mathbf{x}) = \sum_{i=1}^{D} Z_i^2 + f_{bias_1}, \mathbf{z} = \mathbf{x} - \mathbf{o}, \mathbf{x} = [x_1, x_2, \ldots, x_D]$$

.

- **D:** dimensiones
- $\mathbf{o} = [o_1, o_2, \ldots, o_D]$: el óptimo global desplazado.

Propiedades:

- Unimodal
- Desplazada
- Separable
- Escalable
- Dimensiones (D): 100, 500 y 1000
- $\mathbf{x} \in [-100, 100]^D$, Optimo global: $\mathbf{x}^* = \mathbf{o}, F_1(x^*) = f_{bias_1}(x) = -450$

- Unimodal
- Separable
- Desplazada
- Irregularidades locales suaves
- Problema mal condicionado. Es muy sensible a errores de redondeo y pequeñas perturbaciones, lo que hace que la solución sea más difícil de encontrar o menos precisa (número de condición $\approx 10^6$) .

f2: Problema de Schwefel desplazada

$$F_2(\mathbf{x}) = \max_i \{|Z_i|, 1 \leq i \leq D\} + f_{\text{bias}_2}, \quad \mathbf{z} = \mathbf{x} - \mathbf{o}, \quad \mathbf{x} = [x_1, x_2, \ldots, x_D]$$

- **D:** dimensiones
- $\mathbf{o} = [o_1, o_2, \ldots, o_D]$: el óptimo global desplazado.

Propiedades:

- Unimodal
- Desplazada

- No separable
- Escalable
- Dimensiones (D): 100, 500 y 1000
- $\mathbf{x} \in [-100, 100]^D$, Óptimo global: $\mathbf{x}^* = \mathbf{o}, F_2(\mathbf{x}^*) = f_{\text{bias}_2}(\mathbf{x}) = -450$

- Unimodal
- No separable
- Desplazada
- Irregularidades locales suaves
- Problema mal condicionado (número de condición $\approx 10^6$)

A.2. Funciones multimodales:

f3: Función de *Rosenbrock* desplazada

$$f_3(\mathbf{x}) = \sum_{i=1}^{D-1} \left[100(Z_i^2 - Z_{i+1})^2 + (Z_i - 1)^2\right] + f_{\text{bias}_3}, \quad \mathbf{z} = \mathbf{x} - \mathbf{o} + 1, \quad \mathbf{x} = [x_1, x_2, \ldots, x_D]$$

- **D:** dimensiones
- $\mathbf{o} = [o_1, o_2, \ldots, o_D]$: el óptimo global desplazado.

Propiedades:

- Multi-modal
- Desplazada

- No separable
- Escalable
- Tiene un valle muy estrecho entre el óptimo local y el óptimo global
- Dimensiones (D): 100, 500 y 1000
- $\mathbf{x} \in [-100, 100]^D$, Óptimo global: $\mathbf{x}^* = \mathbf{o}, F_3(\mathbf{x}^*) = f_{\text{bias}_3}(\mathbf{x}) = 390$

f4: Función de *Rastrigin* desplazada

$$f_4(\mathbf{x}) = \sum_{i=1}^{D} \left[z_i^2 - 10\cos(2\pi Z_i) + 10\right] + f_{\text{bias}_4}, \quad \mathbf{z} = \mathbf{x} - \mathbf{o}, \quad \mathbf{x} = [x_1, x_2, \ldots, x_D]$$

- **D:** dimensiones
- $\mathbf{o} = [o_1, o_2, \ldots, o_D]$: el óptimo global desplazado.

Propiedades:

- Multi-modal
- Desplazada
- Separable
- Escalable
- El número de óptimos locales es enorme
- Dimensiones (D): 100, 500 y 1000
- $\mathbf{x} \in [-5, 5]^D$, Óptimo global: $\mathbf{x}^* = \mathbf{o}, F_4(\mathbf{x}^*) = f_{\text{bias}_4}(\mathbf{x}) = -330$

f5: Función de *Griewank* desplazada

$$f_5(\mathbf{x}) = \sum_{i=1}^{D} \left(\frac{Z_i^2}{4000}\right) - \prod_{i=1}^{D} \cos\left(\frac{Z_i}{\sqrt{i}}\right) + f_{\text{bias}_5}, \quad \mathbf{z} = \mathbf{x} - \mathbf{o}, \quad \mathbf{x} = [x_1, x_2, \ldots, x_D]$$

- **D:** dimensiones
- $\mathbf{o} = [o_1, o_2, \ldots, o_D]$: el óptimo global desplazado.

Propiedades:

- Multi-modal
- Desplazada
- No-separable
- Escalable
- Dimensiones (D): 100, 500 y 1000
- $\mathbf{x} \in [-600, 600]^D$, Óptimo global: $\mathbf{x}^* = \mathbf{o}, F_5(\mathbf{x}^*) = f_{\text{bias}_5}(\mathbf{x}) = -180$

f6: Función *Ackley* desplazada

$$f_6(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} Z_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi Z_i)\right) + 20 + e + f_{\text{bias}_6}, \quad \mathbf{z} = \mathbf{x} - \mathbf{o}$$

- $\mathbf{x} = [x_1, x_2, \ldots, x_D]$
- **D:** dimensiones
- $\mathbf{o} = [o_1, o_2, \ldots, o_D]$: el óptimo global desplazado.

Propiedades:

- Multi-modal
- Desplazada

- Separable
- Escalable
- Dimensiones (D): 100, 500 y 1000
- $\mathbf{x} \in [-32, 32]^D$, Óptimo global: $\mathbf{x}^* = \mathbf{o}, F_6(\mathbf{x}^*) = f_{\text{bias}_6}(\mathbf{x}) = -140$

f7: Función FastFractal "DoubleDip"

$$f_7(\mathbf{x}) = \sum_{i=1}^{D} fractal1D(x_i + twist(x_{(i \mod D)+1}))$$

$$twist(y) = 4(y^4 - 2y^3 + y^2)$$

$$fractal1D(x) \approx \sum_{k=1}^{3} \sum_{1}^{2^{k-1}} \sum_{1}^{ran2(o)} doubledip(x, ran1(o), \frac{1}{2^{k+1}(2 - ran1(o))})$$

$$doubledip(x, c, s) = \begin{cases} (-6144(x-c)^6 + 3088(x-c)^4 - 392(x-c)^2 + 1)s & \text{si } -0.5 < x < 0.5 \\ 0 & \text{en otro caso.} \end{cases}$$

- $\mathbf{x} = [x_1, x_2, \ldots, x_D]$
- **D:** dimensiones
- $\mathbf{o} = [o_1, o_2, \ldots, o_D]$: el óptimo global desplazado.
- **ran1(o):** número doble pseudoaleatorio generado con semilla o con distribución uniforme entre 0 y 1.
- **ran2(o):** número entero pseudoaleatorio generado con semilla o con distribución uniforme para el conjunto $\{0, 1, 2\}$.
- **fractal1D(x):** aproximación a un algoritmo recursivo.

Propiedades:

- Multi-modal
- No separable
- Escalable
- Dimensiones (D): 100, 500 y 1000
- $\mathbf{x} \in [-1, 1]^D$, Óptimo global desconocido $F_7(\mathbf{x}^*)$ desconocido
- La función tiene un comportamiento fractal con irregularidades notables en diferentes escalas.
- Se recomienda el uso de un ejecutable proporcionado debido a las características de la generación aleatoria.

Apéndice A. Definiciones de los Problemas de Prueba del CEC'2008

Apéndice B

Definiciones de los Problemas de Prueba del CEC'2010

B.1. Funciones base

B.1.1. Función esfera

$$f_{\text{sphere}}(\mathbf{x}) = \sum_{i=1}^{D} x_i^2$$

Propiedades:

- $\mathbf{x}$ es un vector de decisión, $\mathbf{x} = (x_1, x_2, \ldots, x_D)$
- **D** dimensiones
- Es comletamente separable
- Es muy simple y se utiliza generalmente para fines de demostración

- Se utiliza como subcomponente completamente separable para algunas de las funciones parcialmente separables definidas en este conjunto.

B.1.2. Función elíptica

La función elíptica original es separable y está dada por:

$$f_{\text{elliptic}}(\mathbf{x}) = \sum_{i=1}^{D} 10^{6\frac{i-1}{D-1}} x_i^2$$

Propiedades:

- $\mathbf{x}$ es un vector de decisión, $\mathbf{x} = (x_1, x_2, \ldots, x_D)$
- **D** dimensiones
- Para hacer que esta función sea no separable, se usará una matriz ortogonal para rotar las coordenadas
- 10^6 se llama número de condición y se utiliza para transformar una función esfera en una función elíptica [38]
- La función elíptica rotada se define como $f_{\text{rot\_elliptic}}(x) = f_{\text{elliptic}}(z), z = x \cdot M$, donde D es la dimensión, M es una matriz ortogonal de $D \times D$, y $x = (x_1, x_2, \ldots, x_D)$ es un vector fila de D dimensiones (es decir, una matriz de $1 \times D$)

B.1.3. Función de Rastrigin

La función de Rastrigin original es separable y está dada por:

$$f_{\text{rastrigin}}(\mathbf{x}) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$$

Propiedades:

- $\mathbf{x}$ es un vector de decisión, $\mathbf{x} = (x_1, x_2, \ldots, x_D)$
- **D** dimensiones
- Para hacer que esta función sea no separable, se usará una matriz ortogonal para rotar las coordenadas
- Para hacer que esta función sea no separable, se usará una matriz ortogonal para rotar las coordenadas. La función Rastrigin rotada se define como $f_{\text{rot\_rastrigin}}(x) = f_{\text{rastrigin}}(z), z = x \cdot M$, donde D es la dimensión, M es una matriz ortogonal de $D \times D$, y $\mathbf{x} = (x_1, x_2, \ldots, x_D)$ es un vector fila de D dimensiones (es decir, una matriz de $1 \times D$)
- La función de Rastrigin es un problema multimodal clásico. Es difícil porque el número de óptimos locales crece exponencialmente con el aumento de la dimensionalidad

B.1.4. Función de Ackley

La función de Ackley original es separable y está dada por:

$$f_{\text{ackley}}(\mathbf{x}) = -20 \exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e$$

Propiedades:

- $\mathbf{x}$ es un vector de decisión, $\mathbf{x} = (x_1, x_2, \ldots, x_D)$
- **D** dimensiones
- Para hacer que esta función sea no separable, se usará una matriz ortogonal para rotar las coordenadas
- Para hacer que esta función sea no separable, se usará una matriz ortogonal para rotar las coordenadas. La función Ackley rotada se define como $f_{\text{rot\_ackley}}(\mathbf{x}) = f_{\text{ackley}}(\mathbf{z}), \mathbf{z} = \mathbf{x} \cdot \mathbf{M}$, donde D es la dimensión, $\mathbf{M}$ es una matriz ortogonal de $D \times D$, y $\mathbf{x} = (x_1, x_2, \ldots, x_D)$ es un vector fila de D dimensiones (es decir, una matriz de $1 \times D$)

B.1.5. Problema de Schwefel 1.2

El problema de Schwefel 1.2 es no separable y está dado por:

$$f_{\text{schwefel}}(\mathbf{x}) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_i \right)^2$$

Propiedades:

- $\mathbf{x}$ es un vector de decisión, $\mathbf{x} = (x_1, x_2, \ldots, x_D)$
- $\mathbf{D}$ dimensiones

B.1.6. Función de Rosenbrock

La función de Rosenbrock original es no separable y está dada por:

$$f_{\text{rosenbrock}}(\mathbf{x}) = \sum_{i=1}^{D-1} \left(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2 \right)$$

Propiedades:

- $\mathbf{x}$ es un vector de decisión, $\mathbf{x} = (x_1, x_2, \ldots, x_D)$
- $\mathbf{D}$ dimensiones

B.2. Funciones de prueba

B.2.1. Funciones Separables:

B.2.1.1. F1: Función elíptica desplazada

$$F_1(x) = F_{\text{elíptica}}(z) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} z_i^2$$

Dimensión: $D = 1000$

- $x = (x_1, x_2, \ldots, x_D)$: la solución candidata – un vector fila de dimensión D
- $o = (o_1, o_2, \ldots, o_D)$: el óptimo global (desplazado)
- $z = x - o, \quad z = (z_1, z_2, \ldots, z_D)$: la solución candidata desplazada – un vector fila de dimensión D

Propiedades:

1. Unimodal
2. Desplazada
3. Separable
4. Escalable
5. $x \in [-100, 100]^D$
6. Óptimo global: $x^* = o, \quad F_1(x^*) = 0$

B.2.1.2. F2: Función de Rastrigin desplazada

$$F_2(x) = F_{\text{rastrigin}}(z) = \sum_{i=1}^{D} \left[z_i^2 - 10\cos(2\pi z_i) + 10 \right]$$

Dimensión: $D = 1000$

- $x = (x_1, x_2, \ldots, x_D)$: la solución candidata – un vector fila de dimensión D
- $o = (o_1, o_2, \ldots, o_D)$: el óptimo global (desplazado)
- $z = x - o, \quad z = (z_1, z_2, \ldots, z_D)$: la solución candidata desplazada – un vector fila de dimensión D

Propiedades:

1. Multimodal
2. Desplazada
3. Separable
4. Escalable
5. $x \in [-5, 5]^D$
6. Óptimo global: $x^* = o, \quad F_2(x^*) = 0$

B.2.1.3. F3: Función de Ackley desplazada

$$F_3(x) = F_{\text{ackley}}(z) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)\right) + 20 + e$$

Dimensión: $D = 1000$

- $x = (x_1, x_2, \ldots, x_D)$: la solución candidata – un vector fila de dimensión D
- $o = (o_1, o_2, \ldots, o_D)$: el óptimo global (desplazado)
- $z = x - o, \quad z = (z_1, z_2, \ldots, z_D)$: la solución candidata desplazada – un vector fila de dimensión D

Propiedades:

1. Multimodal
2. Desplazada
3. Separable
4. Escalable
5. $x \in [-32, 32]^D$
6. Óptimo global: $x^* = o, \quad F_3(x^*) = 0$

B.2.2. Funciones m no separables grupo simple

B.2.2.1. F4: Función elíptica desplazada y m-rotada de un solo grupo

$$F_4(x) = F_{\text{rot elliptic}}\left[z(P_1 : P_m)\right] \times 10^6 + F_{\text{elliptic}}\left[z(P_{m+1} : P_D)\right]$$

Dimensión: $D = 1000$

Tamaño del grupo: $m = 50$

- $x = (x_1, x_2, \ldots, x_D)$: la solución candidata – un vector fila de dimensión D
- $o = (o_1, o_2, \ldots, o_D)$: el óptimo global (desplazado)
- $z = x - o, \quad z = (z_1, z_2, \ldots, z_D)$: la solución candidata desplazada – un vector fila de dimensión D
- P: una permutación aleatoria de $\{1, 2, \ldots, D\}$

Propiedades:

1. Unimodal
2. Desplazada
3. Rotada m-de un solo grupo
4. m-no separable de un solo grupo
5. $x \in [-100, 100]^D$
6. Óptimo global: $x^* = o, \quad F_4(x^*) = 0$

B.2.2.2. F5: Función Rastrigin desplazada y m rotada de un Solo grupo

$$F_5(x) = F_{\text{rot rastrigin}}\left[z(P_1 : P_m)\right] \times 10^6 + F_{\text{rastrigin}}\left[z(P_{m+1} : P_D)\right]$$

Dimensión: $D = 1000$

Tamaño del grupo: $m = 50$

- $x = (x_1, x_2, \ldots, x_D)$: la solución candidata – un vector fila de dimensión D
- $o = (o_1, o_2, \ldots, o_D)$: el óptimo global (desplazado)

- $z = x - o, \quad z = (z_1, z_2, \ldots, z_D)$: la solución candidata desplazada – un vector fila de dimensión D
- P: una permutación aleatoria de $\{1, 2, \ldots, D\}$

Propiedades:

1. Multimodal
2. Desplazada
3. Rotada m-de un solo grupo
4. m-no separable de un solo grupo
5. $x \in [-5, 5]^D$
6. Óptimo global: $x^* = o, \quad F_5(x^*) = 0$

B.2.2.3. F6: Función Ackley desplazada y m rotada de un solo grupo

$$F_6(x) = F_{\text{rot ackley}}\left[z(P_1 : P_m)\right] \times 10^6 + F_{\text{ackley}}\left[z(P_{m+1} : P_D)\right]$$

Dimensión: $D = 1000$

Tamaño del grupo: $m = 50$

- $x = (x_1, x_2, \ldots, x_D)$: la solución candidata – un vector fila de dimensión D
- $o = (o_1, o_2, \ldots, o_D)$: el óptimo global (desplazado)
- $z = x - o, \quad z = (z_1, z_2, \ldots, z_D)$: la solución candidata desplazada – un vector fila de dimensión D

- P: una permutación aleatoria de $\{1, 2, \ldots, D\}$

Propiedades:

1. Multimodal
2. Desplazada
3. Rotada m-de un solo grupo
4. m-no separable de un solo grupo
5. $x \in [-32, 32]^D$
6. Óptimo global: $x^* = o, \quad F_6(x^*) = 0$

B.2.2.4. F7: Problema de Schwefel 1.2 desplazado m dimensional de un solo grupo

$$F_7(x) = F_{\text{schwefel}}\left[z(P_1 : P_m)\right] \times 10^6 + F_{\text{sphere}}\left[z(P_{m+1} : P_D)\right]$$

Dimensión: $D = 1000$

Tamaño del grupo: $m = 50$

- $x = (x_1, x_2, \ldots, x_D)$: la solución candidata – un vector fila de dimensión D
- $o = (o_1, o_2, \ldots, o_D)$: el óptimo global (desplazado)
- $z = x - o, \quad z = (z_1, z_2, \ldots, z_D)$: la solución candidata desplazada – un vector fila de dimensión D
- P: permutación aleatoria de $\{1, 2, \ldots, D\}$

Propiedades:

1. Unimodal
2. Desplazada
3. m-no separable de un solo grupo
4. $x \in [-100, 100]^D$
5. Óptimo global: $x^* = o, \quad F_7(x^*) = 0$

B.2.2.5. F8: Función de Rosenbrock desplazada m dimensional de un solo grupo

$$F_8(x) = F_{\text{rosenbrock}}\left[z(P_1 : P_m)\right] \times 10^6 + F_{\text{sphere}}\left[z(P_{m+1} : P_D)\right]$$

Dimensión: $D = 1000$

Tamaño del grupo: $m = 50$

- $x = (x_1, x_2, \ldots, x_D)$: la solución candidata – un vector fila de dimensión D
- $o = (o_1, o_2, \ldots, o_D)$: el óptimo global (desplazado)
- $z = x - o, \quad z = (z_1, z_2, \ldots, z_D)$: la solución candidata desplazada – un vector fila de dimensión D
- P: permutación aleatoria de $\{1, 2, \ldots, D\}$

Propiedades:

1. Multimodal

2. Desplazada
3. m-no separable de un solo grupo
4. $x \in [-100, 100]^D$
5. Óptimo global: $x^*(P_1 : P_m) = o(P_1 : P_m) + 1, \quad x^*(P_{m+1} : P_D) = o(P_{m+1} : P_D), \quad F_8(x^*) = 0$

B.2.3. Funciones m no separables grupo $\frac{D}{2m}$

B.2.3.1. F9: Función elíptica desplazada y m rotada de grupo $\frac{D}{2m}$

$$F_9(x) = \sum_{k=1}^{\frac{D}{2m}} F_{\text{rot elliptic}}\left[z\left(P\left((k-1)\times m+1\right) : P_k \times m\right)\right] + F_{\text{elliptic}}\left[z\left(P_{\frac{D}{2}+1} : P_D\right)\right]$$

Dimensión: $D = 1000$

Tamaño del grupo: $m = 50$

- $x = (x_1, x_2, \ldots, x_D)$: la solución candidata – un vector fila de dimensión D
- $o = (o_1, o_2, \ldots, o_D)$: el óptimo global (desplazado)
- $z = x - o, \quad z = (z_1, z_2, \ldots, z_D)$: la solución candidata desplazada – un vector fila de dimensión D
- P: permutación aleatoria de $\{1, 2, \ldots, D\}$

Propiedades:

1. Unimodal
2. Desplazada
3. $\frac{D}{2m}$-grupo m-rotado
4. $\frac{D}{2m}$-grupo m-no separable
5. $x \in [-100, 100]^D$
6. Óptimo global: $x^* = o, \quad F_9(x^*) = 0$

B.2.3.2. F10: Función Rastrigin desplazada y m rotada de grupo $\frac{D}{2m}$

$$F_{10}(x) = \sum_{k=1}^{\frac{D}{2m}} F_{\text{rot rastrigin}}\left[z\left(P\left((k-1)\times m+1\right) : P_k \times m\right)\right] + F_{\text{rastrigin}}\left[z\left(P_{\frac{D}{2}+1} : P_D\right)\right]$$

Dimensión: $D = 1000$

Tamaño del grupo: $m = 50$

- $x = (x_1, x_2, \ldots, x_D)$: la solución candidata – un vector fila de dimensión D
- $o = (o_1, o_2, \ldots, o_D)$: el óptimo global (desplazado)
- $z = x - o, \quad z = (z_1, z_2, \ldots, z_D)$: la solución candidata desplazada – un vector fila de dimensión D
- P: permutación aleatoria de $\{1, 2, \ldots, D\}$

Propiedades:

1. Multimodal

2. Desplazada
3. $\frac{D}{2m}$-grupo m-rotado
4. $\frac{D}{2m}$-grupo m-no separable
5. $x \in [-5, 5]^D$
6. Óptimo global: $x^* = o, \quad F_{10}(x^*) = 0$

B.2.3.3. F11: Función Ackley desplazada y m rotada de grupo $\frac{D}{2m}$

$$F_{11}(x) = \sum_{k=1}^{\frac{D}{2m}} F_{\text{rot ackley}}\left[z\left(P\left((k-1)\times m+1\right) : P_k \times m\right)\right] + F_{\text{ackley}}\left[z\left(P_{\frac{D}{2}+1} : P_D\right)\right]$$

Dimensión: $D = 1000$

Tamaño del grupo: $m = 50$

- $x = (x_1, x_2, \ldots, x_D)$: la solución candidata – un vector fila de dimensión D
- $o = (o_1, o_2, \ldots, o_D)$: el óptimo global (desplazado)
- $z = x - o, \quad z = (z_1, z_2, \ldots, z_D)$: la solución candidata desplazada – un vector fila de dimensión D
- P: permutación aleatoria de $\{1, 2, \ldots, D\}$

Propiedades:

1. Multimodal
2. Desplazada

3. $\frac{D}{2m}$-grupo m-rotado
4. $\frac{D}{2m}$-grupo m-no separable
5. $x \in [-32, 32]^D$
6. Óptimo global: $x^* = o, \quad F_{11}(x^*) = 0$

B.2.3.4. F12: Problema de Schwefel 1.2 desplazado y m rotado de grupo $\frac{D}{2m}$

$$F_{12}(x) = \sum_{k=1}^{\frac{D}{2m}} F_{\text{schwefel}}\left[z\left(P\left((k-1)\times m+1\right) : P_k \times m\right)\right] + F_{\text{sphere}}\left[z\left(P_{\frac{D}{2}+1} : P_D\right)\right]$$

Dimensión: $D = 1000$

Tamaño del grupo: $m = 50$

- $x = (x_1, x_2, \ldots, x_D)$: la solución candidata – un vector fila de dimensión D
- $o = (o_1, o_2, \ldots, o_D)$: el óptimo global (desplazado)
- $z = x - o, \quad z = (z_1, z_2, \ldots, z_D)$: la solución candidata desplazada – un vector fila de dimensión D
- P: permutación aleatoria de $\{1, 2, \ldots, D\}$

Propiedades:

1. Unimodal
2. Desplazada
3. $\frac{D}{2m}$-grupo m-no separable

4. $x \in [-100, 100]^D$
5. Óptimo global: $x^* = o, \quad F_{12}(x^*) = 0$

B.2.3.5. F13: Función de Rosenbrock desplazada y m rotada de grupo $\frac{D}{2m}$

$$F_{13}(x) = \sum_{k=1}^{\frac{D}{2m}} F_{\text{rosenbrock}}\left[z\left(P\left((k-1)\times m+1\right) : P_k \times m\right)\right] + F_{\text{sphere}}\left[z\left(P_{\frac{D}{2}+1} : P_D\right)\right]$$

Dimensión: $D = 1000$

Tamaño del grupo: $m = 50$

- $x = (x_1, x_2, \ldots, x_D)$: la solución candidata – un vector fila de dimensión D
- $o = (o_1, o_2, \ldots, o_D)$: el óptimo global (desplazado)
- $z = x - o, \quad z = (z_1, z_2, \ldots, z_D)$: la solución candidata desplazada – un vector fila de dimensión D
- P: permutación aleatoria de $\{1, 2, \ldots, D\}$

Propiedades:

1. Multimodal
2. Desplazada
3. $\frac{D}{2m}$-grupo m-no separable
4. $x \in [-100, 100]^D$
5. Óptimo global: $x^*(P_1 : P_{D/2}) = o(P_1 : P_{D/2}) + 1, \quad x^*(P_{D/2+1} : P_D) = o(P_{D/2+1} : P_D), \quad F_{13}(x^*) = 0$

B.2.4. Funciones m no separables grupo $\frac{D}{m}$

B.2.4.1. F14: Función elíptica desplazada y m rotada de grupo $\frac{D}{m}$

$$F_{14}(x) = \sum_{k=1}^{\frac{D}{m}} F_{\text{rot elliptic}} \left[z \left(P \left((k-1) \times m + 1 \right) : P_k \times m \right) \right]$$

Dimensión: $D = 1000$

Tamaño del grupo: $m = 50$

- $x = (x_1, x_2, \ldots, x_D)$: la solución candidata – un vector fila de dimensión D
- $o = (o_1, o_2, \ldots, o_D)$: el óptimo global (desplazado)
- $z = x - o, \quad z = (z_1, z_2, \ldots, z_D)$: la solución candidata desplazada – un vector fila de dimensión D
- P: permutación aleatoria de $\{1, 2, \ldots, D\}$

Propiedades:

1. Unimodal
2. Desplazada
3. $\frac{D}{m}$-grupo m-rotado
4. $\frac{D}{m}$-grupo m-no separable
5. $x \in [-100, 100]^D$
6. Óptimo global: $x^* = o, \quad F_{14}(x^*) = 0$

B.2.4.2. F15: Función Rastrigin desplazada y m rotada de grupo $\frac{D}{m}$

$$F_{15}(x) = \sum_{k=1}^{\frac{D}{m}} F_{\text{rot rastrigin}} \left[z \left(P \left((k-1) \times m + 1 \right) : P_k \times m \right) \right]$$

Dimensión: $D = 1000$

Tamaño del grupo: $m = 50$

- $x = (x_1, x_2, \ldots, x_D)$: la solución candidata – un vector fila de dimensión D
- $o = (o_1, o_2, \ldots, o_D)$: el óptimo global (desplazado)
- $z = x - o, \quad z = (z_1, z_2, \ldots, z_D)$: la solución candidata desplazada – un vector fila de dimensión D
- P: permutación aleatoria de $\{1, 2, \ldots, D\}$

Propiedades:

1. Multimodal
2. Desplazada
3. $\frac{D}{m}$-grupo m-rotado
4. $\frac{D}{m}$-grupo m-no separable
5. $x \in [-5, 5]^D$
6. Óptimo global: $x^* = o, \quad F_{15}(x^*) = 0$

B.2.4.3. F16: Función Ackley desplazada y m rotada de grupo $\frac{D}{m}$

$$F_{16}(x) = \sum_{k=1}^{\frac{D}{m}} F_{\text{rot ackley}}\left[z\left(P\left((k-1)\times m+1\right):P_k\times m\right)\right]$$

Dimensión: $D = 1000$

Tamaño del grupo: $m = 50$

- $x = (x_1, x_2, \ldots, x_D)$: la solución candidata – un vector fila de dimensión D
- $o = (o_1, o_2, \ldots, o_D)$: el óptimo global (desplazado)
- $z = x - o, \quad z = (z_1, z_2, \ldots, z_D)$: la solución candidata desplazada – un vector fila de dimensión D
- P: permutación aleatoria de $\{1, 2, \ldots, D\}$

Propiedades:

1. Multimodal
2. Desplazada
3. $\frac{D}{m}$-grupo m-rotado
4. $\frac{D}{m}$-grupo m-no separable
5. $x \in [-32, 32]^D$
6. Óptimo global: $x^* = o, \quad F_{16}(x^*) = 0$

B.2.4.4. F17: Problema de Schwefel desplazado y m dimensional de grupo $\frac{D}{m}$

$$F_{17}(x) = \sum_{k=1}^{\frac{D}{m}} F_{\text{schwefel}}\left[z\left(P\left((k-1)\times m+1\right):P_k\times m\right)\right]$$

Dimensión: $D = 1000$

Tamaño del grupo: $m = 50$

- $x = (x_1, x_2, \ldots, x_D)$: la solución candidata – un vector fila de dimensión D
- $o = (o_1, o_2, \ldots, o_D)$: el óptimo global (desplazado)
- $z = x - o, \quad z = (z_1, z_2, \ldots, z_D)$: la solución candidata desplazada – un vector fila de dimensión D
- P: permutación aleatoria de $\{1, 2, \ldots, D\}$

Propiedades:

1. Unimodal
2. Desplazada
3. $\frac{D}{m}$-grupo m-no separable
4. $x \in [-100, 100]^D$
5. Óptimo global: $x^* = o, \quad F_{17}(x^*) = 0$

B.2.4.5. F18: Función de Rosenbrock desplazada y m dimensional de grupo $\frac{D}{m}$

$$F_{18}(x) = \sum_{k=1}^{\frac{D}{m}} F_{\text{rosenbrock}}\left[z\left(P\left((k-1)\times m+1\right):P_k\times m\right)\right]$$

Dimensión: $D = 1000$

Tamaño del grupo: $m = 50$

- $x = (x_1, x_2, \ldots, x_D)$: la solución candidata – un vector fila de dimensión D
- $o = (o_1, o_2, \ldots, o_D)$: el óptimo global (desplazado)
- $z = x - o, \quad z = (z_1, z_2, \ldots, z_D)$: la solución candidata desplazada – un vector fila de dimensión D
- P: permutación aleatoria de $\{1, 2, \ldots, D\}$

Propiedades:

1. Multimodal
2. Desplazada
3. $\frac{D}{m}$-grupo m-no separable
4. $x \in [-100, 100]^D$
5. Óptimo global: $x^* = o + 1, \quad F_{18}(x^*) = 0$

B.2.5. Funciones no separables

B.2.5.1. F19: Problema de Schwefel desplazado 1.2

$$F_{19}(x) = F_{\text{schwefel}}(z) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_i \right)^2$$

Dimensión: $D = 1000$

- $x = (x_1, x_2, \ldots, x_D)$: la solución candidata – un vector fila de dimensión D
- $o = (o_1, o_2, \ldots, o_D)$: el óptimo global (desplazado)
- $z = x - o, \quad z = (z_1, z_2, \ldots, z_D)$: la solución candidata desplazada – un vector fila de dimensión D

Propiedades:

1. Unimodal
2. Desplazada
3. Totalmente no separable
4. $x \in [-100, 100]^D$
5. Óptimo global: $x^* = o, \quad F_{19}(x^*) = 0$

B.2.5.2. F20: Función de Rosenbrock desplazada

$$F_{20}(x) = F_{\text{rosenbrock}}(z) = \sum_{i=1}^{D-1} \left[100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2 \right]$$

Dimensión: $D = 1000$

- $x = (x_1, x_2, \ldots, x_D)$: la solución candidata – un vector fila de dimensión D
- $o = (o_1, o_2, \ldots, o_D)$: el óptimo global (desplazado)
- $z = x - o, \quad z = (z_1, z_2, \ldots, z_D)$: la solución candidata desplazada – un vector fila de dimensión D

Propiedades:

1. Multimodal
2. Desplazada
3. Totalmente no separable
4. $x \in [-100, 100]^D$
5. Óptimo global: $x^* = o + 1, \quad F_{20}(x^*) = 0$

Apéndice B. Definiciones de los Problemas de Prueba del CEC'2010

Apéndice C

Definiciones de los Problemas de Prueba del CEC'2013

C.1. Funciones base

C.1.1. Función esfera

$$f_{\text{sphere}}(x) = \sum_{i=1}^{D} x_i^2$$

donde x es un vector de decisión de D dimensiones. La función *esfera* es una función unimodal y completamente separable que se utiliza como subcomponente completamente separable para algunas de las funciones parcialmente separables definidas en este conjunto.

C.1.2. Función elíptica

$$f_{\text{elliptic}}(x) = \sum_{i=1}^{D} 10^{6(D-i)} x_i^2$$

C.1.3. Función de Rastrigin

$$f_{\text{rastrigin}}(x) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$$

C.1.4. Función de Ackley

$$f_{\text{ackley}}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + \exp(1)$$

C.1.5. Problema de Schwefel 1.2

$$f_{\text{schwefel}}(x) = \sum_{i=1}^{D}\left(\sum_{j=1}^{i} x_j\right)^2$$

C.1.6. Función de Rosenbrock

$$f_{\text{rosenbrock}}(x) = \sum_{i=1}^{D-1} \left(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2\right)$$

C.2. Diseño

C.2.1. Símbolos

A continuación se describen los símbolos y nomenclaturas utilizadas en este apéndice. Los vectores se escriben en minúsculas en negrita y representan vectores columna (por ejemplo, $\mathbf{x} = (x_1, \ldots, x_D)^T$). Las matrices se escriben en mayúsculas en negrita (por ejemplo, $\mathbf{R}$).

- $\mathbf{S}$: Un multiconjunto que contiene los tamaños de los subcomponentes para una función. Por ejemplo, $S = \{50, 25, 50, 100\}$ indica que hay 4 subcomponentes con 50, 25, 50 y 100 variables de decisión, respectivamente.
- $|S|$: Número de elementos en S. Representa la cantidad de subcomponentes en una función.
- $C_i = \sum_{j=1}^{i} S_j$: La suma de los primeros i elementos de S. Para conveniencia, se define C_0 como cero. C_i se utiliza para construir el vector de decisión de diferentes funciones de subcomponentes con el tamaño correcto.
- D: La dimensionalidad de la función objetivo.
- P: Una permutación aleatoria de los índices de las dimensiones $\{1, \ldots, D\}$.
- w_i: Un peso generado aleatoriamente que se utiliza como el coeficiente del i-ésimo subcomponente no separable para generar el efecto de desequilibrio. Los pesos se generan de la siguiente manera:

  $$w_i = 10^{3N(0,1)},$$

  donde $N(0, 1)$ es una distribución gaussiana con media cero y varianza unitaria.

- x^{opt}: El vector de decisión óptimo para el cual el valor de la función objetivo es mínimo. Éste también se utiliza como un vector de desplazamiento para cambiar la ubicación del óptimo global.

- T_{osz}: Una función de transformación para crear irregularidades locales suaves [70].

  $T_{\text{osz}} : \mathbb{R}^D \to \mathbb{R}^D, \quad x_i \mapsto \text{sign}(x_i) \exp(\hat{x}_i + 0.049(sen(c_1\hat{x}_i) + sen(c_2\hat{x}_i))), \quad \text{para } i = 1, \ldots, D$

  donde:

$$\hat{x}_i = \begin{cases} \log(|x_i|) & \text{si } x_i \neq 0, \\ 0 & \text{de lo contrario,} \end{cases}$$

$$\text{sign}(x) = \begin{cases} -1 & \text{si } x < 0, \\ 0 & \text{si } x = 0, \\ 1 & \text{si } x > 0, \end{cases}$$

$$c_1 = \begin{cases} 10 & \text{si } x_i > 0, \\ 5.5 & \text{de otra manera} \end{cases} \qquad c_2 = \begin{cases} 7.9 & \text{si } x_i > 0, \\ 3.1 & \text{de lo contrario} \end{cases}$$

- T^{β}_{asy}: Una función de transformación para romper la simetría de las funciones simétricas [70].

$$T^{\beta}_{\text{asy}} : \mathbb{R}^D \to \mathbb{R}^D, \quad x_i \mapsto \begin{cases} x_i^{1+\beta \frac{i-1}{\sqrt{D-1}} \sqrt{x_i}} & \text{si } x_i > 0, \\ x_i & \text{de lo contrario} \end{cases}$$

  para $i = 1, \ldots, D$.

- Λ^{α}: Una matriz de D-dimensiones con elementos diagonales $\lambda_{ii} = \alpha^{\frac{1}{2} \frac{i-1}{D-1}}$. Esta matriz se utiliza para crear mal condicionamiento [70]. El parámetro α es el número de condición.

- $\mathbf{R}$: Una matriz de rotación ortogonal que se utiliza para rotar aleatoriamente el paisaje de aptitud alrededor de varios ejes, como se sugiere en [71].

- m: El tamaño de solapamiento entre subcomponentes.

- $\mathbf{1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$: Un vector columna de unos.

C.2.2. Diseño de las funciones parcialmente separables

Este tipo de funciones tienen la siguiente forma general:

$$f(\mathbf{x}) = \sum_{i=1}^{|S|-1} w_i f_{\text{nonsep}}(\mathbf{z}_i) + f_{\text{sep}}(\mathbf{z}_{|S|}),$$

donde w_i es un peso generado aleatoriamente para crear el efecto de desequilibrio, y f_{sep} es una función que puede ser la función Esfera o la versión no rotada de las funciones de Rastrigin o Ackley. Para generar una versión no separable de estas funciones, puede usarse una matriz de rotación.

El vector $\mathbf{z}$ se forma transformando, desplazando y finalmente reordenando las dimensiones del vector x. Una transformación típica se muestra a continuación:

$$\mathbf{y} = \Lambda^{10} T_{\text{asy}}^{0.2}(T_{\text{osz}}(\mathbf{x} - \mathbf{x}^{\text{opt}})),$$

$$\mathbf{z_i} = \mathbf{y}(P_{[C_{i-1}+1]} : P_{[C_i]}),$$

Como se describió anteriormente, el vector x^{opt} es la ubicación del óptimo desplazado, que se utiliza como vector de desplazamiento. El conjunto de permutación P se utiliza para modificar el orden de las variables de decisión, y C_i se usa para construir cada uno de los vectores de subcomponentes (z_i) con el tamaño correspondiente (S_i) especificado en el multiconjunto S.

C.3. Definición de los problemas de prueba

C.3.1. Funciones completamente separables

f1: Función *elíptica* desplazada

$$f_1(\mathbf{z}) = \sum_{j=1}^{D} 10^{6\frac{j-1}{D-1}} z_j^2$$

- $\mathbf{z} = T_{osz}(\mathbf{x} - \mathbf{x}^{opt})$
- $\mathbf{x} \in [-100, 100]^D$
- **Óptimo global**: $f_1(\mathbf{x}^{opt}) = 0$

Propiedades:

- Unimodal
- Separable
- Desplazada
- Irregularidades locales suaves
- Problema mal condicionado. Es muy sensible a errores de redondeo y pequeñas perturbaciones, lo que hace que la solución sea más difícil de encontrar o menos precisa (número de condición $\approx 10^6$) .

f_2: Función *Rastrigin* desplazada

$$f_2(\mathbf{z}) = \sum_{i=1}^{D} \left[z_i^2 - 10\cos(2\pi z_i) + 10\right]$$

- $\mathbf{z} = \Lambda^{10} T_{asy}^{0.2}(T_{osz}(\mathbf{x} - \mathbf{x}^{opt}))$
- $\mathbf{x} \in [-5, 5]^D$
- **Óptimo global**: $f_2(\mathbf{x}^{opt}) = 0$

Propiedades:

- Multimodal
- Separable
- Desplazada
- Irregularidades locales suaves
- Problema mal condicionado. Es muy sensible a errores de redondeo y pequeñas perturbaciones, lo que hace que la solución sea más difícil de encontrar o menos precisa (número de condición ≈ 10)

f_3: Función *Ackley* desplazada

$$f_3(\mathbf{z}) = -20 \exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)\right) + 20 + e$$

- $\mathbf{z} = \Lambda^{10} T_{asy}^{0.2}(T_{osz}(\mathbf{x} - \mathbf{x}^{opt}))$
- $\mathbf{x} \in [-32, 32]^D$
- **Óptimo global**: $f_3(\mathbf{x}^{opt}) = 0$

Propiedades:

- Multimodal
- Separable

- Desplazada
- Irregularidades locales suaves
- Mal condicionado (número de condición ≈ 10)

C.3.2. Funciones parcialmente aditivas separables I

f_4: Función *elíptica* desplazada y rotada de 7-no separable, 1-separable

$$f_4(\mathbf{z}) = \sum_{i=1}^{|S|-1} w_i f_{elliptic}(\mathbf{z}_i) + f_{elliptic}(\mathbf{z}_{|S|})$$

- $\mathbf{S} = \{50, 25, 25, 100, 50, 25, 25, 700\}$
- $D = \sum_{i=1}^{|S|} S_i = 1000$
- $\mathbf{y} = \mathbf{x} - \mathbf{x}^{opt}$
- $\mathbf{y}_i = \mathbf{y}(P[Ci - 1 + 1] : P[Ci])$, $i \in \{1, \ldots, |S|\}$
- $\mathbf{z_i} = T_{osz}(R_i y_i)$, $i \in \{1, \ldots, |S| - 1\}$
- $\mathbf{z}_{|S|} = T_{osz}(\mathbf{y}_{|S|})$
- $\mathbf{R}_i$: una matriz de rotación de $|S_i| \times |S_i|$
- $\mathbf{x} \in [-100, 100]^D$
- **Óptimo global**: $f_4(\mathbf{x}^{opt}) = 0$

Propiedades:

- Unimodal
- Parcialmente separable
- Desplazada

- Irregularidades locales suaves
- Problema mal condicionado (número de condición $\approx 10^6$)

f5: Función *Rastrigin* Desplazada y Rotada de 7-no separable, 1-separable

$$f_5(\mathbf{z}) = \sum_{i=1}^{|S|-1} w_i f_{rastrigin}(z_i) + f_{rastrigin}(z_{|S|})$$

- $\mathbf{S} = \{50, 25, 25, 100, 50, 25, 25, 700\}$
- $D = \sum_{i=1}^{|S|} S_i = 1000$
- $\mathbf{y} = \mathbf{x} - \mathbf{x}^{opt}$
- $y_i = y(P[Ci - 1 + 1] : P[Ci])$, $i \in \{1, \ldots, |S|\}$
- $z_i = \Lambda_{10} T_{asy}(T_{osz}(R_i y_i))$, $i \in \{1, \ldots, |S| - 1\}$
- $z_{|S|} = \Lambda_{10} T_{asy}(T_{osz}(y_{|S|}))$
- R_i: una matriz de rotación de $|S_i| \times |S_i|$
- $\mathbf{x} \in [-5, 5]^D$
- **Óptimo global**: $f_5(\mathbf{x}^{opt}) = 0$

Propiedades:

- Multimodal
- Parcialmente separable
- Desplazada
- Irregularidades locales suaves
- Problema mal condicionado (número de condición ≈ 10)

f6: Función *Ackley* Desplazada y Rotada de 7-no separable, 1-separable

$$f_6(\mathbf{z}) = \sum_{i=1}^{|S|-1} w_i f_{ackley}(z_i) + f_{ackley}(z_{|S|})$$

- $\mathbf{S} = \{50, 25, 25, 100, 50, 25, 25, 700\}$
- $D = \sum_{i=1}^{|S|} S_i = 1000$
- $\mathbf{y} = \mathbf{x} - \mathbf{x}^{opt}$
- $y_i = y(P[Ci - 1 + 1] : P[Ci]),\ i \in \{1, \ldots, |S|\}$
- $z_i = \Lambda_{10} T_{asy}(T_{osz}(R_i y_i)),\ i \in \{1, \ldots, |S| - 1\}$
- $z_{|S|} = \Lambda_{10} T_{asy}(T_{osz}(y_{|S|}))$
- R_i: una matriz de rotación de $|S_i| \times |S_i|$
- $\mathbf{x} \in [-32, 32]^D$
- **Óptimo global**: $f_6(\mathbf{x}^{opt}) = 0$

Propiedades:

- Multimodal
- Parcialmente separable
- Desplazada
- Irregularidades locales suaves
- Problema mal condicionado (número de condición ≈ 10)

f7: Función *Schwefel* Desplazada y Rotada de 7-no separable, 1-separable

$$f_7(\mathbf{z}) = \sum_{i=1}^{|S|-1} w_i f_{schwefel}(z_i) + f_{sphere}(z_{|S|})$$

- $\mathbf{S} = \{50, 25, 25, 100, 50, 25, 25, 700\}$
- $D = \sum_{i=1}^{|S|} S_i = 1000$
- $\mathbf{y} = \mathbf{x} - \mathbf{x}^{opt}$
- $y_i = y(P[Ci - 1 + 1] : P[Ci])$, $i \in \{1, \ldots, |S|\}$
- $z_i = T_{asy}(T_{osz}(R_i y_i))$, $i \in \{1, \ldots, |S| - 1\}$
- $z_{|S|} = T_{asy}(T_{osz}(y_{|S|}))$
- R_i: una matriz de rotación de $|S_i| \times |S_i|$
- $\mathbf{x} \in [-100, 100]^D$
- **Óptimo global**: $f_7(\mathbf{x}^{opt}) = 0$

Propiedades:

- Multimodal
- Parcialmente separable
- Desplazada
- Irregularidades locales suaves

C.3.3. Funciones parcialmente aditivamente separables II

f8: Función *elíptica* desplazada y rotada de 20-no separable

$$f_8(\mathbf{z}) = \sum_{i=1}^{|S|} w_i f_{elliptic}(z_i)$$

- $\mathbf{S} = \{50, 50, 25, 25, 100, 100, 25, 25, 50, 25, 100, 25, 100, 50, 25, 25, 25, 100, 50, 25\}$
- $D = \sum_{i=1}^{|S|} S_i = 1000$

- $\mathbf{y} = \mathbf{x} - \mathbf{x}^{opt}$
- $y_i = y(P[Ci - 1 + 1] : P[Ci])$, $i \in \{1, \ldots, |S|\}$
- $z_i = T_{osz}(R_i y_i)$, $i \in \{1, \ldots, |S|\}$
- R_i: una matriz de rotación de $|S_i| \times |S_i|$
- $\mathbf{x} \in [-100, 100]^D$
- **Óptimo global**: $f_8(\mathbf{x}^{opt}) = 0$

Propiedades:

- Unimodal
- Parcialmente separable
- Desplazada
- Irregularidades locales suaves
- Problema mal condicionado (número de condición $\approx 10^6$)

f9: Función *Rastrigin* Desplazada y Rotada de 20-no separable

$$f_9(\mathbf{z}) = \sum_{i=1}^{|S|} w_i f_{rastrigin}(z_i)$$

- $\mathbf{S} = \{50, 50, 25, 25, 100, 100, 25, 25, 50, 25, 100, 25, 100, 50, 25, 25, 25, 100, 50, 25\}$
- $D = \sum_{i=1}^{|S|} S_i = 1000$
- $\mathbf{y} = \mathbf{x} - \mathbf{x}^{opt}$
- $y_i = y(P[Ci - 1 + 1] : P[Ci])$, $i \in \{1, \ldots, |S|\}$
- $z_i = \Lambda_{10} T_{asy}(T_{osz}(R_i y_i))$, $i \in \{1, \ldots, |S|\}$
- R_i: una matriz de rotación de $|S_i| \times |S_i|$

- $\mathbf{x} \in [-5, 5]^D$
- **Óptimo global**: $f_9(\mathbf{x}^{opt}) = 0$

Propiedades:

- Multimodal
- Parcialmente separable
- Desplazada
- Irregularidades locales suaves
- Problema mal condicionado (número de condición ≈ 10)

f10: Función *Ackley* Desplazada y Rotada de 20-no separable

$$f_{10}(\mathbf{z}) = \sum_{i=1}^{|S|} w_i f_{ackley}(z_i)$$

- $\mathbf{S} = \{50, 50, 25, 25, 100, 100, 25, 25, 50, 25, 100, 25, 100, 50, 25, 25, 25, 100, 50, 25\}$
- $D = \sum_{i=1}^{|S|} S_i = 1000$
- $\mathbf{y} = \mathbf{x} - \mathbf{x}^{opt}$
- $y_i = y(P[Ci - 1 + 1] : P[Ci])$, $i \in \{1, \ldots, |S|\}$
- $z_i = \Lambda_{10} T_{asy}(T_{osz}(R_i y_i))$, $i \in \{1, \ldots, |S|\}$
- R_i: una matriz de rotación de $|S_i| \times |S_i|$
- $\mathbf{x} \in [-32, 32]^D$
- **Óptimo global**: $f_{10}(\mathbf{x}^{opt}) = 0$

Propiedades:

- Multimodal
- Parcialmente separable
- Desplazada
- Irregularidades locales suaves
- Problema mal condicionado (número de condición ≈ 10)

f11: Función *Schwefel* Desplazada y Rotada de 20-no separable

$$f_{11}(\mathbf{z}) = \sum_{i=1}^{|S|} w_i f_{schwefel}(z_i)$$

- $\mathbf{S} = \{50, 50, 25, 25, 100, 100, 25, 25, 50, 25, 100, 25, 100, 50, 25, 25, 25, 100, 50, 25\}$
- $D = \sum_{i=1}^{|S|} S_i = 1000$
- $\mathbf{y} = \mathbf{x} - \mathbf{x}^{opt}$
- $y_i = y(P[Ci - 1 + 1] : P[Ci])$, $i \in \{1, \ldots, |S|\}$
- $z_i = T_{asy}(T_{osz}(R_i y_i))$, $i \in \{1, \ldots, |S|\}$
- R_i: una matriz de rotación de $|S_i| \times |S_i|$
- $\mathbf{x} \in [-100, 100]^D$
- **Óptimo global**: $f_{11}(\mathbf{x}^{opt}) = 0$

Propiedades:

- Unimodal
- Parcialmente separable
- Desplazada
- Irregularidades locales suaves

C.3.4. Funciones superpuestas

f12: Función *Rosenbrock* Desplazada

$$f_{12}(\mathbf{z}) = \sum_{i=1}^{D-1} \left[100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2\right]$$

- $D = 1000$
- $\mathbf{x} \in [-100, 100]^D$
- **Óptimo global**: $f_{12}(\mathbf{x}^{opt} + 1) = 0$

Propiedades:

- Multimodal
- Separable
- Desplazada
- Irregularidades locales suaves

f13: Función *Schwefel* Desplazada con Subcomponentes Superpuestos Conforme

$$f_{13}(\mathbf{z}) = \sum_{i=1}^{|S|} w_i f_{\text{schwefel}}(z_i)$$

- $S = \{50, 50, 25, 25, 100, 100, 25, 25, 50, 25, 100, 25, 100, 50, 25, 25, 25, 100, 50, 25\}$
- $C_i = \sum_{j=1}^{i} S_j$, $C_0 = 0$
- $D = \sum_{i=1}^{|S|} S_i - m(|S| - 1) = 905$
- $\mathbf{y} = \mathbf{x} - \mathbf{x}^{opt}$

- $y_i = \mathbf{y}(P[C_{i-1} - (i-1)m + 1 : C_i - (i-1)m])$, $i \in \{1, \ldots, |S|\}$
- $z_i = \Lambda_{10}\text{Tasy}(\text{Tosz}(R_i y_i))$, $i \in \{1, \ldots, |S| - 1\}$
- $m = 5$: tamaño de solapamiento
- R_i: matriz de rotación $|S_i| \times |S_i|$
- $\mathbf{x} \in [-100, 100]^D$
- **Óptimo global**: $f_{13}(\mathbf{x}^{opt}) = 0$

Propiedades:

- Unimodal
- No-separable
- Solapada
- Desplazada
- Irregularidades locales suaves

f14: Función *Schwefel* Desplazada con Subcomponentes Superpuestos Conflictivos

$$f_{14}(\mathbf{z}) = \sum_{i=1}^{|S|} w_i f_{\text{schwefel}}(z_i)$$

- $S = \{50, 50, 25, 25, 100, 100, 25, 25, 50, 25, 100, 25, 100, 50, 25, 25, 25, 100, 50, 25\}$
- $D = \sum_{i=1}^{|S|} S_i - (m(|S| - 1)) = 905$
- $y_i = \mathbf{x}(P[C_{i-1} - (i-1)m + 1 : C_i - (i-1)m]) - \mathbf{x}_i^{opt}$
- $\mathbf{x}_i^{opt}$: vector de desplazamiento de tamaño $|S_i|$ para el i-ésimo subcomponente
- $z_i = \Lambda_{10}\text{Tasy}(\text{Tosz}(R_i y_i))$
- $m = 5$: tamaño de solapamiento

- R_i: matriz de rotación $|S_i| \times |S_i|$
- $\mathbf{x} \in [-100, 100]^D$
- **Óptimo global**: $f_{14}(\mathbf{x}^{opt}) = 0$

Propiedades:

- Unimodal
- No-separable
- Subcomponentes conflictivos
- Desplazada
- Irregularidades locales suaves

C.3.5. Funciones totalmente no separables

f15: Función *Schwefel* Desplazada

$$f_{15}(\mathbf{z}) = \sum_{i=1}^{D} w_i f_{\text{schwefel}}(z_i)$$

- $D = 1000$
- $z = \Lambda_{10}\text{Tasy}(\text{Tosz}(\mathbf{x} - \mathbf{x}^{opt}))$
- $\mathbf{x} \in [-100, 100]^D$
- **Óptimo global**: $f_{15}(\mathbf{x}^{opt}) = 0$

Propiedades:

- Unimodal
- Totalmente no-separable
- Desplazada
- Irregularidades locales suaves

Bibliografía

[1] Y.-W. Shang and Y.-H. Qiu. A note on the extended rosenbrock fufnction. *Evolutionary Computation*, 14(1):119–126, 2006.

[2] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, 1995.

[3] Shiqin Wang, Zhihua Cai, Yong Zhou, Junhui Wang, and Weicai Zhang. Multiple-strategy learning particle swarm optimization for large-scale optimization problems. *Information Sciences*, 316:487–502, 2015.

[4] Ran Cheng and Yaochu Jin. Competitive swarm optimizer: an efficient global optimizer for large-scale optimization. *IEEE Transactions on Cybernetics*, 45(2):191–204, 2015.

[5] Ran Cheng, Yaochu Jin, Markus Olhofer, and Thomas Bäck. A social learning particle swarm optimization algorithm for scalable optimization. *Information Sciences*, 291:43–60, 2016.

[6] Edmund K. Burke and Graham Kendall, editors. *Search Methodologies*. Springer, New York, NY, 2nd edition, 2014. ISBN: 978-1-4614-6940-7.

[7] Introducción a la Computación Evolutiva (Notas de Curso), 2025. Consultado el 11 de enero de 2025.

[8] C. Reeves. *Modern Heuristic Search Methods*. Blackwell Publishing, Oxford, UK, 1996.

[9] Jian Liu and Jouni Lampinen. Particle swarm optimization with gaussian mutation. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pages 725–730. IEEE, 2004.

[10] Carlos A. Coello Coello. Introducción a la computación evolutiva. `https://delta.cs.cinvestav.mx/~ccoello/compevol/`. Último acceso: 23 de enero de 2025.

[11] Lawrence J. Fogel, A. J. Owens, and M. J. Walsh. Artificial intelligence through a simulation of evolution. In M. Maxfield, A. Callahan, and L. J. Fogel, editors, *Biophysics and Cybernetic Systems: Proceedings of the Second Cybernetic Sciences Symposium*, pages 131–155, Washington, D.C., 1965. Spartan Books.

[12] G. H. Burgin. On playing two-person zero-sum games against non-minimax players. *IEEE Transactions on Systems Science and Cybernetics*, SSC-5:369–370, 1969.

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

[14] W. Burks. Computation, behavior and structure in fixed and growing automata. In M. C. Yovits and S. Cameron, editors, *Self-Organizing Systems*, pages 282–309. Pergamon Press, New York, 1960.

[15] O. G. Selfridge. Pandemonium: A paradigm for learning. In *Proceedings of the Symposium on Mechanization of Thought Processes*, pages 511–529, Teddington, England, 1958.

[16] Terence C. Fogarty. Varying the probability of mutation in the genetic algorithm. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 104–109, San Mateo, California, 1989. Morgan Kaufmann Publishers.

[17] James Kennedy, Russell C. Eberhart, and Yuhui Shi. *Swarm Intelligence*. Morgan Kaufmann, San Francisco, 2001.

[18] James Kennedy. The behavior of particles in swarm optimization. In *Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics*, volume 5, pages 4045–4050. IEEE, 1998.

[19] Leon Festinger. *A Theory of Cognitive Dissonance*. Stanford University Press, Stanford, CA, 1954.

[20] Konrad Lorenz. *The Natural Science of the Human Species*. Harvard University Press, Cambridge, MA, 1980.

[21] Konrad Lorenz. *King Solomon's Ring: New Light on Animal Ways*. Methuen Publishing Ltd, London, 1952. Translated by Marjorie Kerr Wilson.

[22] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, 1995.

[23] James Kennedy and Russell Eberhart. A discrete binary version of the particle swarm algorithm. In *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation*, pages 410–413. IEEE, 1997.

[24] M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, 2002.

[25] Frans Van den Bergh. *An Analysis of Particle Swarm Optimizers*. PhD thesis, University of Pretoria, May 2007. PhD thesis, South Africa, 2007.

[26] Sara Grassi and Lorenzo Pareschi. From particle swarm optimization to consensus based optimization: stochastic modeling and mean-field limit, 2020.

[27] Hui Huang, Jinniao Qiu, and Konstantin Riedl. On the global convergence of particle swarm optimization methods. *Applied Mathematics & Optimization*, 88(2):30, May 2023.

[28] Xiao-Lin Li, Roger Serra, and Julien Olivier. An investigation of particle swarm optimization topologies in structural damage detection. *Applied Sciences*, 11(11), 2021.

[29] Shi Cheng, Yuhui Shi, and Quande Qin. Population diversity based study on search information propagation in particle swarm optimization. In *2012 IEEE Congress on Evolutionary Computation*. IEEE, 2012.

[30] Rethabile Mabaso and Christopher W. Cleghorn. Topology-linked self-adaptive quantum particle swarm optimization for dynamic environments. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2020.

[31] J. Kennedy and R. Mendes. Neighborhood topologies in fully informed and best-of-neighborhood particle swarms. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 36(4):515–519, 2006.

[32] Dehua Peng, Zhipeng Gui, and Huayi Wu. Interpreting the curse of dimensionality from distance concentration and manifold effect. arxiv, 2024.

[33] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[34] Roman Vershynin. *High-Dimensional Probability: The Geometry of Large-Scale Data*. Cambridge University Press, Cambridge, 2018.

[35] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, 2nd edition, 2009.

[36] Hervé Abdi and Lynne J. Williams. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):433–459, 2010.

[37] Maurice Clerc and James Kennedy. Parameter selection in particle swarm optimization. *Evolutionary Computation*, 6(2):115–129, 2002.

[38] Zhi-Hui Zhan, Jun Zhang, Yun Li, and Henry Shu-Hung Chung. An adaptive particle swarm optimization with multiple adaptive methods. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(6):1362–1381, 2009.

[39] S. A. Hamdan. Hybrid particle swarm optimiser using multi-neighborhood topologies. *INFOCOMP Journal of Computer Science*, 7(2):39–44, 2008.

[40] X. S. Yang. Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization. *International Journal of Computational Intelligence and Applications*, 9(2):245–261, 2010.

[41] Ricardo Mendes, James Kennedy, and J. Neves. The fully informed particle swarm: Simpler, maybe better. In *IEEE Transactions on Evolutionary Computation*, volume 8, pages 204–210. IEEE, 2004.

[42] Jing J. Liang, Akshay K. Qin, Ponnuthurai N. Suganthan, and Subramanian Baskar. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation*, 10(3):281–295, 2006.

[43] Xiang Zhang, Liang Zhang, Jun Zhao, and Li Li. Improving particle swarm optimization performance with local search for high-dimensional function optimization. *IEEE Transactions on Evolutionary Computation*, 18(5):735–748, 2014.

[44] J. Yang, C. Liu, X. Li, L. Xie, and L. Zhang. Cooperatively coevolving particle swarms for large scale optimization. *IEEE Transactions on Evolutionary Computation*, 17(4):464–476, 2013.

[45] Y. Chen, X. Zhang, and L. Zhang. A transfer learning-based particle swarm optimization algorithm for traveling salesman problem. *Computers And Operations Research*, 85:138–148, 2017.

[46] Xin-She Yang. Chaos-enhanced accelerated particle swarm optimization. *Applied Soft Computing*, 11(8):7741–7750, 2012.

[47] James Kennedy and Ricardo Mendes. Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC '03)*, volume 3, pages 1931–1938. IEEE, 2003.

[48] N. Hansen and S. Kern. Evaluating the cma evolution strategy on multimodal test functions. In *Proceedings of PPSN VIII*, pages 282–291. Springer, 2004.

[49] A. Auger and N. Hansen. A restart cma evolution strategy with increasing population size. In *Proceedings of IEEE Congress on Evolutionary Computation*, volume 2, pages 1769–1776. IEEE, 2005.

[50] M. Lozano, D. Molina, and F. Herrera. Editorial scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems. *Soft Computing*, 15(11):2085–2087, 2011.

[51] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar. Problem definitions and evaluation criteria for the cec 2008 large-scale global optimization competition. Technical report, Nanyang Technological University, 2008.

[52] J. E. Dennis and Robert B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Society for Industrial and Applied Mathematics, 1996. ISBN: 978-0-898713-64-0.

[53] Ke Tang, Xiaodong Li, P. N. Suganthan, Zhenyu Yang, and Thomas Weise. Benchmark functions for the cec'2010 special session and competition on large-scale global optimization. *Nature Inspired Computation and Applications Laboratory (NICAL), School of Computer Science and Technology, University of Science and Technology of China*, January 8 2010. Available at `http://al-roomi.org/multimedia/CEC_Database/CEC2010/LargeScaleGlobalOptimization/CEC2010_LargeScaleGO_TechnicalReport.pdf`.

[54] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. P. Chen, C. M. Chen, and Z. Yang. Benchmark functions for the cec'2008 special session and competition on large scale global optimization. Nature Inspired Computation and Applications Laboratory, USTC, China, 2007. Technical Report, Available online (Código fuente).

[55] *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2008, June 1-6, 2008, Hong Kong, China*. IEEE, 2008.

[56] Sheng-Ta Hsieh, Tsung-Ying Sun, Chan-Cheng Liu, and Shang-Jeng Tsai. Efficient population utilization strategy for particle swarm optimizer. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):444–456, 2009.

[57] S. Z. Zhao, J. J. Liang, P. N. Suganthan, and M. F. Tasgetiren. Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 3845–3852. IEEE, 2008.

[58] Lin-Yu Tseng and Chun Chen. Multiple trajectory search for multiobjective optimization. In *2007 IEEE Congress on Evolutionary Computation*, pages 3609–3616. IEEE, 2007.

[59] Ali R. Al-Roomi. IEEE Congresses on Evolutionary Computation Repository, 2015.

[60] Ralf Salomon. Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. a survey of some theoretical and practical aspects of genetic algorithms. *Biosystems*, 39(3):263–278, 1996.

[61] *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2010, Barcelona, Spain, 18-23 July 2010*. IEEE, 2010.

[62] Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Swagatam Das. Dynamic multi-swarm particle swarm optimizer with sub-regional harmony search. In *IEEE Congress on Evolutionary Computation*. IEEE, 2010.

[63] Daniel Molina, Manuel Lozano, and Francisco Herrera. Ma-sw-chains: Memetic algorithm based on local search chains for large scale continuous global optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2010, Barcelona, Spain, 18-23 July 2010*. IEEE, 2010.

[64] Antonio LaTorre, Santiago Muelas, and José-María Peña. Large scale global optimization: Experimental results with mos-based hybrid algorithms. In *2013 IEEE Congress on Evolutionary Computation*, pages 2742–2749. IEEE, 2013.

[65] *IEEE Congress on Evolutionary Computation, CEC 2015, Sendai, Japan, May 25-28, 2015*. IEEE, 2015.

[66] Fei Wei, Yuping Wang, and Yuanliang Huo. Smoothing and auxiliary functions based cooperative coevolution for global optimization. In *2013 IEEE Congress on Evolutionary Computation*, pages 2736–2741. IEEE, 2013.

[67] Jinpeng Liu and Ke Tang. Scaling up covariance matrix adaptation evolution strategy using cooperative coevolution. In Hujun Yin, Ke Tang, Yang Gao, Frank Klawonn, Minho Lee, Thomas Weise, Bin Li, and Xin Yao, editors, *Intelligent Data Engineering and Automated Learning – IDEAL 2013*, pages 350–357, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[68] Dongyang Li, Weian Guo, Lei Wang, and Qidi Wu. A modified APSODEE for large scale optimization. In *IEEE Congress on Evolutionary Computation, CEC 2021, Kraków, Poland, June 28 - July 1, 2021*, pages 1976–1982. IEEE, 2021.

[69] Dongyang Li, Weian Guo, Lei Wang, and Qidi Wu. A modified apsodee for large scale optimization. In *2021 IEEE Congress on Evolutionary Computation, CEC'2021*, pages 1976–1982. IEEE, 2021.

[70] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Technical Report RR-6829, INRIA, 2010.

[71] Ralf Salomon. Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. a survey of some theoretical and practical aspects of genetic algorithms. *Biosystems*, 39(3):263–278, 1996.