UNIVERSIDAD VERACRUZANA

Centro de Investigación en Inteligencia Artificial

Un estudio sobre diversidad en optimización evolutiva con restricciones

TESIS

Una tesis presentada en cumplimiento de los requisitos
para obtener el grado de Maestro en Inteligencia Artificial

QUE PRESENTA:

LUIS ENRIQUE CONTRERAS VARELA

DIRECTOR DE TESIS: DR. EFRÉN MEZURA MONTES

ASESORA: DRA. MARGARITA REYES SIERRA

Xalapa, Ver. **31/Enero/2018**

Índice general

Capítulo 1

Introducción

1.1. Antecedentes

Las dificultades que presentan los métodos clásicos para la búsqueda y optimización al enfrentar problemas complejos [1] han ocasionado la necesidad de encontrar vías alternas para atacar dichos problemas. En ese sentido los algoritmos bio-inspirados aparecen como una opción interesante, al ser aplicables en una amplia gama de problemas y otorgar buenos resultados en un tiempo aceptable [2].

A pesar de que los algoritmos bio-inspirados albergan diversas metaheurísticas, como los algoritmos evolutivos o los algoritmos de inteligencia colectiva, la idea central se mantiene: Escoger una representación apropiada del problema, evaluar la calidad de las soluciones potenciales y definir operadores para producir variaciones en el conjunto de soluciones [3]. En cada generación del algoritmo se espera que la población mejore su valor de aptitud y converja hacia la solución óptima; desde las primeras iteraciones (en el caso de maximización) la población abandona las zonas de baja aptitud y se acerca a las regiones más altas. Hacia el final de la ejecución, la población se concentra en pocas colinas del paisaje de aptitud [2].

Al fenómeno anterior se le conoce como convergencia, y una vez que ocurre, los operadores de variación pierden la capacidad de generar nuevos individuos haciendo que todos los descendientes se encuentren en un subconjunto muy pequeño del espacio de búsqueda. Si la convergencia se presenta antes de haber encontrado el óptimo global de la función se cataloga como convergencia prematura [4].

La convergencia prematura es uno de los principales problemas de los algoritmos evolutivos [5] y su principal origen reside en la falta de diversidad en la población, pues una vez que la población se ha tornado redundante (con niveles bajos de diversidad) el algoritmo toma una única trayectoria hacia el mejor individuo sin que éste sea necesariamente el óptimo [6].

La diversidad puede ser vista como un indicador de la disimilitud entre los individuos [6], una medida que aproxima el número de diferentes soluciones en la población en determinado momento [2]. Tener una alta diversidad inicial es muy importante, pues sin ésta es posible que el algoritmo sea incapaz de alcanzar el óptimo [5].

La literatura especializada ahonda en la repercusión de la diversidad en distintos escenarios. Además de la relación entre la diversidad y la convergencia prematura, promover la diversidad en la población a lo largo del proceso de optimización es provechoso para lidiar con funciones multimodales, pues con una población diversa es posible explorar múltiples colinas del paisaje de aptitud simultáneamente. Por otro lado, la dispersión de las soluciones permite que el algoritmo se adapte más rápidamente a los cambios en el paisaje de aptitud en problemas dinámicos [6]. Así mismo al optimizar problemas multi-objetivo se busca que los frentes de Pareto tengan un compromiso entre diversidad y proximidad con el frente óptimo [7].

Debido a lo anterior, el área ha estudiado la diversidad desde distintos enfoques:

- *Promoción de la diversidad.* La literatura es muy abundante en cuanto a mecanismos de preservación de la diversidad. Algunos ejemplos de ello son estrategias como el *nitching*, que permite la búsqueda en diversos picos en paralelo mediante la formación de especies, soluciones con aptitud similar [8]; el *crowding*, donde se incluyen restricciones al reemplazo de soluciones para que cada nuevo elemento sustituya al más similar de la población. Así como la *selección restringida de padres*, la cual limita la selección de acuerdo a criterios como la distancia entre soluciones o la prevención del incesto; *inyección* de números aleatorios en un punto, o de nuevas soluciones a la población, etc [9].

- *Guiar la búsqueda.* Aplicar medidas de diversidad para efectuar un cambio de estrategia en la búsqueda. Bajo esta óptica el trabajo de Ursem [10] utiliza la distancia al punto promedio para cambiar de una estrategia con énfasis en la explotación, cuando la diversidad sobrepasa cierto umbral, a una explorativa mientras el nivel de diversidad sea inferior al umbral.

  Así mismo, Yang et al. [11] calculan la media y la desviación estándar de la población para cada dimensión, si ésta permanece igual por un número determinado de generaciones se realiza un proceso de diversificación para la dimensión que ha convergido o presenta estancamiento.

- *Control de parámetros.* Modificar los valores de ciertos parámetros en función de la diversidad en tiempo de ejecución. Por ejemplo, Zaharie [12] analiza la mutación y recombinación de la evolución diferencial de manera

analítica para determinar la varianza esperada de la población en la siguiente generación. A partir de ella y de la varianza actual se estima la tasa de convergencia, la cual es comparada con el parámetro tasa de convergencia deseada para ajustar los parámetros de evolución diferencial. De tal suerte que la tasa de convergencia se mantenga constante a lo largo del proceso de evolución.

Es evidente que la diversidad responde a la pregunta ¿Qué tan diferentes son los individuos de la población? pero esta diferencia entre soluciones puede expresarse a distinto nivel granular. De tal suerte que existen medidas de diversidad para los dos espacios en los que trabajan los algoritmos bio-inspirados, el espacio genotípico y el fenotípico [13].

En el primer caso, las medidas de diversidad genotípicas (GDM por sus siglas en inglés), se describe la variación del material genético de los individuos dependiendo de la representación seleccionada. Las GDMs están relacionadas con la ubicación de las soluciones en el espacio de búsqueda. Para la codificación binaria se tienen medidas basadas en la frecuencia de los alelos, distancias de Hamming y entropía, mientras que para la codificación real las medidas parten de la dispersión estadística, distancias euclidianas, etc [13].

Después se encuentran las PDMs o medidas de diversidad fenotípicas, también conocidas como diversidad de aptitud [14]. Éstas trabajan con los valores de aptitud de las soluciones, y la mayoría está formada por combinaciones entre los valores de aptitud del mejor individuo, el peor y de la solución promedio [13].

Con la anterior taxonomía es posible identificar el tipo de información que brinda cada clase de medida; las medidas genotípicas (GDMs) monitorean la diversidad en torno a la exploración puesto que trabajan en el espacio de búsqueda. En el caso de las medidas fenotípicas (PDMs) se ofrece información acerca de la explotación al emplear únicamente los valores de aptitud. Si las medidas de diversidad están normalizadas, valores unitarios significan exploración máxima (comúnmente la diversidad inicial de la población) y valores cercanos a cero expresan explotación máxima (cuando el algoritmo ha convergido)[14].

Un punto importante al considerar qué tipo de medida de diversidad debe aplicarse es la complejidad de cómputo. Las GDMs requieren de cálculos de distancias entre pares de vectores (para la codificación real), lo cual puede ser muy pesado si se trabaja con problemas de gran escala (muchas dimensiones). En cambio, las PDMs trabajan únicamente con el valor de aptitud de cada individuo, relajando el cómputo considerablemente [15].

En párrafos anteriores se mencionaron algunos estudios relacionados con la diversidad, ya sea propiciándola en la población usando estrategias como el nitching, crowding, etc. o bien, aplicando medidas de diversidad para guiar la búsqueda del algoritmo o para el control de parámetros. Estos trabajos emplean algún método para medir diversidad, pero para asegurarse que se está incre-

mentando la diversidad en la población aplicar métodos como el nitching no es suficiente, debido a que se requiere conocer el paisaje de aptitud a priori. De modo que las medidas de diversidad son preferibles por ser métodos directos de sensar la diversidad [16]. A pesar de ello la habilidad de las distintas medidas reportadas en la literatura especializada para describir fielmente la diversidad no ha sido investigada exhaustivamente [16].

Para paliar esta situación, en los últimos años se han realizado nuevos estudios que comparan el desempeño de las diferentes medidas de diversidad en diversos escenarios. Por ejemplo, en [17] Olorunda y Engelbrecht analizaron 6 diferentes GDMs aplicadas al algoritmo de cúmulo de partículas (PSO por sus siglas en inglés), en su trabajo encontraron que las medidas que responden mejor son la distancia promedio alrededor del centro del cúmulo y la distancia alrededor de todas las partículas. Otro estudio muy interesante es el de Corriveau et al. [14], ellos hicieron una recopilación de las GDMs propuestas en el área y las emplearon en dos experimentos: 1) un comparativo con algoritmos del estado del arte resolviendo el benchmark CEC 2005 [18], 2) la evaluación del desempeño de las medidas en un simulador de convergencia de su autoría.

Si bien los trabajos referidos anteriormente han encontrado patrones y tendencias respecto al desempeño de las medidas de diversidad, es reconocido que el determinar que una de ellas sea más apropiada que el resto al emplearlas en cualquier tipo de búsqueda sigue siendo un tema abierto [14]. No existe un consenso en torno a las medidas de diversidad, a las condiciones de aplicación, ni a la relación entre un buen manejo de diversidad y buenos resultados finales. Además, sería conveniente realizar estudios sobre el desempeño de las medidas de diversidad en otros tipos de problemas; puesto que existen muy pocos trabajos comparativos y la mayoría de éstos se dirige a la optimización global sin restricciones.

Hasta nuestro conocimiento, no existen medidas de diversidad ni estudios del desempeño de las medidas existentes enfocados específicamente en problemas de optimización con restricciones. Por este motivo, sería deseable evaluar el comportamiento de las medidas de diversidad existentes en la literatura en algoritmos de distinta naturaleza al resolver problemas restringidos, en busca de obtener un acercamiento a la diversidad en optimización con restricciones.

1.2. Definición del Proyecto

Un tipo de problema en el que no se ha profundizado el estudio de diversidad es la optimización con restricciones. En estos problemas la diversidad es importante, pues como menciona Deb, es conveniente mantener la pluralidad de las soluciones y así permitir a los operadores de cruza encontrar constantemente mejores soluciones factibles [19].

Este proyecto propone un estudio de diversidad intentando incluir todos los elementos involucrados con la dinámica de la diversidad de la población y su vínculo con los resultados finales. Específicamente el trabajo presenta dos comparativos empíricos. El primero, centrado en medidas de diversidad empleando algoritmos de optimización del estado del arte. El segundo experimento contempla diversas combinaciones de algoritmos básicos, técnicas de manejo de restricciones y mecanismos para la promoción de la diversidad; pues todas estas variables juegan un papel importante en la dinámica de la diversidad de la población.

1.3. Hipótesis

1. Existe una relación entre resultados finales competitivos (convergencia a la vecindad del óptimo) de los algoritmos bio-inspirados del estado del arte para optimización con restricciones y los resultados de las medidas de diversidad que reflejen precisamente convergencia.
2. Las medidas de diversidad reflejarán el efecto de los mecanismos de promoción de diversidad en algoritmos bio-inspirados para optimización con restricciones.

1.4. Objetivos

1.4.1. Objetivo General

Realizar un comparativo empírico de diversidad en optimización con restricciones, utilizando algoritmos del estado del arte y combinaciones de algoritmos básicos, manejadores de restricciones y técnicas de promoción de la diversidad; para identificar las relaciones entre buenos resultados finales y un buen manejo de diversidad.

1.4.2. Objetivos Específicos

- Seleccionar e implementar algoritmos de optimización con restricciones del estado del arte y medidas de diversidad.

- Comparar el desempeño de las medidas de diversidad en los algoritmos del estado del arte seleccionados.
- Diseñar las combinaciones de algoritmos básicos, manejadores de restricciones y técnicas de manejo de la diversidad a ser probadas.
- Contrastar la dinámica en diversidad y los resultados de todas las combinaciones para establecer patrones entre cada variable.

1.5. Publicaciones

Cabe destacar que a partir de esta investigación se desarrollaron tres manuscritos, de los cuales dos ya fueron publicados y el tercero está próximo a ser sometido:

1. *Luis-Enrique Contreras-Varela y Efrén Mezura-Montes, Un Simulador para Evaluar Diversidad en Optimización Evolutiva con Restricciones, VI Foro Nacional de Divulgación Científica y Tecnológica, Xalapa, Veracruz, México, Noviembre 2016.*
2. *Luis-Enrique Contreras-Varela y Efrén Mezura-Montes, Comparativo empírico de medidas de Diversidad en problemas de optimización evolutiva con restricciones, Congreso Mexicano de Inteligencia Artificial, Toluca, México, Mayo 2017 (en impresión).*
3. *Luis-Enrique Contreras-Varela and Efrén Mezura-Montes, A Diversity Promotion Study in Constrained Optimization, IEEE Congress on Evolutionary Computation, IEEE Press, Rio de Janeiro, Brazil, July 2018 (accepted).*

1.6. Organización del documento

El documento se divide como se describe a continuación:

- El Capítulo 2 presenta una breve reseña de los conceptos más elementales de optimización y una descripción de los métodos clásicos para problemas de optimización. Se hace énfasis en las ventajas y desventajas de utilizar dichos mecanismos con el objetivo de poner en contexto al lector respecto a la dificultad del problema en cuestión: la optimización con restricciones.
- El Capítulo 3 describe las bases del cómputo evolutivo, junto con ciertas técnicas relevantes al estudio. En él se incluyen los componentes de un algoritmo evolutivo, los principales paradigmas, un apartado dedicado a manejadores de restricciones y algunos algoritmos destacados que serán referenciados en los experimentos.
- El Capítulo 4 muestra una revisión de la literatura en torno a la diversidad en el cómputo evolutivo. Se abordan temas centrales en el área; comenzando por la definición de diversidad, el balance entre exploración y explotación, mecanismos para la promoción de la diversidad, medidas de diversidad, estudios comparativos, etc.
- El Capítulo 5 reporta el primer experimento llevado a cabo, en el cual se analiza el desempeño de 6 medidas de diversidad en 2 algoritmos del estado del arte para optimización con restricciones. Se contrastan los resultados estadísticos de los algoritmos con las gráficas de diversidad de cada medida.
- El Capítulo 6 presenta el segundo experimento de este trabajo. En dicho experimento se trabajó con algoritmos básicos aplicando diferentes manejadores de restricciones y técnicas de promoción de la diversidad. Se evaluaron 48 variantes de algoritmos, generadas de las combinaciones entre *algoritmo* + *manejador* + *técnica de promoción*, para seleccionar el subconjunto que consiguió los mejores resultados. La discusión versa sobre la influencia de cada elemento evaluado en la diversidad.
- Finalmente, el Capítulo 7 resume los hallazgos y delinea el trabajo futuro.

Capítulo 2

Optimización Clásica

La palabra optimización viene de la raíz latina *optimus*, la cual hace referencia al ideal último. De este modo se puede entender la optimización como la acción de encontrar aquello que se considere el ideal último en el contexto en turno, lo cual llevado al terreno de las matemáticas corresponde al mayor o menor valor de la función a optimizar [20].

La optimización es una tarea común para las personas, es evidente la presencia de dichos procesos en distintos aspectos de la vida: se sabe que las industrias se esmeran en reducir sus costos de producción y maximizar sus ganancias, los deportistas de alto rendimiento organizan sus esfuerzos, modifican su alimentación y entrenamiento para obtener los mejores resultados posibles, etc.

Debido a su importancia, la optimización ha sido estudiada desde los inicios de las matemáticas hasta la actualidad, tomando particular fuerza durante la segunda guerra mundial. En aquel tiempo los gobiernos comenzaron a contratar científicos de múltiples disciplinas para optimizar problemas complejos de corte militar, resultando en parte importante del desarrollo de avances en el área [20].

A estas técnicas se les conoce como programación matemática y a pesar de haber sido empleadas exitosamente, presentan el inconveniente de ser aplicables sólo en problemas que cumplan ciertos requisitos. Por lo anterior, en los últimos años nuevas estrategias heurísticas se han introducido para trabajar con problemas que no se ajustan a las condiciones que requieren las técnicas de programación matemática. Además de dichas técnicas, existen múltiples algoritmos heurísticos y es tarea del diseñador conocer las bases del problema con el que se enfrenta y las distintas alternativas y así seleccionar la técnica más apropiada [21], la cual puede ser un método clásico o una técnica heurística dependiendo del problema.

Este capítulo introduce los fundamentos de la optimización y presenta una síntesis breve de algunos métodos de programación matemática. Conceptos necesarios para abordar los capítulos posteriores.

2.1. Definición de optimización

En general la optimización se define como:

$$\text{encontrar } \vec{x} \text{ que optimice } f(\vec{x})$$

$$\text{sujeto a:}$$

$$\begin{aligned} g_j(\vec{x}) &\leq 0, & j &= 1, 2, \cdots, J; \\ h_k(\vec{x}) &= 0, & k &= 1, 2, \cdots, K; \\ \vec{x}_i^{(L)} &\leq \vec{x}_i \leq \vec{x}_i^{(U)}, & i &= 1, 2, \cdots, N \end{aligned}$$

Donde $\vec{x}$ es el vector de las soluciones $\vec{x} = (x_1, x_2, ..., x_N)^T$, J el número de restricciones de desigualdad y K el de restricciones de igualdad [21]. Los límites $\vec{x}_i^{(L)}$ y $\vec{x}_i^{(U)}$ definen el espacio de búsqueda S, se conoce como zona factible $F \subset S$ al subconjunto del espacio que satisface todas las restricciones.

De la definición anterior se pueden observar los elementos que forman parte de un problema de optimización:

- **Variables de diseño ($\vec{x}$):** Son el conjunto de variables que deben ser modificadas para conseguir el mejor valor posible de la función objetivo. La combinación adecuada de valores de dichas variables resultará en el objetivo final de cualquier proceso de optimización. Al diseñar las variables para un problema nuevo es importante considerar que la eficiencia de los algoritmos se ve mermado conforme se incrementa el número de variables, por lo que se recomienda seleccionar el menor número de variables posible [21].

- **Función objetivo ($f(\vec{x})$):** Es una abstracción matemática (f((x))) del objetivo a cumplir con la optimización, evalúa de manera cuantitativa la calidad de las soluciones. Funciones objetivo comunes pueden estar orientadas a minimizar los costos de producción, reducir el peso de algún componente, etc.

  Existen objetivos difíciles de cuantificar, como puede ser calificar la calidad de una pintura o de un poema. Así mismo existen objetivos difíciles de evaluar, tal es el caso de la optimización robótica que requiere de simular el comportamiento del robot en cada solución evaluada por el algoritmo usado. Los anteriores puntos deben ser considerados por el diseñador del problema de optimización [21].

  En muchas situaciones es probable que exista más de un objetivo a optimizar simultáneamente. Cuando esto sucede es necesario aplicar una técnica especializada para múltiples objetivos, o en su defecto escoger el objetivo más importante y agregar el resto como restricciones [21].

- **Límite de variables ($\vec{x}_i^{(L)}$ y $\vec{x}_i^{(U)}$):** Establecer un límite para las variables es necesario para aplicar la mayoría de técnicas. Los límites de cada variable de diseño permiten a los algoritmos buscar el óptimo dentro de un espacio de búsqueda definido S [21].

- **Restricciones ($g_i(\vec{x})$ y $h_i(\vec{x})$):** Las restricciones representan relaciones funcionales del problema para satisfacer e incluir en el modelo ciertos fenómenos físicos o limitaciones de aplicación.

  Los tipos de restricciones más usuales son de desigualdad e igualdad. Las restricciones de desigualdad solicitan que al evaluar la función de la restricción la solución presente un valor menor o igual que el que define la restricción (usualmente cero). En el caso de las restricciones de igualdad, éstas exigen que la evaluación resulte en un valor establecido (la convención es definir la restricción igual a cero).

  Debido a que las restricciones de igualdad son las más difíciles de satisfacer es muy común relajar dichas restricciones tratándolas como dos restricciones de desigualdad equivalentes a la restricción original con una pequeña tolerancia [21].

2.2. Condiciones de optimalidad

El objetivo al aplicar alguna técnica para optimización es encontrar el punto óptimo de la función evaluada. Como es evidente, la mejor solución deberá ubicarse en el punto más bajo posible del paisaje de la función objetivo, en el caso de minimización; o en su defecto, poseer el mayor valor factible de la función objetivo, resultando en maximización.

Existen tres diferentes tipos de puntos óptimos [21, 20]:

1. **Óptimo global:** Un punto $\vec{x}^{**}$ es un óptimo global si no existe ningún otro punto con un valor mejor que $\vec{x}^{**}$. El objetivo de un proceso de optimización es encontrar el óptimo global.

$$f(\vec{x}^{**}) \leq f(\vec{x}), \vec{x} \in S$$

2. **Óptimo local:** Se dice que una solución x^* es un óptimo local si no existe ningún punto en el vecindario cercano mejor que el valor de $\vec{x}^*$. Dicha cercanía se define a partir del espacio euclidiano con centro en x^* y radio ϵ; es decir, el subconjunto de vectores en el espacio de búsqueda cuya distancia con respecto a $\vec{x}^*$ sea menor a ϵ. Evidentemente puede existir más de un óptimo local.

$$\text{Sea } B_\epsilon(\vec{x}^*) := \{\vec{y} \in \mathbb{R}^n | \ ||\vec{y} - \vec{x}^*|| < \epsilon\}$$

el espacio euclidiano con radio ϵ con centro en $\vec{x}^*$

$$\exists \epsilon > 0 \text{ tal que } f(\vec{x}^*) \leq f(\vec{x}), \vec{x} \in S \cap B_\epsilon(\vec{x}^*)$$

3. **Punto de inflexión:** Un punto de inflexión $\vec{x}^*$ es aquél donde el valor de aptitud incrementa y decrementa conforme $\vec{x}^*$ se mueve: si $\vec{x}^*$ incrementa, el valor de la función objetivo incrementa; si $\vec{x}^*$ decrementa, el valor de la función objetivo decrementa. Lo anterior tamién puede darse en el sentido inverso siempre que se respeten ambas condiciones.

2.3. Clasificación de problemas de optimización

De acuerdo a las características del problema en cuestión, éste se puede clasificar dentro de varias categorías, en la siguiente Figura se aprecia la clasificación de Yang[22].

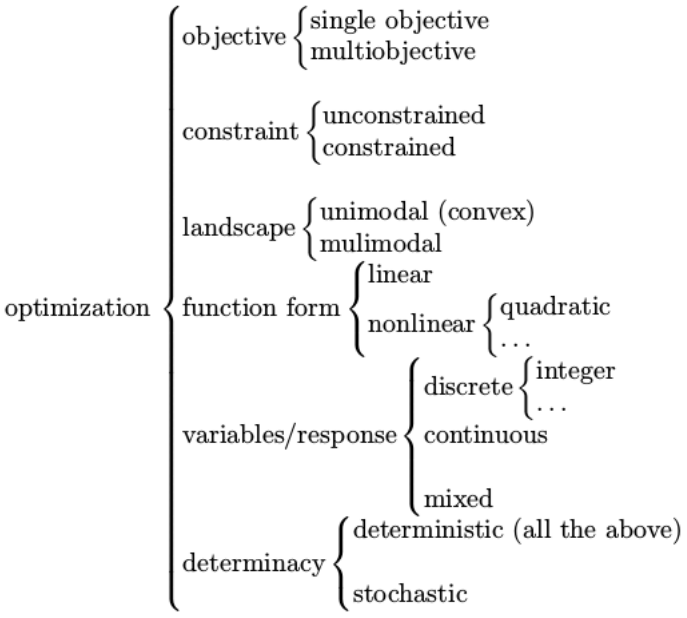

Figura 2.1: Clasificación de problemas de optimización.

Como puede apreciarse en el esquema de la Figura 2.1, un problema de optimización se puede clasificar por sus objetivos, restricciones, tipo de funciones, tipo de variables, etc. Debido a las diferentes características de los problemas, éstos requieren de algoritmos especializados para las condiciones de cada problema.

En este trabajo la clasificación de problemas por restricciones es de particular interés, ya que se investigan distintos aspectos de la diversidad de la población

usando algoritmos evolutivos al resolver problemas de optimización con restricciones.

En la siguiente sección se abordan algunas técnicas para problemas de optimización con una sola variable, puesto que son la base de muchas otras técnicas más sofisticadas para problemas de otras categorías [21].

2.4. Algoritmos de optimización de una sola variable

Los problemas de optimización de una sola variable son los más sencillos dentro de la clasificación de los problemas. Estos problemas se encargan de resolver el problema:

$$minimizar f(\vec{x})$$

Donde $f(\vec{x})$ es la función objetivo y x es una variable real. El propósito de los algoritmos es encontrar una instancia de x con el valor mínimo de la función $f(\vec{x})$ [21].

En la optimización de problemas una sola variable se suele trabajar en dos etapas: el primer paso, consiste en encontrar el intervalo donde yace el óptimo, proceso conocido como *bracketing*; posteriormente se aplica un algoritmo para encontrar el mínimo dentro del intervalo obtenido después de aplicar bracketing.

Algunas técnicas de bracketing son [21]:

- *Búsqueda exhaustiva.* Divide el espacio de búsqueda proporcionado en n puntos intermedios, y evalúa cada tres puntos sucesivos hasta satisfacer la condición $f(x_i) \geq f(x_{i+1}) \leq f(x_{i+2})$. En caso de no satisfacer la condición después de evaluar todos los puntos se concluye que no existe el mínimo en los límites de las variables (x^L y x^U) o que el mínimo se encuentra en alguno de los límites.

- *Método de fase de limitación.* Comienza con un punto arbitrario x_0 para encontrar una dirección de búsqueda dependiendo de los valores de la función de aptitud de los puntos $x_o \pm \Delta$. Si los valores en la función objetivo de los puntos cumplen $f(x_0 - |\Delta|) \geq f(x_0) \geq f(x_0 + |\Delta|)$, la dirección es positiva. En el caso inverso la dirección es negativa ($f(x_0 - |\Delta|) \leq f(x_0) \leq f(x_0 + |\Delta|)$). Si ninguno de estos requisitos se cumple se debe seleccionar otro punto de partida y buscar la dirección de búsqueda nuevamente.

  Una vez seleccionada la dirección de búsqueda, el punto de partida es referido como x_k, siendo $k = 0$. En seguida se calcula el punto $x_{k+1} = x_k 2^k \Delta$, donde Δ será positivo o negativo de acuerdo a la dirección de búsqueda encontrada en los primeros pasos. Si el valor de la función objetivo del punto $f(x_{k+1})$ es menor que el valor de la solución anterior $f(x_k)$, significa

que el valor de la función de aptitud no ha dejado de decrementarse y que puede seguir mejorando; por lo tanto se incrementa en uno el valor de k y se calcula un nuevo punto x_{k+1}.

Cuando el valor de la función objetivo del nuevo punto x_{k+1} es mayor que la solución x_k se termina el procedimiento y se concluye que el mínimo se encuentra en el intervalo (x_{k-1}, x_{k+1}).

En este algoritmo la selección de un Δ grande acelera la búsqueda perdiendo precisión como consecuencia. En su defecto, valores pequeños de Δ son más precisos, pues retornan intervalos más pequeños a costa de un mayor número de evaluaciones de la función objetivo.

Una vez concluido el proceso de bracketing, se tiene el intervalo donde yace el mínimo y no propiamente el valor del óptimo global x^*. Para encontrar el punto mínimo de la función es necesario aplicar técnicas más sofisticadas [21]. Existen diferentes tipos de algoritmos para esta tarea, en adelante se describen tres clases de dichas técnicas y algunos ejemplos de cada tipo.

2.4.1. Métodos de eliminación de regiones

Los algoritmos de eliminación de regiones trabajan con funciones unimodales (al menos unimodales dentro del intervalo proporcionado). Al considerar dos puntos x_1 y x_2 dentro del intervalo resultante del proceso de bracketing (a, b) donde $x_1 < x_2$ es posible emplear las siguientes reglas:

- Si $f(x_1) > f(x_2)$ entonces el mínimo no se encuentra en (a, x_1).
- Si $f(x_1) < f(x_2)$ entonces el mínimo no se encuentra en (x_2, b).
- Si $f(x_1) = f(x_2)$ entonces el mínimo no se encuentra en (a, x_1) ni en (x_2, b).

La aplicación de las reglas produce que el espacio de búsqueda se reduzca y progresivamente se encuentre un intervalo lo suficientemente pequeño en donde se encuentra el mínimo de la función objetivo.

Por ejemplo, se puede dividir el espacio de búsqueda en cuatro regiones y ejecutar las reglas de eliminación de regiones entre los puntos intermedios para reducir los límites a la mitad. Proceso que se realiza hasta que la longitud del intervalo sea lo suficientemente pequeño, como es el caso en el *método de división de intervalos a la mitad*.

Otra opción es realizar una *búsqueda de Fibonacci*, donde los números de la sucesión son empleados para determinar los puntos intermedios. Puesto que para cada número de la serie de Fibonacci se calcula a partir de los dos anteriores, este algoritmo sólo se requiere de una evaluación por iteración.

Una mejora al método anterior se observa en la *búsqueda de la sección áurea*. Esta técnica normaliza el problema de optimización de modo que el nuevo espacio de búsqueda sea $(0, 1)$ y obtiene los puntos intermedios usando la proporción áurea. Este algoritmo es muy similar al de búsqueda de Fibonacci, pues en ambos algoritmos solo se requiere una evaluación de la función objetivo por iteración; a pesar de ello la búsqueda de la sección áurea es superior, ya que no es necesario calcular los números de la sucesión de Fibonacci y la proporción de la sección eliminada es la misma en cada iteración.

2.4.2. Métodos de estimación de puntos

En este tipo de algoritmos no sólo se emplean dos puntos para guiar la búsqueda, también utilizan la magnitud y el signo del valor de la función objetivo para los puntos seleccionados. Dichos valores se calculan para encontrar una función unimodal que se ajuste a los puntos seleccionados, de tal suerte que el mínimo de la nueva función sea el punto de partida una nueva iteración y progresivamente se encuentre el óptimo de la función objetivo original.

Un ejemplo de esta clase de algoritmos es el *método de estimación cuadrática sucesiva*, donde usando un punto inicial y dos soluciones a una distancia Δ del primer punto, se calcula el mínimo $f(\bar{x})$ de una función cuadrática general generada con los tres puntos antes mencionados. Si el valor mínimo de la función objetivo original de los tres puntos y la posición de dicho punto es lo suficientemente cercana al mínimo de la función aproximada y su ubicación, el mínimo de la función aproximada se toma como óptimo $f(\bar{x})$ para la función objetivo original.

2.4.3. Métodos basados en gradiente

Los métodos anteriormente descritos trabajan con los valores de la función objetivo y no con derivaciones de la misma. La propiedad de optimalidad donde en cada óptimo local o global el gradiente es igual a cero puede usarse para determinar el momento de paro de la búsqueda [21].

Desafortunadamente en problemas del mundo real es complicado obtener información de las derivadas, ya sea por la naturaleza de la función objetivo o la complejidad de cómputo para calcular las derivadas. A pesar de ello, las técnicas basadas en gradiente son aplicadas efectivamente en diversos problemas, aunque es recomendable aplicar estos métodos únicamente cuando las derivadas pueden calcularse fácilmente.

Un ejemplo es el *método de Newton-Raphson*. Éste aplica la primera y segunda derivada desde un punto inicial arbitrario x_1; con los valores de $f'(x)$ y $f''(x)$ se obtiene un nuevo punto $x_{k+1} = x_k - f'(k)/f''(k)$, si el valor absoluto de la primera derivada $|f'(x_{k+1})|$ del nuevo punto es lo suficientemente pequeño (verificado a

través del parámetro ϵ) el algoritmo se detiene, en caso contrario se repite el proceso para x_{k+2} y así sucesivamente.

Por otro lado el *método de bisección* no requiere calcular la segunda derivada, proceso que requiere de tres evaluaciones. En su defecto solo se emplea la primera derivada de dos puntos para eliminar una porción del espacio de búsqueda de manera similar a los métodos de eliminación de regiones mencionados anteriormente. Una vez calculadas las derivadas del intervalo que rodea al mínimo (obtenido a través de un algoritmo de bracketing), se dice que el mínimo se encuentra en el intervalo (a, b) si la primera derivada $f'(a) < 0$ y $f'(b) > 0$. Si la condición anterior se cumple el siguiente paso resulta en calcular el punto medio z del intervalo (a, b) y calcular la primera derivada $f'(z)$. Si $|f'(z)| < \epsilon$, donde epsilon es un número muy pequeño que representa la precisión, el algoritmo regresa z como mínimo. En caso contrario, si $f'(z)$ es negativo el nuevo intervalo será (z, b) y si $f'(z) > 0$ los nuevos limites para la siguiente iteración se ajustan a (a, z) y el proceso se repite.

Finalmente, el *método secante* es virtualmente la misma técnica que el método de bisección, excepto por el cálculo del punto intermedio. En el método secante se asume que la primera derivada de la función objetivo varía linealmente entre los extremos de la función, y ya que los límites que envuelven al mínimo de la función objetivo tienen signos opuestos, existe un punto entre a y b cuya primera derivada es 0. Cuando el punto z tenga un valor de la primera derivada $f'(z)$ lo suficientemente cercano a 0 (usando el parámetro ϵ) el algoritmo se detendrá, en caso contrario los límites serán actualizados y se realizará una nueva iteración.

2.5. Algoritmos de optimización multivariable

En el apartado anterior se revisaron los algoritmos de optimización para problemas con una sola variable, como se puede observar en la clasificación de la Figura 2.1 existen muchos tipos de problemas y entre ellos se encuentra un numeroso grupo que presenta más de una variable.

Los problemas de optimización multivariable, como su nombre lo indica, trabajan con un vector de variables $\vec{x}_{(t)} = [x_1, x_2, \cdots, x_N]$ en vez de una sola para encontrar el óptimo de la función.

Para resolver estos problemas se requiere de técnicas diferentes a las mostradas en la sección anterior, aunque en ocasiones se aplican algoritmos de optimización univariable sucesivamente para encontrar el mínimo de una dirección de búsqueda en particular; como es el caso de la *búsqueda unidireccional*.

En general los algoritmos de optimización multivariable pueden dividirse en dos categorías principales: *métodos directos* y *métodos basados en gradiente*. En seguida se abordan ambos tipos de algoritmos mencionando algunos ejemplos.

2.5.1. Métodos directos

Se les conoce como métodos directos a los algoritmos que no necesitan obtener derivaciones de la función objetivo y únicamente trabajan con los valores de la función. Es importante conocer estos métodos, ya que a pesar de que los métodos basados en gradiente son más efectivos estos dependen de derivaciones, las cuales pueden no estar disponibles o ser difíciles de calcular debido al costo computacional de la función objetivo. En tales casos los métodos directos son más apropiados [21].

Algunos ejemplos de métodos directos son:

- *Método de optimización evolutiva.* Este algoritmo trabaja con 2^N+1 puntos simultáneamente, donde 2^N puntos corresponden a los vértices de un hipercubo de dimensión N y el otro punto es el centro del hipercubo, nombrado x_0. En cada iteración se evalúan los valores de la función objetivo para cada punto y se ubica al mejor, en la siguiente iteración si el mejor punto sigue siendo el mismo, el hipercubo se reducirá de acuerdo al parámetro factor de reducción Δ_i, en caso contrario se formará un nuevo hipercubo alrededor de la nueva mejor solución. El algoritmo se detiene cuando el hipercubo es muy pequeño.

  El desempeño de este algoritmo depende en gran medida de la posición inicial del hipercubo y de su tamaño, así como del factor de decremento.

- *Método de búsqueda simplex.* En la búsqueda simplex se utilizan $N + 1$ puntos a los que se les llama simplex. Es necesario que estos puntos no se encuentren en un hipercubo de volumen cero, es decir; si el problema tiene dos variables los puntos del simplex no pueden ubicarse en una línea, en caso de tener tres variables los puntos no pueden situarse en un plano. La misma situación aplica para problemas de mayor dimensión.

  El primer paso del algoritmo es encontrar el peor punto x_h en el simplex y el centroide x_c del resto de puntos del simplex. Al peor punto x_h se aplicará la operación de reflexión (opuesto simétrico, con respecto al centroide x_c, del peor punto del simplex), esperando llevar al peor punto a una zona con mejores valores de la función objetivo.

  Dependiendo del nuevo valor del punto se ejecutará una de tres operaciones: expansión (movimiento hacia la dirección opuesta a x_h de acuerdo al parámetro γ) si el nuevo punto supera al mejor punto del simplex, contracción negativa (movimiento hacia la dirección de x_h de acuerdo a β) si la nueva solución es inferior que el peor punto del simplex o contracción positiva (movimiento hacia la dirección opuesta a x_h de acuerdo a β) si es superior a la peor solución del simplex e inferior al segundo mejor punto del simplex.

Finalmente se elimina el peor punto x_h al ser remplazado por la nueva solución.

Estos tres pasos conforman una iteración del método simplex. Cuando la distancia de los puntos del simplex al centroide sea lo suficientemente pequeña el algoritmo se detiene.

- *Método de búsqueda de patrones Hooke-Jeeves.* La idea central es crear un conjunto de direcciones de búsqueda iterativamente al combinar movimientos exploratorios y movimientos heurísticos por patrón.

  El movimiento exploratorio consiste en perturbar iterativamente las variables positiva y negativamente, a partir de la solución actual $\vec{x}$ se calcula el valor de la función objetivo de dicha solución alterando la variable en turno $f^- = f(x_i - \Delta)$ y $f^+ = f(x_i + \Delta)$ usando el parámetro Δ en cada paso. Del conjunto de puntos se selecciona el mejor y se repite el proceso de perturbación y selección del mejor hasta haber perturbado todas las variables. Si se obtuvo un nuevo punto después de aplicar el movimiento exploratorio éste se considera exitoso.

  El movimiento heurístico por patrón se aplica cuando el movimiento exploratorio fue exitoso, y lleva al punto actual $x^k + 1$ a la ubicación del actual mejor punto x^c más el vector resultante de la diferencia entre el mejor punto en la iteración anterior x^{k-1} y el punto actual $x^k + 1$. En caso de que el movimiento exploratorio no fuese exitoso se decrementa el parámetro Δ, haciendo la perturbación más pequeña, y el algoritmo regresa al paso exploratorio.

- *Método de direcciones conjugadas de Powell.* La idea es crear un conjunto de direcciones de búsquedas linealmente independientes y realizar búsquedas unidireccionales sobre estas direcciones para hallar el óptimo comenzando de la mejor posición anterior en cada iteración. El método está diseñado para funciones cuadráticas, para las cuales solo requiere de cuatro búsquedas unidireccionales para encontrar el óptimo, aunque también es aplicable para funciones polinomiales.

2.5.2. Métodos basados en gradiente

A diferencia de los métodos directos, los basados en gradiente emplean menos evaluaciones al hacer uso de la información de las derivadas de la función. Desafortunadamente las derivadas no son accesibles en algunas funciones como aquellas con variables discretas, discontinuas, etc. Pero cuando las derivadas son accesibles son los métodos más eficientes.

Algunos de estos algoritmos solo requieren de la primera derivada de la función objetivo, mientras que otros necesitan de la primera y segunda [21]. El

vector gradiente de la derivada de primer orden $\bigtriangledown f(x^{(t)})$ representa la dirección de máximo incremento en los valores de la función objetivo, y en consecuencia $-\bigtriangledown f(x^{(t)})$ resulta en la dirección de mayor decremento en los valores de la función objetivo.

Entre estos algoritmos se encuentran [21]:

- *Método de descenso máximo de Cauchy.* Este método parte de un punto cualquiera y calcula su gradiente, posteriormente se realiza una búsqueda unidireccional en la dirección negativa del gradiente. El resultado de esta búsqueda se utiliza como punto de partida para efectuar otra búsqueda unidireccional en el eje del gradiente negativo. El proceso se repite hasta que el vector gradiente sea suficientemente pequeño.

- *Método de Newton.* El método de Newton es prácticamente igual al de Cauchy, con la diferencia de emplear de la segunda derivada, haciendo más veloz la convergencia. Desafortunadamente la matriz Hessiana resultante (matriz que contiene todas las derivadas parciales de segundo orden), además de ser costosa, requiere de ciertas características para tener dirección descendiente, por lo que se recomienda reiniciar con un nuevo punto cuando la dirección de búsqueda no sea descendiente.

- *Método de Madquardt.* Este método aplica los métodos de Cauchy y Newton dependiendo de los resultados intermedios. La motivación se encuentra en que el método de Cauchy es útil cuando el punto inicial se encuentra lejos del mínimo, mientras que el método de Newton funciona mejor cuando el punto de partida se ubica cerca al óptimo.

- *Método de gradiente conjugado.* Similar que en el método de direcciones conjugadas de Powell, esta técnica asume que la función objetivo es cuadrática. La primera dirección de búsqueda que aplica es el gradiente negativo, posteriormente emplea una función recursiva que emplea los cuadrados del gradiente de la posición actual y pasada, así como la dirección de búsqueda anterior. Al aplicarse en funciones lineales o cuadráticas el método de gradiente conjugado es capaz de encontrar el mínimo con pocas iteraciones, mientras que para otros problemas son necesarios más cálculos de direcciones y búsquedas unidireccionales.

- *Método de métrica variable o método DFP.* Este algoritmo realiza una aproximación a la matriz Hessiana, la cual necesita del cómputo costoso de la derivada de segundo orden. Dicha aproximación comienza con la matriz identidad y a través de derivadas de primer orden sucesivas se asemeja a la matriz Hessiana. Gracias a comenzar con la matriz identidad se cumplen los requisitos necesarios para que la dirección de búsqueda sea siempre descendiente.

2.6. Ventajas y desventajas de la optimización clásica

Las técnicas de optimización clásicas garantizan encontrar el óptimo cuando se cumplen las condiciones de aplicación del algoritmo en cuestión, de ser así el resultado es accesible en pocas iteraciones.

Desafortunadamente los métodos de optimización clásica demuestran dificultades cuando se enfrentan a problemas complejos. La mayor dificultad se presenta cuando se desea usar un mismo algoritmo para resolver diferentes problemas. Esto se debe a que cada método clásico está diseñado para resolver de manera eficiente un tipo de problema en particular, fallando al ser usados en el amplio catálogo de problemas del mundo real a los que puede enfrentarse un usuario [1].

Existen alternativas heurísticas que son más flexibles que las técnicas clásicas, entre ellas destacan los algoritmos evolutivos, búsquedas por recocido simulado, etc. Dichas técnicas suelen emular un fenómeno de la naturaleza para resolver problemas de búsquedas y optimización.

Particularmente los algoritmos evolutivos se presentan como una de las técnicas más interesantes para resolver problemas de optimización cuyas características no permiten ser resueltos a través de técnicas clásicas. Tal es el caso en este trabajo, donde se decantó por algoritmos evolutivos para resolver conjuntos de problemas de optimización con restricciones con diferentes características.

En el siguiente capítulo se aborda de manera general el cómputo evolutivo, partiendo de los conceptos básicos hasta los algoritmos relevantes para este trabajo.

Capítulo 3

Cómputo Evolutivo

El cómputo evolutivo es un área de las ciencias de la computación que se encarga de la resolución de problemas utilizando mecanismos inspirados en la naturaleza, con énfasis en la evolución natural. La supervivencia y adaptabilidad de las especies denota el poder de la evolución natural como estrategia para la resolución de problemas, donde los individuos progresivamente desarrollan características especiales para prosperar en sus respectivos ecosistemas.

La idea central del cómputo evolutivo es trasladar el proceso de evolución a un modelo para la solución de problemas mediante prueba y error; donde las soluciones potenciales al problema representan individuos y la calidad de dichas soluciones hace referencia al nivel de aptitud al ambiente por parte de las especies. Conforme avanzan las generaciones se espera que la calidad de las soluciones mejore y que eventualmente una de ellas resuelva exitosamente el problema en cuestión [2].

Para una mejor comprensión de todo lo referente a esta rama de la computación es necesario comentar las bases de las teorías biológicas que la inspiran.

3.1. Inspiración biológica

La teoría de la evolución natural de Charles Darwin expresa que las especies se encuentran en constante disputa por los recursos de su entorno y por reproducirse. Dado que el entorno sólo puede albergar un número limitado de individuos simultáneamente, aquellos con las mejores características para dicho entorno tienen mayores posibilidades de permanecer y reproducirse que el resto de individuos. Al fenómeno anterior se le conoce como selección natural.

Otro elemento de esta teoría consiste en las variaciones en el fenotipo (conjunto de elementos que influyen en la aptitud de un individuo) de los individuos, las cuales pueden ser heredadas de los padres o causadas por mutaciones aleatorias. Con estas variaciones se crean nuevas combinaciones las cuales son 'evaluadas'

por el entorno, de tal suerte, los mejores individuos sobreviven y sus características son heredadas permitiendo que la evolución continúe.

Gregor Mendel sentó las bases de la genética, identificando las dinámicas de herencia entre genes dominantes y recesivos. Con ello demostró que las características visibles y directamente relacionadas con la aptitud que los individuos presentan son codificadas durante la reproducción. Posteriormente se llegaría a la cuenta de que toda la vida en el planeta está descrita por diferentes con figuraciones de codones en el ADN.

Al unificarse la teoría Darwiniana de la evolución y la genética se puede apreciar el origen de la vida. Donde todos los individuos del planeta almacenan un código invisible de sus características observables. La selección natural causa que la calidad de dichas características dicte sus posibilidades de reproducirse y sobrevivir. Como consecuencia las configuraciones genéticas que dieron mejor resultado en el entorno serán replicadas en el tiempo.

Bajo esta óptica es posible interpretar la evolución de las especies como un proceso de optimización que progresivamente mejora el nivel de adaptabilidad de los individuos al entorno. El éxito probado de la evolución vuelve atractiva la metáfora que propone el cómputo evolutivo para la resolución de problemas.

3.2. Componentes de un algoritmo evolutivo

El Cómputo Evolutivo consiste en modelos computacionales de los procesos evolutivos para construir solucionadores de problemas de carácter general [23]. Su poder reside en aplicar simultáneamente métodos de diversificación de las soluciones para facilitar la exploración y mecanismos de selección que permitan mejorar la media de aptitud de la población. Progresivamente el trabajo en conjunto de ambos elementos permite conseguir buenas soluciones al problema en un tiempo aceptable [2]. Cabe destacar que el componente aleatorio de estos algoritmos los convierte en técnicas estocásticas, las cuales no pueden asegurar que se encontrará infaliblemente la solución óptima, no obstante es posible obtener una solución competitiva.

En resumen los algoritmos pertenecientes a este paradigma presentan el siguiente esquema (véase el Algoritmo 1): Se crea una población aleatoria de soluciones potenciales al problema (línea 2). La aptitud de dicha población es evaluada con respecto a la función objetivo (línea 3), cuanto mayor sea el valor de aptitud de un individuo mejor es considerado (en el caso de maximización). Los mejores individuos son elegidos como generadores de nuevas soluciones (línea 5), tambien llamados como padres. Ejecutando operaciones de cruza y/o mutación en los padres (líneas 6 y 7) se consigue un conjunto de nuevas soluciones potenciales al problema. Estos nuevos individuos también son evaluados para competir con el conjunto perteneciente a la primera generación (línea 8). En principio los

Algoritmo 1: Pseudocódigo de un algoritmo evolutivo [2].

```
1  inicio
2      INICIALIZAR población inicial con soluciones candidatas aleatorias.
3      EVALUAR soluciones.
4      repite
5          SELECCIONAR padres.
6          RECOMBINAR pares de padres.
7          MUTAR los descendientes resultantes.
8          EVALUAR nuevas soluciones.
9          SELECCIONAR individuos para la siguiente generación.
10     hasta CONDICIÓN DE PARO satisfecha;
11 fin
```

mejores de ambos grupos permanecerán y el resto será desechado (línea 9). El proceso anterior (líneas 5-9) puede repetirse hasta que se encuentre una solución con calidad aceptable o que algún otro criterio se cumpla (línea 10) [2].

A pesar de existir variantes del esquema general de un algoritmo evolutivo los cambios son mínimos y de carácter técnico. En el siguiente apartado se describen brevemente los componentes de un algoritmo evolutivo.

3.2.1. Representación

La representación de las soluciones consiste en la codificación de los individuos en el espacio de búsqueda donde trabaja el algoritmo para que éstos puedan ser manipulados por una computadora. Podría entenderse como el vínculo entre el problema real y el algoritmo evolutivo. La representación debe ser una abstracción sencilla que permita describir todas las combinaciones posibles de soluciones al problema en turno [2].

Al individuo expresado en la representación seleccionada se le conoce como genotipo o cromosoma, y a su versión decodificada para el problema se le llama fenotipo. Usualmente el genotipo es un vector de una estructura de dato sencilla del tamaño de las dimensiones del problema, a cada posición en el cromosoma se le llama *gen* y al valor que presenta una solución en una posición en específico se le conoce como *alelo*.

Por lo anterior es común encontrar en la literatura especializada indicaciones simples como 'codificación binaria' o 'representación real', haciendo referencia a la estructura de datos con la que se tratará a nivel genotipo.

En este trabajo se resuelven problemas de optimización continua con restricciones, de tal suerte que se emplea un esquema de representación que consiste en un arreglo de números reales del tamaño de las dimensiones del problema en

cuestión.

3.2.2. Población

En esencia se refiere al conjunto de individuos con los que trabaja el algoritmo, usualmente de tamaño fijo y causante de competencia entre las soluciones. Al igual que en la naturaleza, las interacciones de los individuos de la población con el entorno son las que dan origen a la evolución; pues por sí mismos los individuos son incapaces de evolucionar. Por esta razón un concepto importante en el cómputo evolutivo es la diversidad, medida que aproxima el número de soluciones diferentes en la población en determinado momento, pues si la población inicial no es lo suficientemente diversa es posible que los operadores de cruza y mutación tornen homogénea a la población muy rápidamente.

3.2.3. Selección de padres

Este paso del algoritmo pretende hacer una diferenciación entre la calidad de los individuos, favoreciendo las probabilidades de las mejores soluciones de convertirse en padres en la siguiente generación, pero también dejando abierta una pequeña ventana de posibilidad para los individuos con peores valores de la función objetivo.

Tanto la selección de padres como el reemplazo al final de cada generación promueven la mejora en los valores de aptitud de la población.

Algunos esquemas comunes para la selección de padres son:

- **Selección proporcional.** Este tipo de selección escoge individuos como padres a partir de su valor de aptitud. La probabilidad que un individuo sea elegido es directamente proporcional a su influencia en la aptitud general de la población.

  Métodos como la selección por ruleta [24], el Sobrante estocástico [25], la selección estocástica universal [26] o el muestreo determinístico [24] son ejemplos de selección proporcional.

- **Selección jerárquica.** La idea básica es ordenar la población de mejor a peor y asignar el número de copias de cada individuo de acuerdo a una función descendente y posteriormente ejecutar alguna técnica de selección proporcional usando el número de copias establecidas en vez del valor de aptitud.

  La forma más común de selección jerárquica emplea funciones lineales no negativas que decrementan sus valores de acuerdo a ciertas constantes [27].

- **Selección por torneo.** Consiste en escoger un conjunto aleatorio de individuos de la población, tantos como se defina en el *tamaño del torneo.*

De este grupo se selecciona la mejor solución como padre. El proceso se repite hasta que se obtiene el número esperado de padres. Los torneos más habituales involucran únicamente a dos individuos, aunque existen trabajos donde se han utilizado tamaños de torneo mayores.

Existen dos tipos de selección por torneo: la versión determinística, donde siempre el mejor individuo del torneo es elegido como padre; y la versión probabilista, la cual selecciona a la mejor solución bajo cierta probabilidad decrementando la presión de selección y consecuentemente favoreciendo la diversidad [27].

3.2.4. Operadores de variación

Estos operadores tienen como objetivo generar nuevos individuos a partir de los existentes. Existen dos tipos de operadores dependiendo del número de soluciones que necesitan para crear un nuevo individuo: mutación (un individuo) y cruza (más de un individuo) [2].

- **Mutación.** Es un operador unario que hace pequeñas modificaciones en el genotipo de manera estocástica, pues no existe ningún sesgo respecto a qué gen modificará su valor.

  La mutación tiene diferentes tareas en cada familia de algoritmos evolutivos pues es utilizada en mayor o menor medida. Por ejemplo, en un algoritmo genético es un operador secundario que permite que se mantenga cierto nivel de diversidad en la población, mientras que en la programación genética la mutación es el único factor de cambio en la población al no existir operador de cruza [2].

- **Cruza.** La recombinación o cruza es un operador que trabaja sobre dos o más individuos para generar nuevos individuos. Al igual que la mutación, las decisiones de cómo combinar a los individuos son realizadas de manera estocástica.

  La idea general es que al combinar dos individuos genéticamente distintos y con buenos valores de aptitud, propiedad que persigue la selección de padres, se espera que los descendientes posean a su vez características deseables provenientes de ambos padres.

  Es importante destacar que la recombinación, al igual que la mutación, es dependiente de la representación de soluciones elegida.

3.2.5. Reemplazo

El reemplazo o selección de supervivientes es el equivalente en los algoritmos evolutivos de la supervivencia del más apto. Al igual que la selección de

padres, discrimina las malas soluciones de las mejores en favor de la explotación; con la diferencia de ser aplicado al final del ciclo evolutivo, justo después de haber generado y evaluado a los descendientes. El reemplazo moldea la población de la siguiente generación al decidir cuáles soluciones son descartadas y cuáles continúan.

Generalmente es determinista con base en los valores de aptitud de cada individuo, aunque existen trabajos donde factores como el tiempo que las soluciones han permanecido en la población son considerados [2]. Estrategias como el elitismo, donde el mejor individuo de la población garantiza su permanencia en la siguiente generación, son consideradas parte del reemplazo.

3.3. Principales paradigmas

El cómputo evolutivo tiene sus inicios en la década de los 40's [2, 23] con propuestas y trabajos iniciales de Turing, Bremermann, etc [2]; pero fue hasta los 60's que comenzaron a formarse los principales paradigmas desde diferentes latitudes: *Programación Evolutiva*, *Estrategias Evolutivas* y *Algoritmos Genéticos*; los cuales se describen en los siguientes párrafos.

3.3.1. Programación evolutiva

Desarrollada por Fogel en 1966, la programación evolutiva (PE por sus siglas) es un paradigma que pretendía simular la evolución natural a través de un proceso de aprendizaje para generar inteligencia artificial, considerando inteligencia como adaptabilidad al entorno [2].

En programación evolutiva no existe una representación por excelencia, a diferencia de los algoritmos genéticos por ejemplo; la representación es seleccionada para satisfacer al problema.

Posiblemente la principal característica de la PE es prescindir del operador de cruza y sólo valerse de la mutación para generar nuevos individuos. En cada generación los N individuos de la población son seleccionados para mutar, resultando en un total de $2N$ soluciones de las que se debe seleccionar N para volver al tamaño poblacional original [23].

Un elemento importante es el tipo de mutación que se aplica en PE, conocido como perturbación Gaussiana (en el caso de representación real). Dicha mutación es más flexible y puede provocar cambios similares a los que se alcanzan ejecutando cruza si se desea [23].

3.3.2. Estrategias evolutivas

Inventada en los 60's por los alemanes Ingo Rechenberg y Hans-Paul Schwefel. Debido a la inquietud de resolver problemas hidrodinámicos, el área se ha

desempeñado usando representaciones reales [23].

En sus inicios las estrategias evolutivas (ES por sus siglas en inglés) contemplaban un sólo individuo que generaba un descendiente en cada generación, el reemplazo podía darse de dos maneras: en la $(1 + 1) - EE$ el hijo sustituía al padre en caso de ser mejor; a diferencia de la $(1, 1) - EE$, donde el reemplazo del padre es realizado independientemente de los valores de aptitud de ambos individuos.

Años más tarde llegaría el concepto de población a las EE, así como la nomenclatura característica en la que se emplea μ para referirse a la población y λ para el número de descendientes esperados en cada generación. Dando lugar a las versiones $(\mu + \lambda) - EE$ y $(\mu, \lambda) - EE$.

Otro factor trascendental en la influencia de las EE en el área ha sido la innovación en torno a adaptación y auto-adaptación, impulsada en un principio por la regla de 1/5 de éxito propuesta por Rechenberg para ajustar la desviación estándar (factor de mutación) a partir de la razón de mutaciones exitosas y mutaciones realizadas [2].

En los algoritmos de estrategias evolutivas más sofisticados se implementan técnicas de auto-adaptación de la desviación estándar y el ángulo de rotación [2].

La auto-adaptación de parámetros consiste en introducir los parámetros dentro del individuo y someterlo a evolución. Del mismo modo en que se espera que conforme pasan las iteraciones del proceso evolutivo los elementos de los individuos se ajusten para tener mejores valores de aptitud, también se pretende que los parámetros de cada individuo mejoren sus posibilidades de obtener mejores posiciones en el espacio de búsqueda; y por consecuencia mejores valores de la función objetivo.

3.3.3. Algoritmos genéticos

Propuestos por Holland en 1975 para estudiar el comportamiento adaptativo en sistemas naturales y artificiales [2], son el paradigma más famoso del cómputo evolutivo.

El algoritmo básico también conocido como SGA (*Simple Genetic Algorithm*) cuenta con representación binaria usando una cadena de bits, operador de selección proporcional, mutación *bit-flip* con baja probabilidad y énfasis especial en la recombinación como operador principal para generar nuevos individuos [23]. La mutación pasa a un plano secundario, siendo aplicada en pocas ocasiones.

En términos generales un algoritmo genético inicializa la población, posteriormente selecciona los N padres con algún operador de selección proporcional con base en los valores de aptitud, genera N descendientes a los que aplicará mutación con una probabilidad pequeña, la nueva generación sustituye a la anterior y el proceso se repite hasta cumplirse una condición de paro [23].

Gracias a su fama y al tiempo que llevan siendo estudiados, los últimos algoritmos genéticos han mejorado considerablemente los mecanismos del SGA. Se ha descartado la selección proporcional en favor de selección jerárquica, se han incluido elementos para incrementar la velocidad de convergencia (como el elitismo), después de haber descubierto los problemas de codificación binaria se han desarrollado algoritmos con otras representaciones, etc. [2]

3.4. Algoritmos destacados

3.4.1. Optimización por cúmulos de partículas PSO

La optimización por cúmulos de partículas (o PSO por sus siglas en inglés) difiere un poco de otros algoritmos evolutivos, al intercambiar la metáfora evolutiva por un movimiento inspirado en el comportamiento social de algunas especies, como el vuelo de parvadas de aves o el nado de bancos de peces; donde los movimientos de los individuos son influidos por el colectivo y viceversa.

A pesar de no existir evolución propiamente, el comportamiento algorítmico se ajusta al esquema general de un algoritmo evolutivo [2].

PSO carece de operador de cruza, de modo que las soluciones serán siempre las mismas. Las soluciones en PSO se conocen como partículas y están formadas por tres partes: la posición actual de la partícula x_i, la velocidad v_i y la posición con el mejor valor de aptitud visitado por dicha partícula $pbest_i$ [28]. Gracias a que las soluciones almacenan su respectivo $pbest_i$, las partículas pueden desplazarse libremente por todo el espacio de búsqueda sin perder de vista las zonas promisorias conocidas. A la mejor posición antes visitada $pbest_i$ de toda la población se le conoce como $gbest$.

Al iniciar el algoritmo se asignan valores aleatorios a todos lo componentes de las partículas, donde x_i y $pbest$ tienen el mismo valor a la primera iteración.

Conforme pasan las iteraciones se calcula la velocidad de las partículas de acuerdo a la siguiente fórmula [2]:

$$v_i' = w \cdot v_i + c_1 r_1 (pbest_i - x_i) + c_2 r_2 (gbest - x_i)$$

Donde v_i es la velocidad anterior de la partícula x_i con mejor posición $pbest_i$. c_1 y c_2 son parámetros responsables de la influencia personal y social: c_1 favorece la dirección hacia la mejor posición conocida por la partícula $pbest_i$, mientras que c_2 atrae a las soluciones a la mejor posición de toda la población $gbest$. r_1 y r_2 son números reales aleatorios en el intervalo $(0, 1)$ y w es una constante llamada inercia, la cual dificulta los cambios de dirección en la velocidad.

La nueva velocidad de una partícula dependerá de la velocidad anterior, el peso del factor social y personal, además del componente estocástico. Una vez

calculada la velocidad, se aplica el operador de vuelo en cada partícula:

$$x'_i = x_i + v'_i$$

Al final de la iteración las partículas actualizan el valor de $pbest_i$ en caso de haber llegado a un punto con mejor valor de aptitud.

Existen diversas variantes de PSO, algunas de ellas designan vecindarios de partículas. De tal suerte que el comportamiento social no sea liderado por la mejor posición de toda la población, si no por los líderes de cada vecindario. Al enfoque anterior se le conoce como PSO local best [29].

Para el Experimento 1 de este trabajo se implementó un algoritmo del estado del arte con base en PSO, llamado SAM-PSO [30], el cual será descrito posteriormente en el documento.

3.4.2. Evolución diferencial

Evolución diferencial es un algoritmo de optimización propuesto por Storn y Price en 1995. Es un algoritmo sencillo cuya principal característica es su operador de mutación: la mutación diferencial.

En cada iteración, por cada uno de los NP individuos de la población se genera un vector mutante:

$$u_{i,G+1} = x_{r1,G} + F \cdot (x_{r2,G} - x_{r3,G})$$

El vector de perturbación, o vector mutante, resulta de la suma del vector base $x_{r_1,G}$ más la diferencia entre dos vectores $x_{r_2,G}$ y $x_{r_3,G}$ (también aleatorios), escalados de acuerdo al parámetro F [31]. Donde $r_1, r_2, r_3 \in \{1, 2, \ldots, NP\} \wedge r_1 \neq r_2 \neq r_3$

Después de la mutación, el siguiente paso es aplicar el operador de cruza entre el individuo original (también conocido como *target*) y el vector de perturbación:

$$u_{j,i,G+1} = \begin{cases} v_{j,i,G+1} & \text{si } (aleatorio(j) > CR) \text{ o } j = k; \\ x_{j,i,G} & \text{de otro modo} \end{cases}$$

El producto de la cruza tendrá valores del vector original y mutante dependiendo de acuerdo al parámetro CR, o probabilidad de cruza. El operador garantiza la inclusión de al menos un elemento del vector mutante cuando el contador j sea igual a la posición garantizada para el vector mutante k, la cual es seleccionada aleatoriamente entre todas las dimensiones.

Finalmente, si el valor de aptitud del nuevo individuo generado a partir de la cruza $f(u_{i,G+1})$ es mejor que el correspondiente al target $f(x_{i,G})$, el vector original es remplazado por el nuevo individuo.

Debido al éxito de la evolución diferencial en el cómputo evolutivo, continúa siendo una de las técnicas más importantes. Existen diversas variantes al algoritmo, algunas de ellas modifican los criterios de selección de los vectores involucrados en la mutación. El esquema mencionado corresponde a la estrategia original, $DE\backslash rand\backslash 1\backslash bin$; donde DE significa Evolución Diferencial, *rand* hace referencia a que los vectores son seleccionados de manera aleatoria, 1 al número de vectores diferencia y *bin* al tipo de cruza empleada, en este caso cruza binomial.

Bajo la misma nomenclatura se han definido otros esquemas, por ejemplo $DE\backslash best\backslash 1\backslash bin$, éste modifica el criterio de selección del vector base en la mutación; en vez de escoger un individuo de la población aleatoriamente $x_{r1,G}$, se emplea el mejor individuo de la población como vector base $x_{best,G}$.

En este trabajo dos versiones de evolución diferencial fueron utilizadas en la experimentación, la versión clásica $DE\backslash rand\backslash 1\backslash bin$ y un algoritmo del estado del arte para optimización con restricciones basado en evolución diferencial [32].

3.5. Manejo de restricciones

Para que un algoritmo evolutivo pueda resolver problemas de optimización con restricciones, como es el caso en la mayoría de los problemas del mundo real [21], es necesario incluir una técnica de manejo de restricciones.

Se han propuesto diversos manejadores de restricciones, los cuales pueden clasificarse dentro de los siguientes grupos [33]:

1. **Funciones de penalización:** La idea base es transformar el problema en uno no restringido al modificar el valor de aptitud de las soluciones que violen las restricciones.

2. **Representación y operadores especiales:** Otro enfoque resulta en modificar la representación de la soluciones y los operadores del algoritmo para simplificar la forma y tamaño de búsqueda, con el objetivo de preservar la factibilidad de las soluciones.

3. **Algoritmos de reparación:** Las técnicas de reparación atacan el problema desde el otro extremo: ¿Cómo modificar una solución no factible para llevarla a la zona factible? En vez de definir penalizaciones o modificar la representación para dar con individuos factibles.

4. **Separación de objetivos y restricciones:** Una estrategia diferente resulta en tratar por separado la función objetivo y las restricciones. Ejemplos de este tipo de técnicas son la co-evolución, la superioridad de puntos o la memoria conductual.

5. **Métodos híbridos:** Finalmente los métodos híbridos albergan mecanismos de diferente índole, al no poder ser clasificados en los grupos anteriores. Entre ellos se encuentran sistemas inmunes artificiales, técnicas de lógica difusa, etc.

Algunos de los mecanismos que pertenecen a estas categorías requieren de un diseño específico dependiente del problema; como es el caso de las funciones de penalización, algoritmos de reparación, representación y operadores especiales, etc.

A continuación se describen tres técnicas de manejo de restricciones que han sido aplicadas en diversos algoritmos del estado del arte. Estos manejadores de restricciones pertenecen a la clase de separación de objetivos y restricciones, requieren de pocos parámetros y no necesitan de grandes modificaciones para ser incluidos en un algoritmo evolutivo cualquiera, haciéndolos particularmente atractivos para el comparativo que presenta este trabajo.

3.5.1. Reglas de factibilidad de Deb

Consiste en pautas para la jerarquización de soluciones sin necesidad de parámetros aplicando las siguientes reglas para seleccionar el mejor entre dos individuos [19]:

1. Una solución factible es siempre preferible que una no factible.
2. Entre dos soluciones factibles la que posee mejor valor de aptitud es seleccionada.
3. Entre dos soluciones no factibles aquella con la menor suma de violación de restricciones $\phi(x)$ es considerada mejor.

3.5.2. ϵ constrained method

ϵ constrained method [34] lidia con las restricciones de manera progresiva: al principio permite que las soluciones no factibles cuya suma de violación de restricciones no excede cierto umbral ϵ sean comparadas únicamente por los valores de aptitud, conforme la avanzan las generaciones el umbral es reducido gradualmente hasta que no existe tolerancia a soluciones no factibles.

$$\phi = \sum_{j}^{J} ||max(0, g_j(x))||^p + \sum_{k}^{K} ||h_k(x)||^p$$

Sea ϕ la suma de violación de restricciones, donde p es un número positivo, h y g representan restricciones de desigualdad e igualdad. La comparación entre

dos individuos x_1 y x_2 por el método ϵ constrained se calcula de la siguiente manera:

$$(f(x_1), \phi(x_1)) <_\epsilon (f(x_2), \phi(x_2)) \longleftrightarrow \begin{cases} f(x_1) < f(x_2), & \text{si } \phi(x_1), \phi(x_2) < \epsilon; \\ f(x_1) < f(x_2), & \text{si } \phi(x_1) = \phi(x_2); \\ \phi(x_1) < \phi(x_2), & \text{de otro modo} \end{cases}$$

Si la suma de violación de ambas soluciones es igual o es menor que el umbral ϵ los individuos se comparan por valores de aptitud, en caso contrario la comparación se efectúa en términos de suma de violación de restricciones.

$$\epsilon(0) = \phi(x_\theta) \quad \epsilon(t) = \begin{cases} \epsilon(0)(1 - \frac{t}{Tc})^{cp}, & 0 < t < Tc; \\ 0, & t \geq Tc \end{cases}$$

El umbral reducirá su valor en cada iteración de usando la regla anterior, donde t es la iteración actual, Tc el número máximo de iteraciones y cp la tasa de reducción de ϵ.

3.5.3. Stochastic ranking

Stochastic ranking, propuesto por Runarsson y Yao [35], es un manejador de restricciones que intenta encontrar un balance entre soluciones factibles y no factibles. La idea es simple, a partir una probabilidad se decide el tipo de comparación que se realizará. Si dos soluciones son factibles, la comparación siempre será realizada usando los valores de aptitud. En caso contrario de acuerdo a la probabilidad P_f se determina si los individuos serán comparados por suma de violación de restricciones o por sus respectivos valores de la función objetivo.

Cabe destacar que Stochastic ranking fue diseñada como un operador de selección de padres, por lo que el proceso de evaluar la probabilidad para elegir el tipo de comparación es realizado en conjunción con el algoritmo de ordenamiento burbuja para seleccionar a los individuos en las posiciones más altas como padres. El ordenamiento es limitado por el parámetro λ, el cual indica el número de individuos que serán ordenados.

El Algoritmo 2 presenta el detalle de este método, el cual se describe a continuación:

Durante N iteraciones (línea 2) se comparan los individuos contiguos desde la posición $j = 1$ hasta $j = \lambda - 1$ (línea 3), donde N es el tamaño de la población. En cada iteración del segundo ciclo se genera un número aleatorio $u \in (0, 1)$ (línea 4), posteriormente la línea 5 en el Algoritmo 2 analiza si los dos individuos comparados x_j y x_{j+1} son factibles, o en su defecto, el número aleatorio u es menor que el valor del parámetro P_f. Si alguna de las dos condiciones anteriores se cumple la comparación entre x_j y x_{j+1} se llevará a cabo en términos de valores

aptitud (línea 6) y en caso de que el individuo x_{j+1} posea un mejor valor de aptitud que la solución x_j éstos intercambiarán su posición en la lista jerarquizada (línea 7).

Si ninguna de las dos condiciones de la línea 6 se cumple la comparación entre los indiviudos x_j y x_{j+1} se realiza contrastando la suma de violación de restriciones (línea 9), si el individuo x_{j+1} presenta menor suma de violación de restricciones que la solución x_j se realiza un intercambio (línea 10).

Una vez que se ha concluido una iteración sin realizar intercambios se detiene el algoritmo (línea 14).

Algoritmo 2: Stochastic ranking [35].

```
1  inicio
2      para i = 1 a N hacer
3          para j = 1 a λ − 1 hacer
4              aleatorio u ∈ (0, 1)
5              si (φ(x_j) = 0 ∧ φ(x_{j+1}) = 0) ∨ (u < P_f) entonces
6                  si f(x_j) > f(x_{j+1}) entonces
7                      intercambiar(x_j,x_{j+1})
8                  fin
9                  si no, si φ(x_j) > φ(x_{j+1}) entonces
10                     intercambiar(x_j,x_{j+1})
11                 fin
12             fin
13         fin
14         Si no se realizó intercambio, romper ciclo
15     fin
16 fin
```

Capítulo 4

Diversidad

4.1. Concepto de diversidad

> *El progreso en la evolución depende fundamentalmente en la existencia de variación en la población. Desafortunadamente un problema clave en el cómputo evolutivo es la pérdida de diversidad a través de la convergencia prematura. Esta falta de diversidad a menudo lleva al estancamiento, pues el sistema se encuentra atrapado en óptimos locales, perdiendo la variedad genética necesaria para escapar.*
>
> Macphee et al., 1999

La cita anterior podría resumir la pertinencia de considerar la diversidad como un elemento central en el cómputo evolutivo. Tan es así que existe un consenso en el área respecto a la importante relación de la diversidad con el desempeño en general de un algoritmo evolutivo [36], especialmente cuando se intenta evitar la convergencia prematura o escapar de óptimos locales [10].

Establecer una definición absoluta de diversidad ha sido un tanto problemático, ya que ésta puede darse en distintos niveles de granularidad. En general se tiene una concepto de diversidad arropado por toda el área que engloba las ideas principales, permitiendo que las medidas de diversidad lo interpreten de maneras ligeramente distintas.

La diversidad se refiere a las diferencias entre individuos [37], también puede ser vista como un indicador de la disimilitud entre las soluciones [6], una medida que aproxima el número de diferentes individuos en la población en determinado momento [2].

4.2. Balance exploración/explotación

El comportamiento de un algoritmo evolutivo está determinado por la relación entre exploración y explotación (E/E) durante la ejecución [13]. En cierta medida el CE ha sido exitoso debido a su notable compromiso E/E [36] en comparación con otras metaheurísticas.

Debido a la gran cantidad de algoritmos pertenecientes al cómputo evolutivo, delimitar exploración y explotación es complicado, pues no existe un consenso de los significados de ambos conceptos [2]. Por ejemplo, en el caso de los algoritmos genéticos varios trabajos consideran a los operadores de variación (cruza y mutación) como elementos en favor de la exploración y a los procesos de selección como promotores de explotación, mientras que otros como Wong [38] interpretan la mutación como operador explorativo y vinculan la cruza con la explotación.

A los anteriores desacuerdos puede sumarse la reflexión de Črepinšek [37], donde presenta buenos argumentos para interpretar ambos operadores de variación tanto explotativos como explorativos. En el caso de la mutación, ésta podría considerarse en favor de la exploración al modificar aleatoriamente la población e incrementar la diversidad, o en pro de la explotación; ya que conserva la mayor parte del material genético del individuo al que se aplica. La cruza es explotativa si se contempla que se combinan padres aptos para tener mayores posibilidades de generar aún mejores descendientes; por otro lado los hijos resultantes de la recombinación pueden caer en zonas no exploradas, y por lo tanto el operador tendría carácter explorativo. Al número aproximado de maneras de generar nuevos individuos por parte de los operadores de variación se le conoce como *poder explorativo* [39].

De lo anterior se puede concluir que no existe un operador totalmente de exploración o explotación y que cada proceso debe ser analizado en contexto con el resto de mecanismos del algoritmo utilizado. Exploración y explotación no son necesariamente fuerzas opuestas, si no magnitudes ortogonales [14] afectadas por todos los elementos del algoritmo: operadores de variación, presión de selección, tamaño de la población, representación de soluciones y configuración de parámetros [37].

Históricamente la manera de controlar el compromiso entre exploración y explotación ha sido a través del manejo de parámetros, pues las distintas combinaciones de estos dictarán gran parte del comportamiento del algoritmo. Existen diversas estrategias para elegir la composición de parámetros: prueba y error, utilizar los valores de parámetros sugeridos, análisis estadístico para apoyarse en la selección o el ajuste y control de parámetros [37].

Las técnicas de selección de parámetros han demostrado mejorar el desempeño de los algoritmos efectivamente, sin embargo son una manera indirecta de controlar el balance entre exploración y explotación [37]. Cómo medir tal balance

continúa como una pregunta abierta [40]. En este sentido, las medidas de diversidad surgen como métodos cuantitativos para sensar la dinámica de la población y así intentar dilucidar entre exploración y explotación [37].

A pesar del reconocimiento que se da a la diversidad respecto al desempeño de los algoritmos evolutivos, la relación entre diversidad, exploración y explotación sigue sin ser clara; mayor investigación es necesaria, en especial para identificar los tipos y las magnitudes de diversidad en las diferentes etapas del proceso evolutivo [37].

4.3. Técnicas de promoción de la diversidad

Una de las maneras más directas para controlar el balance entre exploración y explotación es a través de técnicas de promoción de diversidad.

A partir de la década de los noventas, surgieron numerosos trabajos con diferentes enfoques para mantener la diversidad.

A continuación se describe a grandes rasgos la clasificación que Črepinšek [37] ofrece respecto a los mecanismos para la promoción de la diversidad en la literatura. Posteriormente se explican las técnicas de diversidad relevantes para este trabajo.

4.3.1. Taxonomía

- **Técnicas de nitching**

  El concepto de nitching, acuñado por De Jong, propone retomar la metáfora natural y visualizar el espacio de búsqueda como un ecosistema donde los recursos son limitados y deben ser compartidos por los individuos de la población. Al aplicar alguna de estas estrategias progresivamente emergerán nichos, o clústers marcados, de individuos que comparten una sección del espacio de búsqueda (el recurso compartido). Lo anterior permite explorar diferentes zonas simultáneamente, a diferencia de la convergencia a un punto que presentan los algoritmos evolutivos tradicionales. El nitching es particularmente atractivo cuando se intenta optimizar funciones multimodales [9].

  Las técnicas de nitching se dividen en:

  1. **Basadas en fitness-sharing:** El fitness-sharing intenta reducir la influencia de las regiones más pobladas en el paisaje de aptitud para permitir que las zonas con menos individuos no sean desocupadas. Específicamente modifica el valor de aptitud de cada solución en la población dividiéndolo entre el número de individuos dentro del nicho en el que se encuentra. La manera de delimitar los nichos es a través

del parámetro radio de nicho σ, el cual es uno de los principales problemas ya que la distancia entre nichos es dependiente del problema [8].

Existen otras variantes de fitness-sharing como el clearing [49], o trabajos donde el parámetro σ se actualiza de acuerdo a la iteración de la búsqueda [50].

2. **Basadas en reemplazo:** Črepinšek incluye también a las técnicas de crowding en este apartado, pues al aplicarse se establecen competencias directas entre padres e hijos cercanos entre sí, lo cual resulta en la emergencia de nichos [37].

- **Técnicas sin nitching**

1. **Basadas en población**
   - *Incrementar tamaño de población:* Uno de los enfoques más simples, consiste en incrementar el tamaño de población para así también aumentar las posibilidades de muestrear mejor el espacio de búsqueda. Desafortunadamente algunos trabajos reportan que incrementar el tamaño de la población no asegura incrementos en la diversidad [41].
   - *Eliminación de duplicados:* Este enfoque intenta evitar que una zona sea ocupada por más de un individuo, las soluciones repetidas pueden ser eliminadas de la población. Otra opción es mutar la nueva instancia hasta que sea diferente al duplicado [4].
   - *Técnicas de infusión:* La idea central de estos mecanismos es introducir nuevos elementos aleatorios a la población, propiciando el descubrimiento de nuevos atractores ubicados en zonas no exploradas. La inserción de individuos o reinicialización de una porción de la población es efectuada periódicamente después de cierto número de generaciones. Existen múltiples técnicas de infusión como los inmigrantes aleatorios [42], el diseño y cruza ortogonal, etc. En este trabajo se hace uso de algunas técnicas de infusión que serán retomadas con mayor profundidad más adelante.
   - *Archivos externos:* Parte del problema en algoritmos evolutivos reside en la facilidad de perder soluciones rápidamente después de encontrar otro atractor. Tal situación se ha atendido a través del uso de memorias o archivos externos que almacenan individuos de generaciones pasadas. En la optimización multi-objetivo se han desarrollado varios trabajos donde los archivos son utilizados para guardar soluciones no dominadas. Otros trabajos han incluido archivos de soluciones inferiores que se usan en la selección de uno de los vectores aleatorios [43].

- *Migración de subpoblaciones:* Otra manera de prevenir la convergencia prematura es utilizar subpoblaciones que trabajan aisladamente por ciertas generaciones y posteriormente intercambian individuos con otra subpoblación. Los migrantes pueden sustituir a soluciones aleatorias, individuos similares o inferiores. La idea base es que en cada subpoblación pueden existir diferentes zonas promisorias, que pueden ayudar a mantener la diversidad al realizar intercambio de individuos [37].

2. Basadas en selección

- *Cambiar presión de selección:* Estas técnicas de promoción de la diversidad se centran en impedir que se favorezca siempre a los individuos más aptos, debido a que esto enfatiza la explotación y tiende hacia la convergencia prematura. Un ejemplo de este tipo de mecanismos es el operador de selección basada en familia propuesto por Matsui en [44], el cual dictamina que después de aplicar la recombinación el individuo más apto (del conjunto de padres e hijos involucrados) junto con el más lejano pasarán a la siguiente generación.
- *Restricciones de reemplazo:* Consiste en incluir en los operadores de reemplazo condiciones que eviten la rápida homogeneización de soluciones. Uno de los principales exponentes es el crowding, propuesto por De Jong [24], donde los nuevos individuos sustituirán a los elementos más similares dentro de un subconjunto de la población de acuerdo al factor de crowding cf.

3. Basadas en cruza o mutación

- *Restricciones para selección de padres (mating):* De manera similar a las técnicas con restricciones de reemplazo, estos mecanismos permiten a los individuos formar parejas si se cumplen ciertas condiciones; como la prevención del incesto [45], funciones de seducción [46], diferenciación suficiente [47], etc.
- *Operadores disruptivos:* Una estrategia para incrementar la diversidad es introducir grandes modificaciones a la población, a partir de técnicas como la hipermutación de Cobb; donde se incrementa temporalmente el porcentaje de mutación cuando se detecta que el desempeño en la optimización ha bajado [48].

4.3.2. Técnicas relevantes

En esta sección se describen las técnicas de promoción de la diversidad relevantes para este trabajo. La idea detrás de esta selección es escoger técnicas de

distintas clases que sean aplicables en algoritmos diferentes, por lo que se tiene una lista reducida con énfasis en los trabajos de infusión (técnicas 1-5) y nitching (técnicas 6 y 7), gracias a no depender de operadores específicos, como es el caso del crowding y otras técnicas.

1. **Inmigrantes aleatorios.** Es una técnica de inserción sencilla, consiste en introducir individuos aleatorios en la población que sustituyen a cierto porcentaje (usando un parámetro llamado *tasa de remplazo*) de los peores individuos [42].

2. **Oposición.** En el trabajo de Rahnamayan se propone la técnica de oposición, cuya idea base es la estimación del opuesto de cada punto evaluado. Los autores afirman que evaluar los puntos opuestos al tratar un problema sin conocimiento a *priori* de muchas dimensiones ayuda a acelerar la búsqueda con mayores probabilidades de encontrar individuos con mejor aptitud [51].

   En concreto, la oposición funciona de la siguiente manera: Bajo una posibilidad Jr, en cada iteración se obtiene el vector opuesto de todos los individuos en la población, dando lugar a $2NP$ soluciones. Del total de individuos se conservan los NP mejores [52]. El opuesto de un individuo se define como $\breve{x_i} = lim_{inf_i} + lim_{sup_i} - x_i$ para $X = x_1, x_2, \ldots, x_i$.

   El proceso anterior se realiza dos veces durante la ejecución del algoritmo (evolución diferencial en el caso de los autores): para inicializar la población y al final de cada generación, modificando ligeramente la fórmula para usar los individuos extremos para definir un nuevo espacio de búsqueda. Los autores nombraron a los pasos anteriores *inicialización basada en oposición* y *salto generacional basado en oposición*, respectivamente.

3. **Búsqueda local caótica.** El caos es un fenómeno en la naturaleza caracterizado por la aleatoriedad y sensibilidad a las condiciones iniciales. Gracias a estas características el caos ha sido efectivamente implementado en búsquedas locales [53], como es el caso de este mecanismo para promover la diversidad.

   Para dicho mecanismo, Jia agregó una búsqueda local caótica (también llamada *CLS* por sus siglas en inglés) a partir de la posición del mejor individuo después de cada iteración, decrementando progresivamente el espacio de búsqueda (en la *CLS*) para conservar el comportamiento esperado en los algoritmos evolutivos donde se requiere mayor exploración al inicio y movimientos más finos hacia el final del proceso de optimización.

La fórmula de la búsqueda caótica local es:

$$\vec{x_{best}}' = (1-\lambda)\vec{x_{best}} + \lambda\beta_c$$

$$\lambda = 1\left|\frac{FEs-1}{FEs}\right|^m$$

$$\beta_c = A + \beta_j^k \cdot (B-A)$$

$$\beta_j^{k+1} = \mu\beta_j^k(1-\beta_j^k), k = 2,3,\cdots;\ \beta_j \in (0,1)\ \beta_j \neq 0.25, 0.50 \text{ y } 0.75$$

Donde $\vec{x_{best}}$ es el mejor individuo después de cada generación, λ es el factor de encogimiento, FEs corresponde al número de evaluaciones hasta el momento, B_c se obtiene de la sucesión caótica β_j^{k+1}; la cual comienza con un valor aleatorio $\beta_j^1 \in (0,1)$, y del espacio de búsqueda del problema $[A,B]$.

4. **Cruza ortogonal.** La cruza ortogonal es un operador de recombinación que aplica un muestreo por diseño ortogonal para generar nuevos descendientes. Tal técnica es usada en diseño experimental cuando se desea encontrar una muestra significativa de todas las posibles combinaciones de experimentos. Llevado al terreno del cómputo evolutivo, el diseño ortogonal devuelve un número reducido de vectores que representa el espacio de búsqueda proporcionado [54]. La Figura 4.1 representa un ejemplo simplificado de diseño ortogonal, donde las posiciones de los vectores $\vec{e}$ y $\vec{g}$ definen el espacio de búsqueda, el cual es utilizado para generar un conjunto de vectores ortogonales (en este caso se produjeron 7 vectores). En la cruza ortogonal el espacio de búsqueda es proporcionado por los padres, del conjunto de vectores ortogonales obtenidos del diseño ortogonal se escoge aquél con el mejor valor de aptitud como resultado.

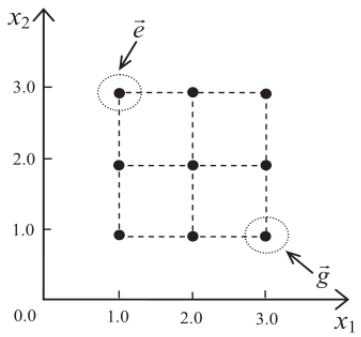

Figura 4.1: Ejemplo de diseño ortogonal en el espacio entre $\vec{e}$ y $\vec{g}$ [54].

5. **Reinicialización diente de sierra.** En esta propuesta el tamaño de la población varia constantemente a lo largo del proceso de optimización de manera periódica, describiendo un comportamiento de diente de sierra como el que se muestra en la Figura 4.2. Cada periodo el tamaño de población es reducido linealmente hasta llegar al punto de reinicio, donde el tamaño de población regresa al valor máximo y se introducen nuevos individuos aleatorios para completar la población [55].

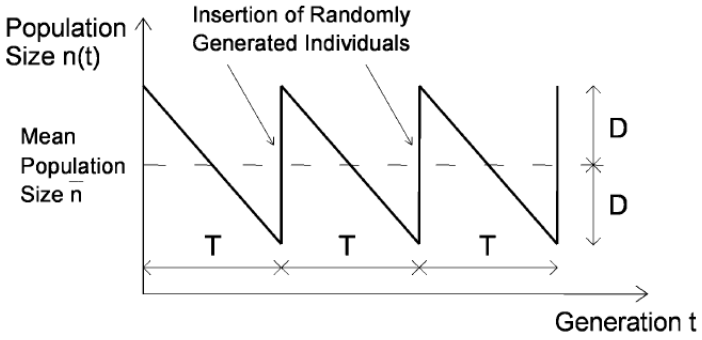

Figura 4.2: Esquema de variación de población y reinicialización [55].

Cada generación, el tamaño de población estará definido por:

$$n(t) = int\Big\{\bar{n} + D - \frac{2D}{T-1}\big[t - T\ int(\frac{t-1}{T}) - 1\big]\Big\}$$

Donde $n(t)$ es el tamaño de la población en la generación t, int es un operador de conversión de números reales a números enteros, $\bar{n}$ es el promedio del tamaño de población, D y T son los parámetros de amplitud y periodicidad, respectivamente.

6. **Fitness sharing.** El fitness sharing es la técnica clásica para nitching. Ésta modifica el paisaje de aptitud al homogeneizar los valores de los individuos ubicados en nichos, es decir las soluciones ubicadas dentro de cierto umbral de disimilitud, llamado radio de nicho σ[8].

   Cada generación debe calcularse el nuevo valor de aptitud de cada individuo de acuerdo a la siguiente fórmula:

$$f'(x_i) = \frac{f(x_i)}{\sum_{j=1}^{n} sh(d_{ij})}$$

$$sh(d) = \begin{cases} 1 - (\frac{d}{\sigma})^{\alpha} & \text{si } d < \sigma \\ 0 & \text{de otro modo} \end{cases}$$

7. **Clearing.** Es una técnica de nitching muy similar al fitness sharing, donde los recursos no son compartidos con todos los individuos que se encuentran en cada nicho. En su defecto, sólo un número reducido de individuos, llamados ganadores, pueden permanecer en un nicho. Los ganadores son los individuos más aptos que llegaron a tal zona, si se genera un individuo inferior en una zona ocupada tal individuo morirá. En caso contrario, si surge un nuevo elemento mejor que los presentes en un nicho, uno de los viejos ganadores será eliminado para dar lugar al nuevo ganador [49]. Al igual que en el fitness sharing clásico, el parámetro para delimitar los nichos es σ, mientras que κ es el parámetro que controla el número de ganadores permitidos por nicho.

Es importante destacar que la ninguna de estas técnicas tiene retroalimentación por parte de indicadores de diversidad para determinar si los mecanismos mantienen efectivamente la diversidad, simplemente se asume que la técnica mantiene la diversidad por sí misma [37].

4.4. Medidas de diversidad

4.4.1. Taxonomía

Como se comentaba anteriormente, la diversidad persigue describir las diferencias entre los individuos de la población. Tales diferencias pueden observarse desde distintos niveles.

- **Espacio genotípico:** Las medidas de diversidad genotípicas (GDMs) estudian las diferencias entre los cromosomas de la población [37] a través de la dispersión de los individuos en el espacio de búsqueda [14]. Se han propuesto numerosas medidas, cuya formulación es dependiente de la representación seleccionada. En [37] se ofrece la siguiente categorización de GDMs:

    1. *Basadas en frecuencia:* Interpretan la diversidad como la frecuencia de diferencias en la población, por ejemplo el recuento de genotipos [56], frecuencia de alelos [24], etc.
    2. *Basadas en distancia:* Utilizan alguna medida de distancia, sea distancia de Hamming para representaciones binarias, Euclidiana para codificación real, etc., para determinar la dispersión de la población en el espacio. En este trabajo se hace énfasis en este tipo de medidas al ser directamente aplicables en optimización continua con restricciones. Instancias de esta clase de medidas son distancia al punto promedio

[10], la media de la distancia entre pares [57] o el diámetro de la población [17].

3. *Basadas en entropía:* Representa cuantitativamente el nivel de desorden en la población, donde incrementos en la entropía significan incrementos en la diversidad [37].
4. *Basadas en probabilidad:* Conformado por medidas como el índice de diversidad de Simpson, inspirado en la manera de estudiar la biodiversidad en ecología, que calcula la probabilidad de que dos individuos seleccionados aleatoriamente de la población pertenezcan a la misma especie [58].
5. *Basadas en el histórico de poblaciones:* Esta clase de medida toma en cuenta generaciones pasadas, llamadas ancestros, para determinar el nivel de diversidad de la generación actual. La idea es encontrar el porcentaje de individuos de las primeras iteraciones que figuran como ancestros comunes [41].

- **Espacio fenotípico:** Las medidas de diversidad fenotípicas (PDMs) definen la distribución de aptitud de la población, analizando las diferencias en los valores de la función objetivo de las soluciones [37]. Algunas de estas medidas están formadas por combinaciones entre los valores de aptitud del mejor individuo, el peor y de la solución promedio [13].

  Medir diversidad en el espacio fenotípico requiere manejar un sólo valor por individuo, a diferencia de las GDMs las cuales tienden a ser complejas computacionalmente cuando se trata con espacios de alta dimensionalidad. Por otro lado la presencia de valles en el paisaje de aptitud puede causar estimaciones inadecuadas de la distribución de los puntos en el espacio de decisión [15].

  Al igual que para las GDMs, Črepinšek propone una clasificación de los diferentes tipos de medidas de diversidad en el espacio fenotípico [37]:

  1. *Basadas en diferencia:* Consisten en contar los diferentes fenotipos en la población, sea de manera absoluta o a través de la generación de clases donde valores de aptitud similares conforman un mismo grupo [59].
  2. *Basadas en distancia:* Al igual que en las medidas del espacio genotípico, interpretan la diversidad de acuerdo a la separación entre los individuos. En este caso, respecto a los valores de aptitud.
  3. *Basadas en entropía:* Algunos autores como Burke [60] han aplicado el concepto de entropía sobre los valores de aptitud de la población.

4. *Basadas en probabilidad:* La medida de diversidad de Simpson también ha sido aplicada como PDM [61].

Es importante destacar que al emplear una medida de diversidad se describe únicamente la dispersión de la población en el espacio que el que trabaja la medida seleccionada. Para obtener información acerca de la diversidad de la población de manera holística es deseable que la representación de las soluciones posea la propiedad de *localidad*. Dicha propiedad describe qué tan bien se corresponden los genotipos en un vecindario con sus respectivos fenotipos, la representación presentará alta localidad si existe tal correspondencia [62]. El caso ideal es tener una relación uno a uno entre la representación de las soluciones y los valores en el rango de la función objetivo.

Cuando se escoge una representación con baja localidad los movimientos en un espacio no son coherentes con los que presenta la población en el otro [63], causando que los valores reportados por una medida de diversidad sean insuficientes para comprender la dinámica de la población.

4.4.2. Medidas de diversidad estudiadas

Como puede apreciarse gracias a la clasificación de la sección anterior, se han propuesto numerosas medidas de diversidad interpretando de manera diferente la disimilitud de la población.

Debido a que es infactible incluir en la experimentación a todas las medidas de diversidad disponibles en la literatura, en este trabajo se hizo una selección de medidas de diversidad partiendo de las medidas empleadas en los estudios sobre medidas de diversidad de Corriveau et al. [14].

En la Tabla 4.1 se muestran las medidas de diversidad más relevantes para este trabajo, las cuales son únicamente medidas en el espacio genotípico, y posteriormente se describe la motivación detrás de cada una de ellas.

1. *Diámetro de la población* D_{DP}^{N}[17]: Interpreta la diversidad usando la distancia entre los dos individuos más alejados de la población y normalizando los valores con la diagonal del espacio de búsqueda (o LD, acrónimo de *landscape diagonal*, la cual se obtiene calculando la distancia entre los puntos formados por los límites inferiores y superiores del espacio de búsqueda). El no contemplar la dispersión de la población y únicamente ofrecer un valor con base en los extremos puede causar que no se describa correctamente la diversidad. Por otro lado utilizar LD para normalizar hace que la medida sea sensible a la dimensionalidad, pues las distancias se incrementan conforme aumenta el número de dimensiones [14].

2. *Distancia al punto promedio* D_{TAP}^{N2}[10]: Consiste en la media de las distancias de cada individuo de la población al vector promedio $\vec{\bar{x}}$.

Número	Medida	Fórmula
1	Diámetro de la población	$D_{DP}^{N} = \frac{1}{LD} \max_{i \neq j \in (1,2,\ldots,N)} \sqrt{\sum_{k=1}^{n} (x_{i,k} - x_{j,k})^2}$
2	Distancia al punto promedio	$D_{TAP}^{N2} = \frac{\frac{1}{N} \sum_{i=1}^{N} \sqrt{\sum_{k=1}^{n} (x_{i,k} - \bar{x}_k)^2}}{NMDF}$
3	Momento de inercia	$D_{MI}^{N} = \frac{\sum_{k=1}^{n} \sum_{i=1}^{N} (x_{i,k} - \bar{x}_k)^2}{NMDF}$
4	Diversidad verdadera	$D_{TD}^{N} = \frac{\frac{1}{n} \sqrt{\sum_{k=1}^{n} (\bar{x^2}_k - \bar{x}_k)^2}}{NMDF} \quad \bar{x^2} = \frac{1}{N} \sum_{i=1}^{N} x_{i,k}^2$
5	Media de la distancia entre pares	$D_{PW}^{N} = \frac{\frac{2}{N(N-1)} \sum_{i=2}^{N} \sum_{j=1}^{i-1} \sqrt{\sum_{k=1}^{n} (x_{i,k} - x_{j,k})^2}}{NMDF}$
6	Distancia Euclidiana al mejor individuo	$D_{ED}^{N} = \frac{\bar{d} - d_{min}}{d_{max} - d_{min}}$ $\bar{d} = \frac{1}{N} \sum_{i=1}^{N} \sqrt{\sum_{k=1}^{n} (x_{i,k} - x_{best,k})^2}$ $d_{max} = \max_{i \in (1,2,\ldots,N)} \sqrt{\sum_{k=1}^{n} (x_{i,k} - x_{best,k})^2}$ $d_{min} = \min_{i \in (1,2,\ldots,N)} \sqrt{\sum_{k=1}^{n} (x_{i,k} - x_{best,k})^2}$

Tabla 4.1: Medidas de diversidad utilizadas en el comparativo.

3. *Momento de inercia* D_{MI}^{N} [64]: Tiene el mismo principio que la distancia al punto promedio D_{TAP}^{N2}, aunque D_{MI}^{N} eleva al cuadrado las distancias haciendo más influyentes los valores atípicos.

4. *Diversidad verdadera* D_{TD}^{N} [65]: Cuantifica la diversidad a través de la desviación estándar en cada dimensión de los individuos en la población.

5. *Media de la distancia entre pares* D_{PW}^{N} [57]: Esta medida intuitiva requiere que se calcule la distancia entre todas las combinaciones de pares de individuos, gracias a esto D_{PW}^{N} es la medida más completa aunque como consecuencia el cómputo se vuelve muy pesado.

6. *Distancia euclidiana al mejor individuo* D_{ED}^{N} [13]: Como es evidente al observar su fórmula, D_{ED}^{N} difiere bastante en relación con las demás medidas presentadas ya que requiere conocer de antemano la mejor solución de la población en la generación. Nótese que la normalización se está llevando a cabo usando la diferencia de la distancias del individuo más alejado y más cercano al mejor, conforme avancen las generaciones dichas distancias decrecerán haciendo incapaz a la función de rastrear la convergencia de la población [14].

4.5. Estudios de medidas diversidad

En la sección anterior se abordó la taxonomía básica de las distintas medidas de diversidad, haciendo mención de algunas instancias de cada grupo. En cambio, este apartado introduce algunos trabajos que han intentado estudiar las medidas de diversidad y su capacidad para reflejar correctamente la diversidad de la población.

Para empezar, un punto importante respecto a las medidas de diversidad es establecer claramente cuáles son las características deseables. En [14] se describen tres cualidades que toda medida debería presentar:

1. Demostrar repetibilidad respecto a poblaciones similarmente dispersas.
2. Ser robusta a cambios en los parámetros, como el tamaño poblacional o la dimensionalidad.
3. Manejar adecuadamente los valores atípicos de los individuos de la población.

La tarea de encontrar la medida de diversidad que presente dichas características no es trivial. Hasta el momento se considera que la habilidad de las distintas medidas reportadas en la literatura especializada para describir de manera adecuada la diversidad no ha sido investigada exhaustivamente [16].

Existen pocos trabajos que se encargan de comparar las distintas medidas de diversidad bajo una metodología que permita obtener conclusiones, pues existe un sesgo causado por los parámetros y mecanismos específicos de cada algoritmo al medir diversidad, ya que se torna ambiguo conocer si el comportamiento que presentan las medidas de diversidad se debe a las particularidades del algoritmo seleccionado o a las características del problema en cuestión [66].

Algunos estudios han abordado esta situación, tal es el caso del comparativo empírico que realizan Olorunda y Engelbrecht sobre el algoritmo de optimización por cúmulos de partículas (o PSO por sus siglas en inglés) [17]. Dicho trabajo analiza el desempeño de cinco medidas de diversidad (algunas solo aplicables en PSO). El estudio analiza las tres propiedades deseables acuñadas por Corriveau et al. [14] y concluye que las medidas de diversidad basadas en distancia; específicamente la distancia al punto promedio y media de la distancia entre pares (D^N_{TAP} y D^N_{PW} respectivamente), demuestran un mejor comportamiento que el resto de medidas de diversidad evaluadas.

Otro estudio muy interesante es el de Corriveau et al. [14], ellos hicieron una recopilación de quince GDMs propuestas en el área para codificación real. Después de realizar descartes por similitudes entre las medidas se seleccionaron descriptores potencialmente adecuados e inadecuados para evaluar la diversidad.

El estudio comprende dos experimentos, el primero sobre un simulador de convergencia de su autoría. Este simulador trata de eliminar el sesgo que causan los mecanismos específicos y la elección de parámetros del algoritmo (el balance E/E del algoritmo). El segundo experimento fue realizado utilizando algoritmos del estado del arte para optimización global. En ambos experimentos se realizan pruebas respecto a repetibilidad, robustez y sensibilidad a valores atípicos. Un hallazgo importante en el trabajo de Corriveau es la incapacidad de las medidas de diversidad basadas en distancia de reproducir fielmente cuando existe convergencia multimodal, es decir; si la población está concentrada en dos o más puntos los valores que retornan las medidas son muy elevados, a diferencia de los resultados cuando existe convergencia en un solo punto (valores cercanos a 0). Lo anterior no corresponde con la apreciación intuitiva de la convergencia de los individuos a sólo un par de puntos donde se espera que las medidas de diversidad presenten un valor bajo.

Al igual que en el comparativo de Olorunda y Engelbrecht, se identifica que la distancia media entre pares es la medida que domina al resto en los objetivos antes mencionados. A pesar de ello los autores afirman que la diferencia es mínima y se requiere mayor investigación para afirmar que una medida es aplicable en cualquier tipo de búsqueda [14].

Este trabajo realiza un estudio similar al de Corriveau, evaluando la diversidad a través un conjunto de GDMs en algoritmos del estado del arte al tratar con problemas restringidos. Hasta nuestro conocimiento no existen trabajos relacionados con la diversidad de la población de algoritmos evolutivos para optimización con restricciones, de tal forma que se desconoce si las medidas de diversidad reportadas en la literatura presentaran un comportamiento similar que el reportado al trabajar con problemas sin restricciones.

Por otro lado, el segundo experimento investiga el desempeño y los comportamientos de diversidad de diferentes combinaciones de algoritmos evolutivos, manejadores de restricciones y mecanismos de promoción de la diversidad. Cabe destacar que el estudio de mecanismos de promoción de la diversidad en optimización con restricciones tampoco ha sido abordado anteriormente.

Capítulo 5

Experimento 1

5.1. Metodología

El primer experimento consiste en un comparativo empírico del desempeño de distintas medidas de diversidad aplicadas en dos algoritmos del estado del arte de distinta naturaleza al resolver problemas de optimización con restricciones. El objetivo de comparar la diversidad en algoritmos de diferentes familias es no sesgar el estudio a causa de las particularidades de los mismos, pues algunos comportamientos pueden darse gracias a los mecanismos específicos de un tipo de algoritmo.

Los algoritmos seleccionados para el estudio son CMODE [32] y SAM-PSO [30], los cuales se describen en el siguiente apartado.

5.1.1. Algoritmos del estado del arte empleados

CMODE [32] (Combining Multiobjective Optimization with Differential Evolution): Es un algoritmo basado en Evolución Diferencial cuya manera de lidiar con las restricciones es transformar el problema original en uno multi-objetivo. El nuevo problema tendrá dos objetivos a optimizar: la suma de violación de restricciones y la función objetivo.

En cada iteración se generan γ descendientes R, si el descendiente $\vec{x_i} \in R$ domina $n > 1$ individuos del conjunto Q de padres, $\vec{x_i}$ sustituirá a un padre seleccionado aleatoriamente. En caso de no producirse ninguna solución que domine al menos un individuo del conjunto Q de padres el mejor descendiente no factible $\vec{x'}$ se incluye en el archivo de soluciones no factibles A. Cada k generaciones los individuos del archivo sustituyen al mismo número de soluciones en la población, excluyendo del proceso a la mejor solución para que, en caso de que existan soluciones factibles, permanezca al menos un individuo en la zona factible encontrada.

Es importante resaltar que CMODE necesita tener soluciones no dominadas para que se aplique el reemplazo de individuos o, en su defecto, que se cumplan las k generaciones definidas para incluir soluciones no factibles. Debido a lo anterior, es muy probable que existan iteraciones donde la población no presente ningún cambio.

SAM-PSO [30] (Self-adaptive mix of particle swarm methodologies):

Este algoritmo es una extensión a la optimización por cúmulo de partículas (PSO por sus siglas en inglés) que aplica diferentes estrategias simultáneamente durante el proceso de optimización y auto-adapta los parámetros de cada individuo. SAM-PSO divide la población de acuerdo al número de estrategias seleccionadas, progresivamente las proporciones de cada sub-cúmulo cambiarán en favor de la estrategia con mejores resultados. En la implementación de los autores, al igual que en la de este experimento, se incluyen las siguientes variantes de PSO:

- *PSO local.* Trabaja de la misma manera que la variante de PSO original (PSO global) excepto que cada partícula es atraída por la mejor posición obtenida por los miembros de su vecindario local y no por la mejor posición de toda la población.
- *PSO con archivo.* La idea detrás de esta variante es conservar un archivo de soluciones pasadas para que exista un balance entre exploración y explotación de la búsqueda. Debido a que el archivo está formado por buenas soluciones e individuos cercanos al vector promedio, éste apoya tanto a la explotación de zonas prometedoras como a la variación de la población.
- *PSO sub-cúmulo* [67]. Esta estrategia divide la población en sub-cúmulos, y en cada uno de ellos la mejor partícula es identificada como el líder ese grupo. Los individuos no líderes de cada sub-cúmulo son actualizados escogiendo como g_{best} la mejor posición de una partícula aleatoria de su sub-cúmulo. Para el caso de los líderes, éstos evolucionarán usando la información de la mejor posición de otro líder elegido de manera aleatoria.

Adicionalmente, SAM-PSO añade mutación para prevenir convergencia prematura; problema importante en PSO, donde gracias a su rápida convergencia se incrementan las posibilidades de caer en óptimos locales.

5.1.2. Medidas de diversidad analizadas

En este documento se muestra un comparativo de algunas de las medidas de diversidad publicadas, en concreto se aplicaron seis medidas de diversidad en el espacio de las variables basadas en distancia (espacio genotípico): D_{DP}^{N},

D^{N2}_{TAP}, D^{N}_{TD}, D^{N}_{MI}, D^{N}_{PW} y D^{N}_{ED}. En la Tabla 4.1 se listan las medidas y sus correspondientes formulaciones.

La selección de dichas medidas de diversidad tiene una intención similar a la del trabajo de Corriveau et al. [14], contrastar las medidas de diversidad más robustas contra las que mayores dificultades presentan, aunque en este caso aplicadas a optimización con restricciones. En [14] catalogan a D^{N}_{DP} y D^{N}_{ED} como malos descriptores de diversidad mientras que D^{N2}_{TAP}, D^{N}_{TD}, D^{N}_{MI} y D^{N}_{PW} son algunas de las medidas con mejores resultados en sus experimentos sobre problemas sin restricciones.

5.1.3. Diseño experimental

Se realizaron 25 ejecuciones con un máximo de 500000 evaluaciones permitidas por ejecución, como recomienda el benchmark CEC 2006 [68], de los algoritmos CMODE y SAM-PSO solucionando los problemas del conjunto de prueba antes mencionado. En este documento se presentan los resultados correspondientes a los primeros 13 problemas. En ambos algoritmos se conservaron los parámetros reportados en la experimentación de su correspondiente publicación, a excepción del tamaño poblacional; en ambos algoritmos se asignó un tamaño de población $NP = 180$ (valor reportado en la publicación original de CMODE [32]).

En ambos algoritmos, por cada problema se tomó la ejecución con el resultado final ubicado en la mediana para graficar la diversidad con cada una de las medidas de la Tabla 4.1 (en la Sección 4.4.2). Por cuestiones de espacio, en el documento se muestran únicamente las gráficas de diversidad de las funciones más significativas.

Se calculó el valor de todas las GDMs cada $NP = 180$ evaluaciones realizadas por los algoritmos. En el caso de SAM-PSO las medidas de diversidad emplearon la posiciones actuales de las partículas, y no los mejores valores de cada individuo. La Tabla 5.1 lista los parámetros de CMODE y SAM-PSO.

También se obtuvieron valores estadísticos (véase la Tabla 5.2) para no evaluar únicamente los comportamientos de convergencia descritos por las medidas de diversidad, si no también establecer el contexto de los resultados de cada algoritmo y así relacionar las estadísticas de los resultados finales con las gráficas de diversidad para cada problema resuelto de acuerdo a los mecanismos de cada algoritmo.

5.2. Resultados y discusión

5.2.1. Desempeño de los algoritmos

En la Tabla 5.1 se aprecian las estadísticas de los resultados finales obtenidos en 25 ejecuciones por CMODE y SAM-PSO en los problemas $g01 - g13$. Se

CMODE		SAM-PSO	
NP	180	NP	180
FES	500000	FES	500000
CP	rand(0.90,0.95)	τ	[0.4,0.729]
F	rand(0.50,0.60)	c_1	[1.4,2]
γ	8	c_2	[1.4,2]
			PSO local
k	22	Variantes	*PSO con archivo*
			PSO sub-cúmulo
Estrategia ED	*rand*	NP_{arch}	$\frac{NP}{2}$
		N (archivo)	$\frac{NP}{5}$
		sub-cúmulos	5

Tabla 5.1: Parámetros de CMODE y SAM-PSO en el Experimento 1.

incluye el valor del mejor, peor, mediana, media, desviación estándar (σ^2) y tasa de factibilidad (SR) por cada uno de los problemas. Se utilizó el indicador $-$ para denotar una ejecución del algoritmo en cuestión donde no se consiguió llegar a la zona factible.

Respecto a los resultados finales de las ejecuciones de ambos algoritmos se puede observar que CMODE encuentra mejores soluciones que SAM-PSO. En cuanto a factibilidad, el primero consigue que las 25 ejecuciones lleguen a la zona factible en casi todos los problemas; la excepción son los problemas $g03$ y $g07$, donde sólo el 16 % y 80 % alcanzaron el área factible, respectivamente. En cambio, SAM-PSO únicamente logra que el 100 % de ejecuciones satisfaga las restricciones en las funciones en los problemas $g06$, $g08$ y $g12$. Aunque para problemas con un espacio de búsqueda pequeño SAM-PSO puede no tener individuos en la zona factible al finalizar la ejecución debido a la mutación, lo cual no significaría que las partículas no guardan una posición factible en su memoria.

Otro aspecto notorio en el desempeño de los algoritmos es la estabilidad que demuestra CMODE en relación a SAM-PSO, en la mayoría de las funciones del benchmark la desviación estándar (σ^2) en SAM-PSO es mucho mayor que en CMODE. Particularmente existen diferencias considerables de desviación estándar en las funciones $g01$, $g04$, $g05$, $g07$, $g08$, $g10$ y $g11$; en cambio, sólo en la función $g03$ la desviación estándar es mayor en CMODE que SAM-PSO.

En general se aprecia que CMODE es el que mejor desempeño tiene entre los dos algoritmos. El mejor resultado es superior al de SAM-PSO en los problemas $g02$, $g03$, $g05$, $g07$, $g09$ y $g10$, mientras que en el resto de problemas ambos descubren la misma solución.

Tabla 5.2: Estadísticas de 25 ejecuciones de $g01$ a $g13$ en CMODE y SAM-PSO.

$g01$	CMODE	SAM-PSO	$g02$	CMODE	SAM-PSO	$g03$	CMODE	SAM-PSO
Mejor	-15.00005	-15.00005	Mejor	-0.803623	-0.803365	Mejor	-1.0010	-1.0008511
Media	-15.00005	-14.94535	Media	-0.800686	-0.792155	Media	-0.6115	-0.999879
Mediana	-15.00005	-14.88988	Mediana	-0.803623	-0.788432	Mediana	-0.6467	-0.989127
Peor	-15.00005	-	Peor	-0.785269	-	Peor	-	-
σ^2	0.0	0.0794	σ^2	0.00539	0.0088	σ^2	0.31788	0.00297
SR	1.0	0.56	SR	1.0	0.68	SR	0.16	0.52
$g04$	CMODE	SAM-PSO	$g05$	CMODE	SAM-PSO	$g06$	CMODE	SAM-PSO
Mejor	-30665.61	-30665.61	Mejor	5126.4961	5127.39	Mejor	-6961.93	-6961.93
Media	-30665.61	-30659.78	Media	5126.4961	5173.81	Media	-6961.93	-6961.93
Mediana	-30665.61	-30659.35	Mediana	5126.4961	5211	Mediana	-6961.93	-6961.93
Peor	-30665.61	-	Peor	5126.4961	-	Peor	-6961.93	-6961.93
σ^2	0.0	13.72	σ^2	$6.289e^{-13}$	42.02	σ^2	0.0	0.0
SR	1.0	0.64	SR	1.0	0.32	SR	1.0	1.0
$g07$	CMODE	SAM-PSO	$g08$	CMODE	SAM-PSO	$g09$	CMODE	SAM-PSO
Mejor	24.306037	24.89125	Mejor	-0.095825	-0.095825	Mejor	680.62994	680.68862
Media	24.306037	26.15804	Media	-0.095825	-0.095825	Media	680.62994	680.84156
Mediana	24.306037	26.07737	Mediana	-0.095825	-0.095825	Mediana	680.62994	680.82510
Peor	-	-	Peor	-0.095825	-0.095825	Peor	680.62994	-
σ^2	$8.50e^{-15}$	1.3104	σ^2	1.11e−17	0.0	σ^2	$9.37e^{-14}$	0.2302
SR	0.88	0.84	SR	1.0	1.0	SR	1.0	0.80
$g10$	CMODE	SAM-PSO	$g11$	CMODE	SAM-PSO	$g12$	CMODE	SAM-PSO
Mejor	7048.7269	7160.465	Mejor	0.7498	0.74983	Mejor	-1.0	-1.0
Media	7048.7269	7656.016	Media	0.7498	0.74988	Media	-1.0	-1.0
Mediana	7048.7269	7888.102	Mediana	0.7498	0.74993	Mediana	-1.0	-1.0
Peor	7048.7269	-	Peor	0.7498	-	Peor	-1.0	-1.0
σ^2	$9.67e^{-8}$	539.10	σ^2	$1.11e^{-16}$	$5.02e^{-5}$	σ^2	0.0	$1.1537e^{-16}$
SR	1.0	0.60	SR	1.0	0.80	SR	1.0	1.0

$g13$	CMODE	SAM-PSO
Mejor	0.05393	0.48163
Media	0.36180	0.85308
Mediana	0.43877	0.99823
Peor	0.43877	-
σ^2	0.1539	0.16757
SR	1.0	40 %

5.2.2. Resultados del comparativo de diversidad

En las Figuras 5.1, 5.2 y 5.3 se presentan los resultados de diversidad de la mediana de las ejecuciones de los algoritmos CMODE y SAM-PSO. Se tienen dos gráficos por función, correspondientes a cada uno de los algoritmos, exhibiendo las curvas de diversidad producidas por las seis GDMs en los dos algoritmos. A continuación se discuten los resultados, resumiendo los hallazgos por medida de diversidad evaluada.

Diámetro de la población. D_{DP}^{N}, representada en las gráficas con el color azul oscuro, utiliza la distancia entre los dos individuos más separados de la población y usa como factor de normalización la diagonal del espacio de búsqueda. Debido a lo anterior se puede prever en cierto sentido el comportamiento, pues mientras el algoritmo se encuentre en periodo de exploración es probable que la distancia más larga entre dos soluciones oscile fuertemente y por lo tanto su curva de diversidad se vea escarpada. Tal es el caso al resolver las funciones $g02$ y de $g03$

con CMODE o $g02$, $g03$, $g11$ y $g13$ para SAM-PSO. Este problema no se ve reflejado en las gráficas donde la convergencia se da tan rápido que los vectores más alejados están prácticamente en el foco de convergencia desde las primeras iteraciones.

Existe una situación particular con cada uno de los algoritmos al aplicar esta medida de diversidad: SAM-PSO incluye mutación, lo que propicia la exploración pero también podría hacer que la distancia entre los individuos más separados crezca. En CMODE sucede algo similar gracias a la introducción de soluciones no factibles del archivo cuando el algoritmo no es capaz de encontrar soluciones no dominadas, fenómeno visible en las funciones $g08$, $g09$ y $g11$; cada vez que individuos del archivo se incluyen en la población se observa un salto muy grande de diversidad.

Distancia al punto promedio, momento de inercia, diversidad verdadera y media de la distancia entre pares. Este conjunto de GDMs se muestra en las gráficas con los colores verde, rojo, azul claro y morado. Como se comentó en secciones anteriores, las cuatro medidas toman en cuenta a todos los individuos para describir la diversidad de la población y todas normalizan los resultados con la diversidad calculada en la primera generación, la mejor forma de normalizar de acuerdo al estudio de Corriveau et al. [14].

A pesar de tener diferencias en sus formulaciones, D^{N2}_{TAP}, D^{N}_{MI}, D^{N}_{TD} y D^{N}_{PW} trazan curvas de diversidad muy similares e incluso en algunos problemas el comportamiento es casi idéntico; tal es el caso en casi todas las funciones evaluadas. Existen algunos casos donde, aunque el comportamiento es similar, las gráficas difieren un poco: para CMODE esto sucede con las funciones $g02$ y $g08$, donde claramente se aprecia que D^{N}_{MI} y D^{N}_{TD} se encuentran alineadas pero separadas de D^{N2}_{TAP} y D^{N}_{PW}, que a su vez delinean los mismos valores de diversidad. Una situación distinta a las anteriores ocurre en la función $g03$, problema con un espacio de búsqueda pequeño ($[0,1]^{10}$); todas las medidas evaluadas indican estancamiento en la población aunque cada una sugiere una dispersión diferente en la población. D^{N}_{TD} y D^{N}_{MI} tienen los valores más bajos, lo cual es ocasionado por el tamaño del espacio de búsqueda, ya que D^{N}_{MI} eleva al cuadrado las distancias mientras que D^{N}_{TD} lo hace con la diferencia de la media de cada dimensión al alelo correspondiente del punto promedio. Debido a que la distancia más larga que puede existir en la población es LD, cuyo valor es de 3.1622, al elevar los valores a la segunda potencia estos se homogeneizan, resultando en valores de diversidad más bajos. Para SAM-PSO también se obtuvieron gráficas de diversidad que exhiben los comportamientos antes mencionados: en la gran mayoría de los problemas las cuatro medidas dibujan sus curvas de diversidad con casi la misma trayectoria, $g02$, $g03$ y $g13$ guardan el mismo comportamiento con una separación mínima entre las curvas. Una situación especial es el resultado del problema $g11$, cuyo espacio de búsqueda $[-1,1]^2$ resulta tan pequeño que la mutación de SAM-PSO

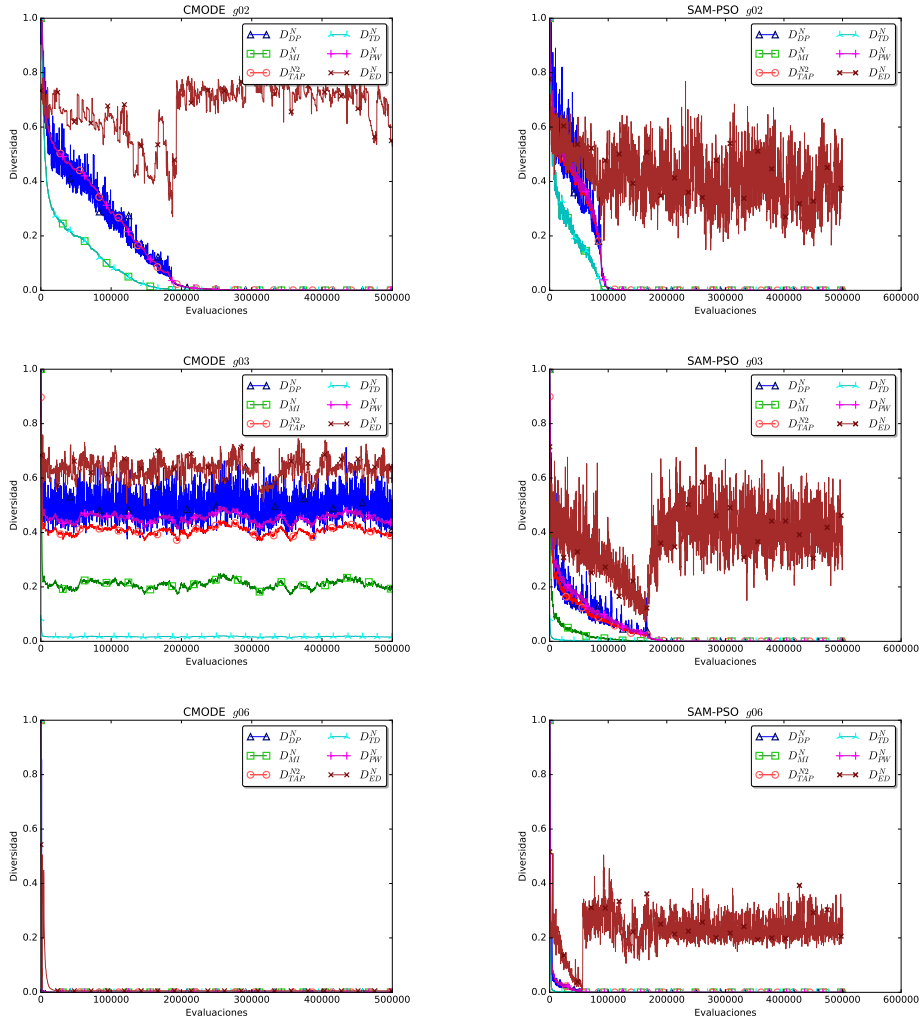

Figura 5.1: Resultados para el problema $g02$, $g03$ y $g06$ con CMODE y SAM-PSO.

hace que la población nunca converja, sin que esto signifique que no se optimizó la función pues las partículas pueden tener almacenada en la memoria una buena solución. En cuanto a D^{N2}_{TAP} y D^{N}_{PW}, permanecen cercanas en todas las funciones de prueba de ambos algoritmos. Teóricamente D^{N}_{PW} tendría que ofrecer la mejor descripción de diversidad entre las cuatro ya que contempla la distancia entre todos los individuos, aunque el costo computacional es mucho más elevado que el resto de medidas.

Distancia Euclidiana al mejor individuo. La última medida considerada, trazada en color café , tiene muchos problemas. Al observar las curvas de diversidad

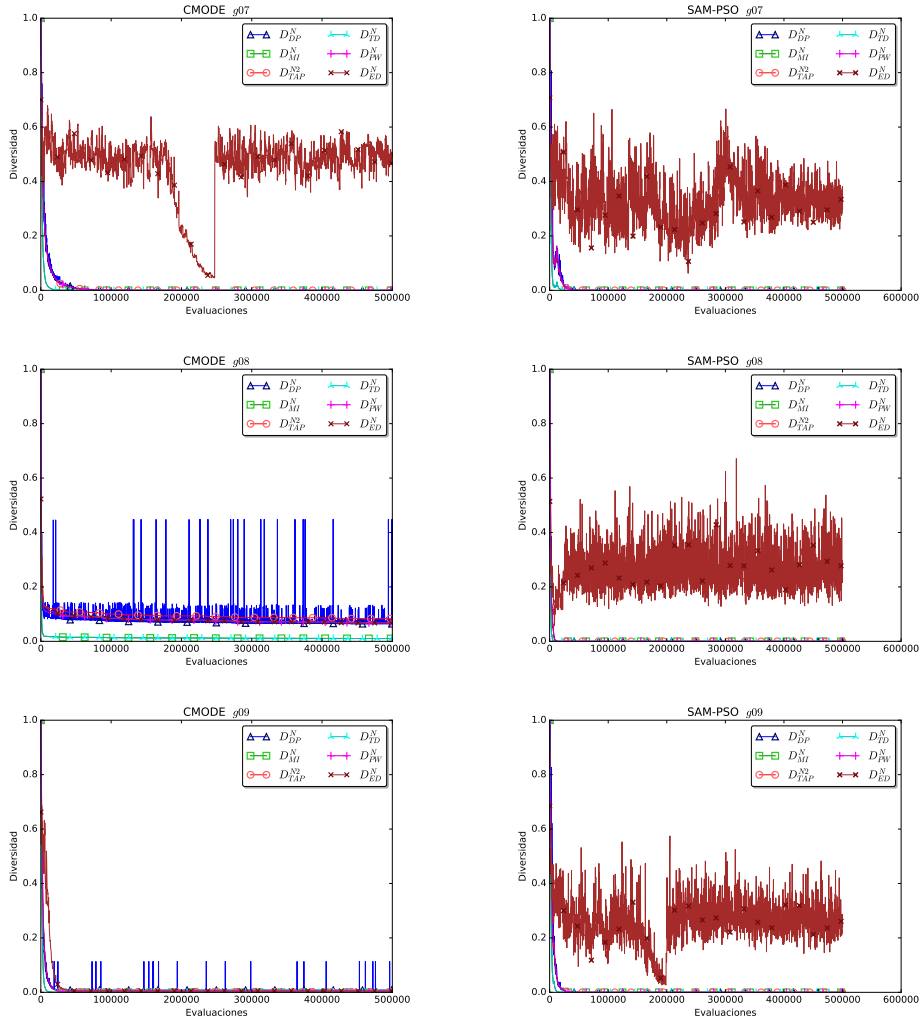

Figura 5.2: Resultados para el problema $g07$, $g08$ y $g09$ con CMODE y SAM-PSO.

que D_{ED}^{N} dibuja por cada una de las funciones éstas resultan poco informativas. Únicamente si la convergencia se da muy pronto es posible que D_{ED}^{N} describa la diversidad correctamente, por ejemplo en las funciones $g06$, $g08$, $g09$ y $g11$ con CMODE. El principal problema reside en que la normalización es incorrecta; si el algoritmo converge, la distancia promedio al mejor vector y el factor de normalización $(d_{max} - d_{min})$ decrecerán, causando una curva de diversidad que se mantiene en el mismo intervalo de diversidad sin demostrar la concentración de individuos que presenta la población. En caso de estancamiento o exploración sin convergencia, las distancias al mejor vector cambiarán muy poco, también se

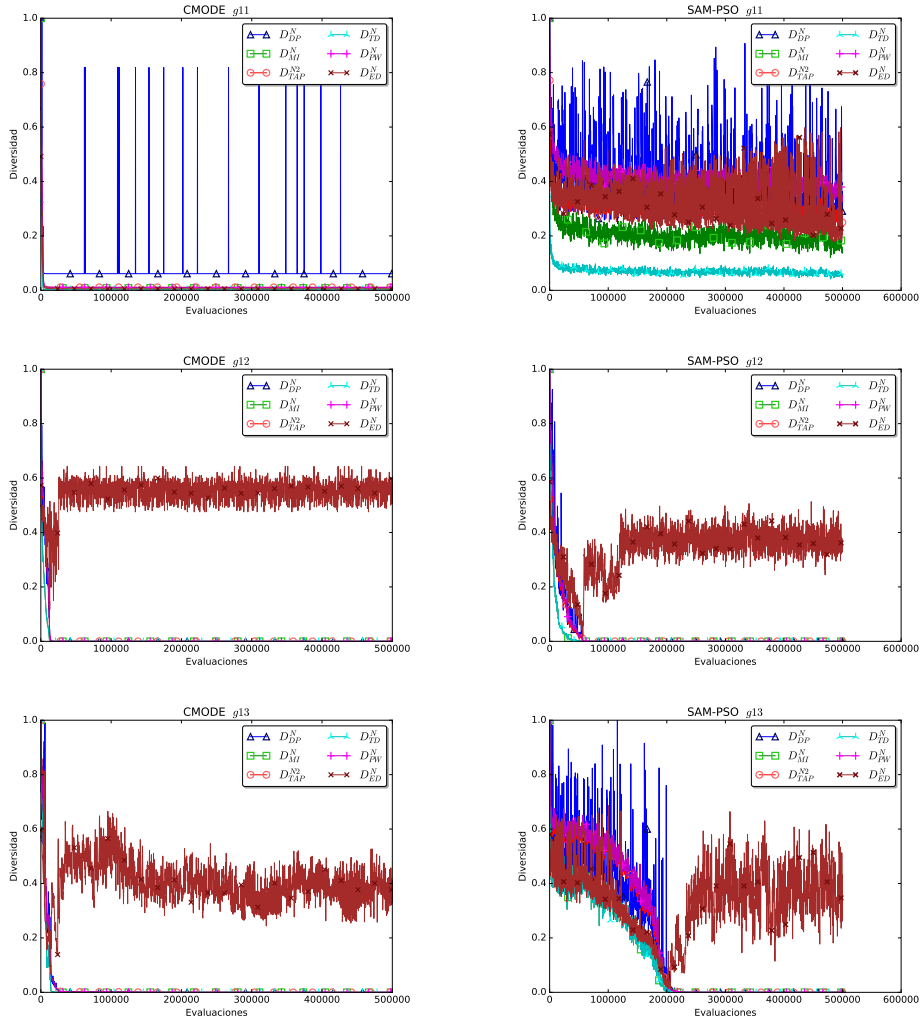

Figura 5.3: Resultados para el problema $g11$, $g12$ y $g13$ con CMODE y SAM-PSO.

obtendrá una trayectoria escarpada y atrapada en valores cercanos de diversidad.

5.3. Conclusiones

En este experimento se presentó un comparativo empírico del desempeño de seis medidas de diversidad aplicadas a dos algoritmos del estado del arte, CMODE [32] y SAM-PSO [30], al resolver las funciones de prueba del benchmark CEC2006, estudio necesario ya que únicamente se han publicado comparativos de medidas de diversidad para problemas sin restricciones.

En primera instancia se aprecia la superioridad de CMODE sobre SAM-PSO. A pesar de que conocer cuál algoritmo tiene mejor desempeño no es el punto central de este experimento permite contextualizar de mejor manera las curvas de diversidad.

Los resultados respecto a diversidad permiten observar, al igual que en el trabajo de Corriveau [14], que también en optimización con restricciones las medidas de diversidad que contemplan a toda la población (D_{MI}^{N}, D_{TD}^{N}, D_{TAP}^{N2} y D_{PW}^{N}) dibujan curvas que corresponden a los movimientos esperados de la población (a partir de los mecanismos de los algoritmos y los resultados finales) en ambos algoritmos y todas las funciones de prueba evaluadas.

D_{ED}^{N} y D_{DP}^{N} son malos descriptores, pero a pesar de que D_{DP}^{N} no calcula correctamente la diversidad podría ser útil para rastrear mecanismos de promoción de diversidad, como sucede en CMODE cuando se introducen soluciones no factibles y el gráfico de diversidad lo refleja con cambios abruptos.

Destacan como las mejores medidas D_{TAP}^{N2} y D_{PW}^{N}, al proporcionar mayor sensibilidad cuando se presenta estancamiento que el resto de medidas; D_{MI}^{N}y D_{TD}^{N} consiguen representar el estancamiento con valores más cercanos a 0, lo que significa menor reflejo de la dispersión de la población. En algunos escenarios el costo computacional es un problema, cuando éste es el caso D_{TAP}^{N} podría ser la mejor opción pues sólo calcula NP distancias, a diferencia de D_{PW}^{N}, la cual computa todas los pares de distancias posibles.

Finalmente se debe reconocer que un comparativo de este estilo no puede demostrar si las medidas de diversidad reflejan fielmente la diversidad en todos los algoritmos de optimización con restricciones, ya que es necesario medir diversidad en escenarios donde se tenga un comportamiento esperado antes de evaluar el desempeño de las medidas [14]. Al medir diversidad en algoritmos, el comportamiento está sesgado por las estrategias y mecanismos que ellos implementan.

Por este motivo el segundo experimento se enfocó en investigar la influencia del manejador de restricciones, las particularidades de algoritmos básicos con operadores ya estudiados, además de la inclusión de técnicas de promoción de la diversidad como elemento de cambio, pues todos estos elementos modifican la dinámica de la población durante la optimización.

Capítulo 6

Experimento 2

6.1. Metodología

El segundo experimento que se presenta en este documento es un comparativo empírico, con algunas modificaciones al descrito en la sección anterior. Al realizar el primer experimento se llegó a la cuenta de que es complicado establecer las dinámicas de la población respecto a la diversidad partiendo de algoritmos complejos. Tal es el caso de CMODE [32] y SAM-PSO [30], pues ambos algoritmos incluyen mecanismos sofisticados que les permiten alcanzar buenos resultados. Debido a esta situación en este experimento se trabajó con algoritmos de optimización básicos, con la finalidad de evitar elementos no presentes en la versión original que compliquen el análisis.

6.1.1. Algoritmos

Se seleccionaron dos algoritmos básicos de diferente naturaleza para este experimento: Evolución diferencial (en adelante abreviado *DE*) [31] en su versión original ($DE/rand/1/bin$) y una implementación del algoritmo genético (referido como *GA*) con los operadores de selección por torneo binario, cruza binaria simulada (SBX [69, 70]) y mutación polinomial (PM [70]).

Ambos algoritmos fueron introducidos en el Capítulo 3, en el caso del algoritmo genético únicamente se mencionó la versión original, por este motivo se describen brevemente los operadores que se incluyeron en la optimización elaborada para la experimentación. Para mayor profundidad se invita al lector a dirigirse a las publicaciones correspondientes.

Cruza binaria simulada (SBX)[69, 70]

La idea detrás de SBX es transportar el operador de cruza clásico binario del algoritmo genético a representación real. Lo anterior se realiza utilizando una distribución de probabilidad para determinar el factor de dispersión β, cuyo

valor indicará si los nuevos descendientes son una copia de los padres, cuando $\beta = 1$, o aparecerán alejados de éstos, con valores de β menores o mayores a 1. El valor de β depende del parámetro η_c, llamado índice de distribución, el cual recibe enteros positivos; cuanto mayor sea su valor más cercanos se encontrarán los hijos producidos de la cruza SBX.

Mutación polinomial (PM)[70]

El caso de la mutación polinomial es parecido a SBX, pues también se apoya en una distribución de probabilidad para inducir perturbaciones en las soluciones. Usando el parámetro η_m, el cual indica la máxima perturbación permitida dentro del rango definido $[a, b]$, se obtiene el factor de perturbación $\delta = \frac{(a,b)}{\eta_m}$. A partir de un numero aleatorio uniforme u se genera un nuevo factor de perturbación $\bar{\delta}$, cuyo producto con μ_m modificará a la solución.

6.1.2. Manejadores de restricciones

Otro factor primordial en el desempeño de los algoritmos al lidiar con problemas con restricciones es la técnica de manejo de restricciones que el algoritmo implementa. Existen varias estrategias para tratar con espacios restringidos, en este trabajo se enfatizan tres técnicas prominentes de la literatura especializada: las *reglas de factibilidad de Deb (deb)* [19], ϵ *constrained method (*ϵ*cm)* [34] y *Stochastic ranking (sr)* [35]. Los tres manejadores antes mencionados fueron descritos en el Capítulo 3, correspondiente a cómputo evolutivo.

Cabe destacar que el manejador de restricciones se utilizó como método de comparación no sólo para el reemplazo, si no para todas las comparaciones en el algoritmo y las técnicas de promoción de la diversidad.

6.1.3. Técnicas de promoción de la diversidad

Además de la muy importante relación entre el balance E/E de un algoritmo y el manejador de restricciones utilizado existen otros elementos que pueden modificar la diversidad de la población a lo largo del proceso evolutivo. Entre dichos elementos se encuentran las técnicas de promoción de la diversidad como el nitching o los métodos de infusión, las cuales han sido usadas exitosamente para optimización global, problemas dinámicos y optimización multi-objetivo.

Cuando de problemas restringidos se trata, existen muy pocos trabajos que incluyan alguno de estos mecanismos para promover la diversidad y mejorar el desempeño. La parte interesante resulta en conocer si no existen tales trabajos porque las técnicas de promoción no consiguen obtener *diversidad útil* (aquella que viene acompañada de buenos resultados finales), o si simplemente no se ha explorado la posibilidad. En este sentido, el añadir técnicas de promoción de la diversidad tiene dos objetivos: extender el análisis a estrategias que modifican la

diversidad de una manera en específico, haciendo el experimento más rico; y observar el rendimiento de las distintas combinaciones de algoritmos, manejadores de diversidad y técnicas de promoción de la diversidad en busca de una configuración con un buen compromiso exploración/explotación que presente buenos resultados.

Se seleccionaron siete técnicas de promoción de la diversidad, descritas en el Capítulo 4, considerando si eran compatibles con los algoritmos incluidos en el comparativo. A continuación se enumeran dichas técnicas y se incluyen comentarios acerca de las condiciones de implementación para cada algoritmo:

1. Inmigrantes aleatorios (*RI*)[42]: Esta técnica sustituye un porcentaje de la población por un nuevo conjunto de individuos generados de manera aleatoria en el espacio de búsqueda al final de cada generación, no requirió de ajustes para ningún algoritmo.

2. Oposición (*OPP*)[52]: Gracias a que tanto la *inicialización y el salto generacional basado en oposición* trabajan sobre toda la población fuera del ciclo principal de los algoritmos no se necesitó ajuste alguno.

3. Búsqueda local caótica (*CLS*)[53]: Debido a que sólo se aplica sobre el mejor individuo de cada generación, esta técnica es de fácil aplicación.

4. Cruza ortogonal (*ORT*)[54]: A pesar de aplicar de distinta manera el operador de cruza por parte de DE y GA, la cruza ortogonal simplemente debe ajustarse en el cálculo de k. De todas las cruzas que se realizan durante una iteración k, será aquella donde, en vez de utilizar la cruza binomial en DE o la cruza binaria simulada con GA, se emplea la cruza ortogonal. En el caso de Evolución diferencial k es un número aleatorio entre 1 y el tamaño de la población, pues se realiza una cruza entre cada individuo (target) y el vector mutante (trial). En el algoritmo genético se aplican $N/2$ cruzas, ya que se generan $N/2$ parejas de individuos para aplicar recombinación.

5. Reinicialización diente de sierra (*SAW*)[55]: Este método únicamente modifica el tamaño poblacional e incluye nuevos individuos aleatorios. Entra en funcionamiento al final de cada iteración y no requirió modificaciones, se usó el algoritmo original en DE y GA.

6. Clearing (*CLE*)[49]: Esta técnica pertenece al grupo de nitching y requiere comparar a cada individuo con el resto para formar nichos con cupo limitado, donde sólo los mejores en cada zona permanecerán y el resto será remplazado por soluciones aleatorias. En ambos algoritmos Clearing se aplicó al finalizar cada generación. Otro factor importante que tuvo que ser añadido para poder implementar técnicas de nitching fue el trato de la

suma de violación de restricciones $cvs(x)$, pues este valor es un elemento que no se considera en el algoritmo original de clearing ni fitness sharing. La modificación consiste simplemente en aplicar la misma fórmula para $f(x)$ con los valores de $cvs(x)$ (la misma modificación se agregó para fitness sharing).

7. Fitness Sharing (FS)[9]: En fitness sharing, al igual que para Clearing, se necesitó usar un mecanismo para determinar el valor de σ, también conocido como radio del nicho, el cual es un parámetro dependiente del problema. Debido a la naturaleza de los problemas de prueba se optó por otorgar a σ el valor del 10 % de la diagonal del espacio de búsqueda (definida en el Capítulo 4), de manera similar al esquema utilizado en [71] al asignar valores proporcionales al tamaño del problema.

6.1.4. Medidas de diversidad

Gracias a los hallazgos de Corriveau et al. [14] y del experimento con los algoritmos del estado del arte, se decidió medir la diversidad en las ejecuciones con las medidas que mejor reflejan la diversidad entre las evaluadas: la distancia al punto promedio D^{N2}_{TAP} y la media de la distancia entre pares D^{N}_{PW}. En la sección de resultados se presentan únicamente las gráficas de diversidad utilizando D^{N}_{PW}.

6.1.5. Funciones de prueba

La pieza restante para el experimento resulta en el conjunto de problemas a resolver, en este caso se optó por los problemas del benchmark del CEC 2010 [72]. Estos problemas presentan la particularidad de someterse a rotaciones, lo que vuelve más complicado llegar al mejor valor conocido, referido como óptimo (en aras de simplicidad) a lo largo de este capítulo.

El detalle de las características de las funciones de prueba del benchmark CEC 2010 se puede encontrar en el Apéndice A.

Los problemas del conjunto de prueba del benchmark CEC 2010 se presentan en 10 y 30 dimensiones. Bien sabido es que la complejidad de los problemas se incrementa conforme se añaden dimensiones (fenómeno conocido como la maldición de la dimensionalidad). Al incrementarse la dificultad, los algoritmos pueden encontrar el óptimo en 10 dimensiones y fracasar al resolver los problemas en 30D. En este experimento se considera la dimensionalidad como otro elemento que influye en la diversidad de la población.

6.1.6. Diseño experimental

Este experimento plantea un comparativo empírico entre elementos involucrados en la dinámica de la población al resolver problemas de optimización con

restricciones: algoritmos, manejadores de restricciones y técnicas de promoción de la diversidad.

Concretamente se contrasta el desempeño y la diversidad de las combinaciones resultantes de los algoritmos *DE* y *GA*; con cada uno de los tres manejadores de restricciones *deb*, ϵcm y *sr*; así como las 7 técnicas de promoción de la diversidad *RI*, *OPP*, *CLS*, *ORT*, *SAW*, *CLE* y *FS*; más una versión de cada algotimo sin aplicar técnicas de promoción de la diversidad, denotado con la etiqueta *SIN*. Resultando en un total de 48 combinaciones, véase la Figura 6.1.

En adelante se utilizará la nomenclatura siguiente para hacer referencia a una combinación en específico:

<algoritmo>__<manejador>__<técnica de diversidad>

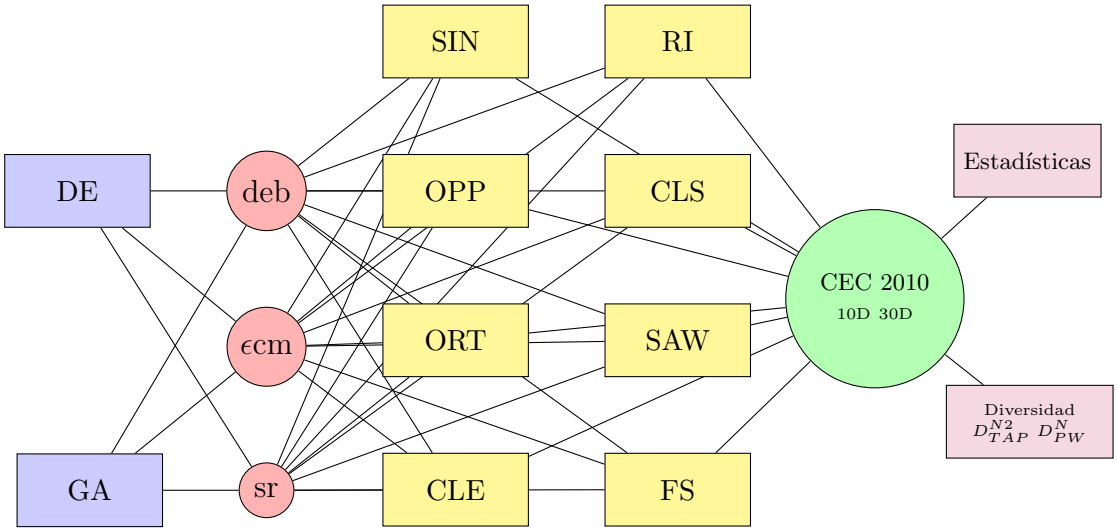

Figura 6.1: Esquema general del Experimento 2.

Se realizaron 25 ejecuciones independientes, como recomienda el reporte técnico del CEC 2010, para cada función de prueba. Para efectos de simplicidad se aplicaron los valores de parámetros sugeridos en las correspondientes publicaciones, tanto para los manejadores de restricciones como para los mecanismos de diversidad y parámetros de los algoritmos. En la Tabla 6.1 se muestra la configuración de parámetros utilizada.

De la muestra de ejecuciones independientes se generaron datos estadísticos y gráficas de diversidad, agrupadas por problema y manejador de restricciones. De tal suerte que en cada gráfico se muestran ocho curvas de diversidad referentes a la ejecución ubicada en la mediana de cada técnica de promoción de diversidad estudiada, incluyendo a la variante carente de técnica de promoción.

En el apartado de resultados y discusión solo se muestran resultados repre-

sentativos, para observar el total de datos estadísticos y gráficas de diversidad puede dirigirse al Apéndice B.2.

Debido a las dimensiones del experimento es necesario emplear un método de comparación que permita conocer si existen diferencias significativas entre las variantes y cuáles de ellas son las que tienen mejor comportamiento. El alto número de combinaciones a evaluar hace inviable utilizar pruebas de hipótesis para múltiples comparaciones, debido a que el tamaño de la muestra es insuficiente, es decir, existen más algoritmos que funciones de prueba en el diseño del experimento. Por este motivo, se aplicó la prueba de hipótesis de suma de rangos de Wilcoxon en parejas de las combinaciones con mejores resultados usando un valor de significancia del 5 %.

Para discriminar variantes y encontrar cuales de ellas presentaron los mejores resultados se calcularon los valores de las métricas tasa de factibilidad (FR) y tasa de éxito (SR), ambas presentadas en el reporte técnico del benchmark del CEC 2006 [68]. La tasa de factibilidad se obtiene al dividir el número de ejecuciones donde al menos un individuo de la población final se encuentra en la zona factible entre el total de ejecuciones. De modo similar, la tasa de éxito es la razón entre el número de ejecuciones que llegaron al óptimo de la función y el total de ejecuciones. En [68] no se encuentra información acerca del óptimo de cada función, valor necesario para calcular la tasa de éxito. Por este motivo se tomaron los mejores valores obtenidos por algoritmos del estado del arte reportados por Elsayed et al [73].

6.2. Resultados

6.2.1. Resultados finales

En las Figuras 6.2, 6.3, 6.4 y 6.5 pueden apreciarse los diagramas de cajas correspondientes a la tasa de factibilidad y éxito para cada variante en los 18 problemas del benchmark CEC2010 en 10 y 30 dimensiones. Éstos nos permiten visualizar la distribución de los valores obtenidos en ambas métricas por cada combinación al resolver las funciones de prueba.

Al observar el diagrama de cajas de la tasa de factibilidad en 10D (Figura 6.2), es visible que las combinaciones $DE\_\epsilon cm\_SIN$, $DE\_\epsilon cm\_OPP$, $DE\_\epsilon cm\_CLS$ y $DE\_\epsilon cm\_ORT$ presentan la mayor tendencia hacia la factibilidad en comparación con el resto de variantes. El primer cuartil de dichas combinaciones se encuentra alrededor de 0.8, lo que indica que al menos en el 75 % de las funciones se obtienen valores de FR iguales o mayores que 0.8.

De manera muy contrastada la Figura 6.3, correspondiente a la tasa de factibilidad en 30D, no permite ver una tendencia tan clara hacia la factibilidad como en 10D. Únicamente las variantes $DE\_\epsilon cm\_CLS$ y $DE\_\epsilon cm\_ORT$ consiguen des-

Algoritmo	Parámetro	Valor
Generales	NP	100
	FES	500000
DE	F	0.5
	CR	0.9
GA	CP	0.9
	MP	0.1
	η_c [70]	1
	η_m [70]	100
ϵ constrained method	cp [34]	0.5
	Tc [34]	$0.2 * FES/NP$
Stochastic ranking	P_f [35]	0.45
RI	$\%RI$ [42]	0.3
OPP	Jr [52]	0.3
CLS	m [53]	1500
ORT	Q [54]	3
	K [54]	4
SAW	T [55]	40
	D [55]	75
CLE, FS	σ	$0.1 * LD$
	α [8]	1

Tabla 6.1: Parámetros experimento 2.

tacar al elevar el primer cuartil a valores de FR mayores de 0. A pesar de ello es notorio que las variantes de DE consiguen valores más altos de factibilidad al tener una mediana más elevada en ambas dimensiones, situación acentuada en 30D.

Las Figura 6.4 y 6.5 muestran el diagrama de cajas de la tasa de éxito, métrica que sensa la frecuencia en la que un algoritmo consigue encontrar el óptimo en un problema de optimización (en este caso el mejor valor conocido de cada problema). Claramente se obtienen mucho mejores resultados en 10 que en 30 dimensiones, donde la mayoría de variantes de DE presenta cajas con volumen a diferencia del diagrama de 30D donde al carecer de éstas indican que se fracasó en encontrar el óptimo al menos en el 75 % de los problemas.

Sin embargo es posible reconocer que las combinaciones identificadas a partir de la tasa de factibilidad, $DE\_\epsilon cm\_SIN$, $DE\_\epsilon cm\_OPP$, $DE\_\epsilon cm\_CLS$ y $DE\_\epsilon cm\_ORT$, consiguen mejores resultados que el resto de variantes al tener un valor de SR en la mediana muy superior a las demás.

En general se observan resultados muy pobres, donde sólo las variantes $DE\_\epsilon cm\_SIN$, $DE\_\epsilon cm\_OPP$, $DE\_\epsilon cm\_CLS$ y $DE\_\epsilon cm\_ORT$ consiguen resolver un porcentaje considerable de los problemas de prueba en ambas dimensiones, mientras que el resto fracasa en la gran mayoría de problemas. También es notorio que las variantes de DE en general presentan mejores resultados finales que los arrojados por las combinaciones procedentes del algoritmo genético.

Otro factor que destaca en los resultados es la superioridad de las variantes que emplean ϵ constrained method. En el caso de DE se obtuvieron las mejores combinaciones usando este manejador e incluso en las variantes de GA se observan mejores valores usando ϵcm, donde dichas combinaciones lograron alcanzar el óptimo en algunas funciones en 10D en mayor medida que GA con el resto de manejadores.

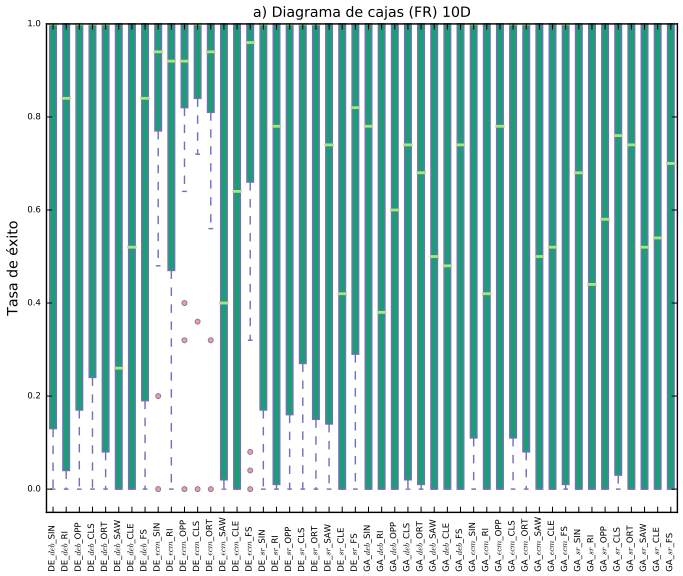

Figura 6.2: Diagrama de cajas de la tasa factibilidad (FR) en 10D.

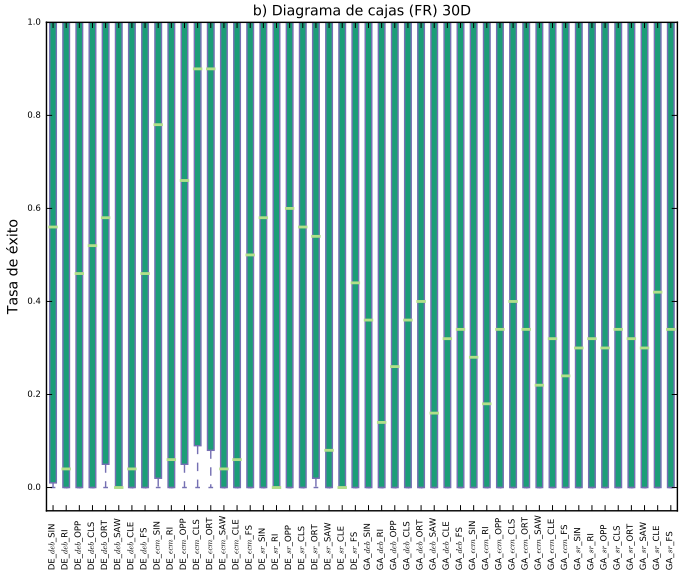

Figura 6.3: Diagrama de cajas de la tasa factibilidad (FR) en 30D.

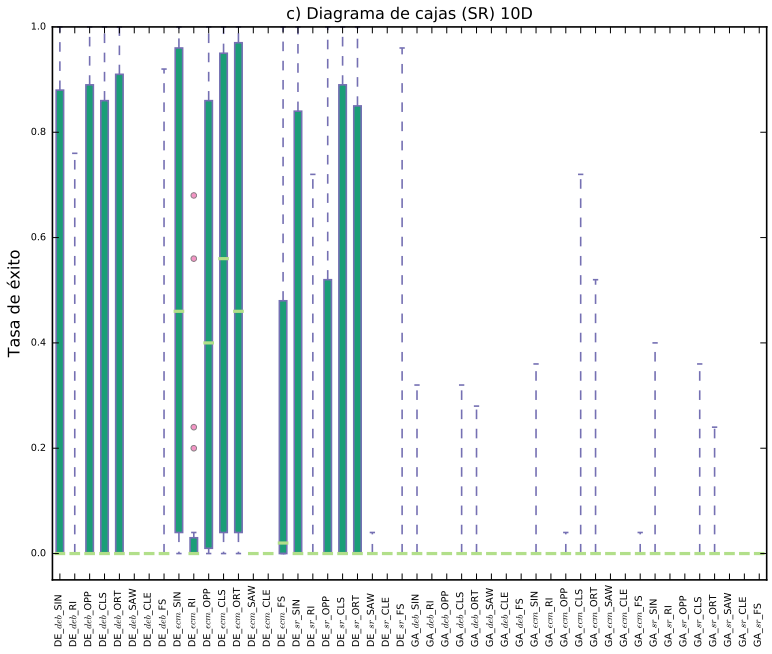

Figura 6.4: Diagrama de cajas de la tasa éxito(SR) en 10D.

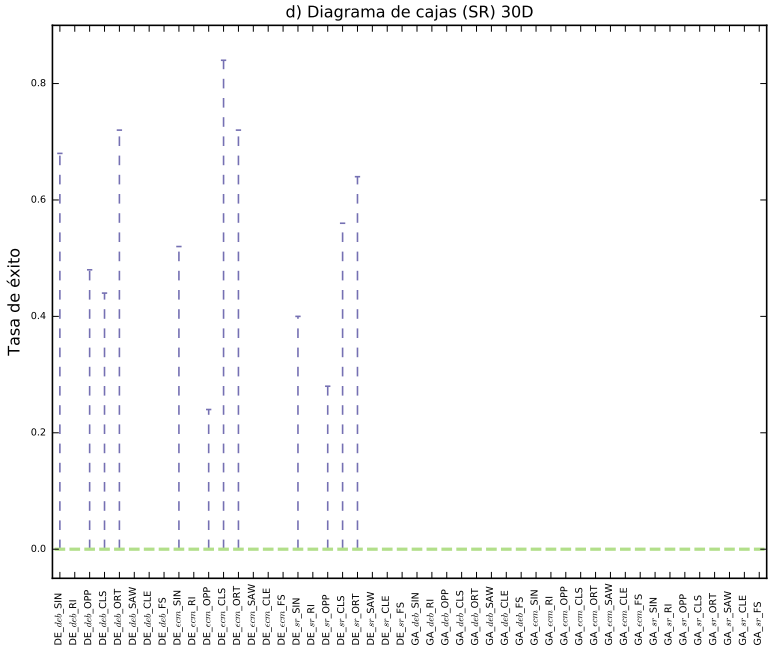

Figura 6.5: Diagrama de cajas de la tasa de éxito (SR) en 30D.

A partir de las observaciones anteriores, es evidente la superioridad de $DE\_\epsilon cm\_SIN$, $DE\_\epsilon cm\_OPP$, $DE\_\epsilon cm\_CLS$ y $DE\_\epsilon cm\_ORT$ sobre las otras 44 variantes, de tal suerte que las pruebas de hipótesis sólo consideran estas cuatro variantes. La Tabla 6.2 presenta el resume los resultados obtenidos al aplicar la prueba de Wilcoxon sobre los resultados finales de las variantes más destacadas en 10 y 30 dimensiones. Donde la primera columna indica la hipótesis a probar, la segunda columna indica los resultados en los problemas de 10 dimensiones y la tercera muestra los correspondientes a 30 dimensiones, en ambos casos se indica la cantidad de diferencias significativas organizadas por victorias, derrotas y empates respecto al primer algoritmo.

Al comparar $DE\_\epsilon cm\_CLS$ contra la variante $DE\_\epsilon cm\_ORT$, la prueba de hipótesis encontró diferencias significativas en los resultados en ambas dimensiones, donde $DE\_\epsilon cm\_CLS$ sólo aventaja a $DE\_\epsilon cm\_ORT$ en una función en 10D y presenta el mismo número de victorias y derrotas.

Dicho lo anterior, los resultados demuestran que las variantes $DE\_\epsilon cm\_CLS$ y $DE\_\epsilon cm\_ORT$ tuvieron el desempeño más alto entre las 48 combinaciones de algoritmo, manejador de restricciones y técnica de promoción de la diversidad estudiadas.

Hipótesis	10D			30D		
	+	−	=	+	−	=
$DE\_\epsilon cm\_CLS$ vs $DE\_\epsilon cm\_SIN$	1	0	17	4	0	14
vs $DE\_\epsilon cm\_OPP$	6	1	11	8	0	10
vs $DE\_\epsilon cm\_ORT$	1	0	17	4	4	10

Tabla 6.2: Resumen de la prueba de Wilcoxon entre las variantes con mejores resultados en 10 y 30D.

		CLS	SIN	OPP	ORT			CLS	SIN	OPP	ORT
c01	FR	1.0	1.0	0.92	0.96	c10	FR	0.72	0.76	0.8	0.64
	Mejor	-7.4731E-01	-7.4731E-01	-7.4731E-01	-7.4731E-01		Mejor	4.1726E+01	0.0000E+00	4.1726E+01	1.3326E+01
	Mediana	-7.4731E-01	-7.4731E-01	-7.4731E-01	-7.4731E-01		Mediana	4.1726E+01	4.1726E+01	4.2284E+01	4.1726E+01
	Media	-7.4270E-01	-7.4328E-01	-7.3997E-01	-7.3936E-01		Media	2.3493E+11	4.0225E+01	4.4435E+01	2.2358E+07
	Peor	-7.0064E-01	-7.0064E-01	-7.0064E-01	-6.7018E-01		Peor	4.2287E+12	6.5159E+01	6.3567E+01	3.5773E+08
	Std	9.8468E-03	9.2459E-03	1.3090E-02	1.9286E-02		Std	9.6864E+11	1.2353E+01	5.9143E+00	8.6593E+07
Test			=	=	=	Test			=	=	=
		CLS	SIN	OPP	ORT			CLS	SIN	OPP	ORT
c02	FR	0	0	0	0	c11	FR	1.0	0.96	0.96	1.0
	Mejor	-	-	-	-		Mejor	-1.5227E-03	-1.5227E-03	-1.5227E-03	-1.5227E-03
	Mediana	-	-	-	-		Mediana	-1.5227E-03	-1.5227E-03	-1.5227E-03	-1.5227E-03
	Media	-	-	-	-		Media	-1.5227E-03	-1.5227E-03	-1.5227E-03	-1.5227E-03
	Peor	-	-	-	-		Peor	-1.5227E-03	-1.5227E-03	-1.5227E-03	-1.5227E-03
	Std	-	-	-	-		Std	1.0956E-12	2.2470E-13	1.7076E-11	5.2103E-12
Test						Test			=	+	+
		CLS	SIN	OPP	ORT			CLS	SIN	OPP	ORT
c03	FR	1.0	1.0	1.0	1.0	c12	FR	0.84	0.84	0.92	0.92
	Mejor	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00		Mejor	-5.7009E+02	-5.7009E+02	-5.7009E+02	-5.7009E+02
	Mediana	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00		Mediana	-1.9925E-01	-1.2667E+01	-3.7245E+02	-4.2313E+02
	Media	0.0000E+00	3.5502E-01	7.1004E-01	0.0000E+00		Media	-1.9169E+02	-2.1587E+02	-2.3534E+02	-2.8489E+02
	Peor	0.0000E+00	8.8756E+00	8.8756E+00	0.0000E+00		Peor	-1.9925E-01	-1.9925E-01	9.7048E+01	-1.9925E-01
	Std	0.0000E+00	1.7392E+00	2.4079E+00	0.0000E+00		Std	2.1770E+02	2.3302E+02	2.3740E+02	2.2048E+02
Test			=	=	=	Test			=	=	=
		CLS	SIN	OPP	ORT			CLS	SIN	OPP	ORT
c04	FR	1.0	1.0	1.0	1.0	c13	FR	1.0	1.0	1.0	1.0
	Mejor	-4.9752E-03	-4.9752E-03	-4.9752E-03	-4.9752E-03		Mejor	-6.8429E+01	-6.8429E+01	-6.8429E+01	-6.8429E+01
	Mediana	-4.9752E-03	-4.9752E-03	-4.9752E-03	-4.9752E-03		Mediana	-6.8429E+01	-6.8429E+01	-6.8429E+01	-6.8429E+01
	Media	-4.9752E-03	-4.9752E-03	-4.9752E-03	-4.9752E-03		Media	-6.8315E+01	-6.8429E+01	-6.8201E+01	-6.8429E+01
	Peor	-4.9752E-03	-4.9752E-03	-4.9752E-03	-4.9752E-03		Peor	-6.5578E+01	-6.8429E+01	-6.5578E+01	-6.8429E+01
	Std	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00		Std	5.5866E-01	3.5188E-08	7.7341E-01	8.7773E-08
Test			=	=	=	Test			=	+	=
		CLS	SIN	OPP	ORT			CLS	SIN	OPP	ORT
c05	FR	0.36	0.2	0.32	0.32	c14	FR	1.0	1.0	1.0	1.0
	Mejor	-4.8361E+02	-4.8361E+02	-4.8361E+02	-4.8361E+02		Mejor	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	Mediana	-4.8361E+02	-4.8361E+02	-4.8315E+02	-4.8361E+02		Mediana	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	Media	-3.9517E+02	-4.8339E+02	-4.8265E+02	-3.5981E+02		Media	7.3527E+06	1.6450E+07	5.8442E+10	2.4307E+09
	Peor	3.1235E+02	-4.8250E+02	-4.8002E+02	5.0681E+02		Peor	1.8382E+08	4.1124E+08	8.2437E+11	6.0768E+10
	Std	2.5015E+02	4.4607E-01	1.2446E+00	3.2755E+02		Std	3.6021E+07	8.0586E+07	1.8028E+11	1.1908E+10
Test			=	-	=	Test			=	=	=
		CLS	SIN	OPP	ORT			CLS	SIN	OPP	ORT
c06	FR	1.0	0.8	0.88	0.84	c15	FR	1.0	1.0	1.0	1.0
	Mejor	-5.7866E+02	-5.7866E+02	-5.7824E+02	-5.7865E+02		Mejor	0.0000E+00	0.0000E+00	3.6732E+00	0.0000E+00
	Mediana	-5.7718E+02	-5.7718E+02	-5.7717E+02	-5.7717E+02		Mediana	3.6732E+00	3.6732E+00	3.6732E+00	3.6732E+00
	Media	-5.7729E+02	-5.6515E+02	-5.7713E+02	-5.4026E+02		Media	1.7955E+12	2.4718E+12	1.1316E+13	9.8601E+12
	Peor	-5.7708E+02	-3.3447E+02	-5.7671E+02	1.4278E+01		Peor	4.3727E+13	2.3508E+13	1.8549E+14	1.1849E+14
	Std	4.0623E-01	5.2923E+01	2.8741E-01	1.3015E+02		Std	8.5623E+12	6.5967E+12	3.7637E+13	2.8649E+13
Test			=	+	=	Test			=	+	=
		CLS	SIN	OPP	ORT			CLS	SIN	OPP	ORT
c07	FR	1.0	1.0	1.0	1.0	c16	FR	0.88	0.48	0.4	0.56
	Mejor	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00		Mejor	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	Mediana	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00		Mediana	0.0000E+00	9.9846E-01	1.0183E+00	9.5881E-01
	Media	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00		Media	3.0576E-01	9.5100E-01	8.5876E-01	6.9038E-01
	Peor	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00		Peor	1.0301E+00	1.2601E+00	1.0338E+00	1.0442E+00
	Std	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00		Std	4.4933E-01	3.0231E-01	3.2516E-01	4.2421E-01
Test			=	=	=	Test			+	+	+
		CLS	SIN	OPP	ORT			CLS	SIN	OPP	ORT
c08	FR	1.0	1.0	1.0	1.0	c17	FR	0.84	0.64	0.88	0.92
	Mejor	2.1561E-12	3.4058E-10	1.2448E-08	1.8512E-12		Mejor	6.1630E-33	0.0000E+00	6.1630E-33	6.1630E-33
	Mediana	2.3961E-08	1.9738E-08	3.1752E-06	6.2260E-09		Mediana	1.0714E-03	1.0749E-03	1.8613E-01	7.9252E-04
	Media	7.4341E-08	3.0447E-08	1.5947E-01	5.2685E-08		Media	3.7799E-01	7.1149E+01	3.6672E+01	2.4958E+01
	Peor	6.8814E-07	1.2441E-07	3.9866E+00	3.8816E-07		Peor	2.3580E+00	6.5977E+02	4.8994E+02	2.9583E+02
	Std	1.4215E-07	3.5226E-08	7.8121E-01	1.0629E-07		Std	6.4214E-01	1.9002E+02	1.1778E+02	7.3432E+01
Test			=	+	=	Test			=	=	=
		CLS	SIN	OPP	ORT			CLS	SIN	OPP	ORT
c09	FR	0.8	0.88	0.64	0.8	c18	FR	0.92	0.92	0.88	0.92
	Mejor	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00		Mejor	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	Mediana	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00		Mediana	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	Media	4.9340E+10	3.7397E+00	2.2519E+03	8.9797E-01		Media	2.7644E+03	1.0504E+03	2.7086E+03	2.8615E+03
	Peor	9.8679E+11	7.7832E+01	3.5985E+04	8.3203E+00		Peor	1.6034E+04	8.4499E+03	1.1711E+04	1.1522E+04
	Std	2.1507E+11	1.6181E+01	8.7099E+03	2.3112E+00		Std	4.4879E+03	2.3650E+03	3.9052E+03	4.3018E+03
Test			=	=	=	Test			=	=	=

Tabla 6.3: Resultados DE_ecm con SIN,OPP, CLS y ORT en 10D.

		CLS	SIN	OPP	ORT			CLS	SIN	OPP	ORT
c01	FR	1.0	0.96	0.92	1.0	c10	FR	0.08	0.08	0.08	0.08
	Mejor	-8.2183E-01	-8.2184E-01	-8.2143E-01	-8.1426E-01		Mejor	1.7895E+07	8.0459E+12	2.3760E+13	6.1395E+09
	Mediana	-8.0826E-01	-8.1063E-01	-8.0726E-01	-7.9626E-01		Mediana	2.3428E+07	3.8140E+13	5.3890E+13	3.8007E+13
	Media	-8.0632E-01	-8.0714E-01	-8.0501E-01	-7.9211E-01		Media	2.3428E+07	3.8140E+13	5.3890E+13	3.8007E+13
	Peor	-7.8001E-01	-7.6236E-01	-7.7514E-01	-7.6639E-01		Peor	2.8962E+07	6.8235E+13	8.4020E+13	7.6007E+13
	Std.	1.0308E-02	1.3505E-02	1.0614E-02	1.3366E-02		Std.	5.5333E+06	3.0094E+13	3.0130E+13	3.8001E+13
Test			=	=	+	Test			=	=	=
		CLS	SIN	OPP	ORT			CLS	SIN	OPP	ORT
c02	FR	0	0	0	0	c11	FR	0	0	0	0
	Mejor	-	-	-	-		Mejor	-	-	-	-
	Mediana	-	-	-	-		Mediana	-	-	-	-
	Media	-	-	-	-		Media	-	-	-	-
	Peor	-	-	-	-		Peor	-	-	-	-
	Std.	-	-	-	-		Std.	-	-	-	-
Test						Test					
		CLS	SIN	OPP	ORT			CLS	SIN	OPP	ORT
c03	FR	0.76	0.68	0.16	1.0	c12	FR	0.88	0.88	0.64	0.8
	Mejor	4.1618E-05	1.1686E+01	1.3243E+03	1.1786E-06		Mejor	-1.9926E-01	-1.9926E-01	-1.9924E-01	-1.9926E-01
	Mediana	1.1072E+02	1.2247E+02	2.3446E+04	1.0589E+02		Mediana	-1.9923E-01	-1.9921E-01	-1.9864E-01	-1.9926E-01
	Media	1.0789E+03	4.3324E+02	9.5394E+04	1.3007E+03		Media	-1.9914E-01	-1.9825E-01	-1.9430E-01	-1.9923E-01
	Peor	9.4373E+03	2.9267E+03	3.3336E+05	1.3027E+04		Peor	-1.9857E-01	-1.8339E-01	-1.6862E-01	-1.9895E-01
	Std.	2.3462E+03	7.9653E+02	1.3854E+05	3.3033E+03		Std.	1.9254E-04	3.2898E-03	8.5979E-03	7.5740E-05
Test			=	+	=	Test			=	+	-
		CLS	SIN	OPP	ORT			CLS	SIN	OPP	ORT
c04	FR	1.0	1.0	1.0	1.0	c13	FR	1.0	1.0	1.0	1.0
	Mejor	-2.6578E-03	-2.6643E-03	-2.6553E-03	-2.6645E-03		Mejor	-5.4377E+01	-6.2995E+01	-4.8029E+01	-5.3616E+01
	Mediana	-2.2954E-03	-2.4696E-03	-2.4126E-03	-2.6374E-03		Mediana	-4.8137E+01	-4.7466E+01	-4.4406E+01	-4.7215E+01
	Media	1.6191E-02	6.2757E-03	1.7295E-02	1.2916E-02		Media	-4.7535E+01	-4.8212E+01	-4.5038E+01	-4.6853E+01
	Peor	2.8718E-01	2.1223E-01	2.5778E-01	3.6280E-01		Peor	-3.8693E+01	-3.6520E+01	-4.2208E+01	-4.0069E+01
	Std.	6.4440E-02	4.2043E-02	6.5689E-02	7.1535E-02		Std.	3.9959E+00	5.1608E+00	1.7039E+00	3.1652E+00
Test			=	=	-	Test			=	+	=
		CLS	SIN	OPP	ORT			CLS	SIN	OPP	ORT
c05	FR	0	0	0	0	c14	FR	1.0	1.0	1.0	1.0
	Mejor	-	-	-	-		Mejor	5.8650E+00	9.1934E+00	1.3633E+01	6.5987E+00
	Mediana	-	-	-	-		Mediana	8.9135E+00	2.4956E+10	5.0081E+12	5.6190E+10
	Media	-	-	-	-		Media	1.0834E+07	1.4514E+13	1.6555E+13	4.4490E+12
	Peor	-	-	-	-		Peor	2.7082E+08	1.2447E+14	8.4794E+13	2.9926E+13
	Std.	-	-	-	-		Std.	5.3069E+07	2.8814E+13	2.2635E+13	8.5151E+12
Test						Test			+	+	+
		CLS	SIN	OPP	ORT			CLS	SIN	OPP	ORT
c06	FR	0	0	0	0	c15	FR	1.0	1.0	1.0	1.0
	Mejor	-	-	-	-		Mejor	2.1603E+01	1.6344E+10	5.8227E+08	1.4854E+08
	Mediana	-	-	-	-		Mediana	2.1605E+01	2.1959E+12	1.3098E+14	5.4955E+13
	Media	-	-	-	-		Media	1.2747E+13	7.9615E+13	2.0643E+14	8.0684E+13
	Peor	-	-	-	-		Peor	2.2063E+14	7.2487E+14	5.5734E+14	4.7239E+14
	Std.	-	-	-	-		Std.	4.6074E+13	1.6262E+14	2.1020E+14	1.1036E+14
Test						Test			+	+	+
		CLS	SIN	OPP	ORT			CLS	SIN	OPP	ORT
c07	FR	1.0	1.0	1.0	1.0	c16	FR	1.0	0.56	0.68	0.56
	Mejor	2.4057E-02	5.7489E+00	8.9720E+00	1.5563E-04		Mejor	0.0000E+00	1.5283E-08	1.2072E-04	9.4158E-13
	Mediana	7.1286E+00	8.2563E+00	1.1059E+01	6.0666E+00		Mediana	3.7420E-12	5.7750E-05	9.6172E-04	1.2656E-05
	Media	6.8354E+00	8.2512E+00	1.3679E+01	5.1022E+00		Media	6.1091E-04	7.8272E-02	1.2836E-01	1.1187E-01
	Peor	1.0158E+01	1.3055E+01	6.8897E+01	1.3154E+01		Peor	9.9486E-03	1.0951E+00	1.0646E+00	1.0332E+00
	Std.	2.7350E+00	1.6522E+00	1.1373E+01	3.1079E+00		Std.	2.1202E-03	2.8202E-01	3.4176E-01	2.7312E-01
Test			=	+	-	Test			+	+	+
		CLS	SIN	OPP	ORT			CLS	SIN	OPP	ORT
c08	FR	1.0	1.0	1.0	1.0	c17	FR	0.68	0.32	0.4	0.44
	Mejor	2.4936E+01	2.4409E+01	2.6727E+01	2.1794E+01		Mejor	3.8480E+00	3.9384E+00	2.7245E+01	3.3094E+00
	Mediana	2.6154E+01	2.6706E+01	2.8808E+01	2.3308E+01		Mediana	1.5139E+02	2.3641E+02	7.9749E+02	7.0634E+01
	Media	3.6637E+01	4.2106E+01	3.7429E+01	3.1193E+01		Media	5.0881E+02	7.5195E+02	1.1378E+03	2.1744E+02
	Peor	1.0426E+02	1.6263E+02	1.5286E+02	1.0388E+02		Peor	2.5460E+03	2.4008E+03	2.9642E+03	1.1080E+03
	Std.	2.4329E+01	3.6843E+01	2.7402E+01	2.1977E+01		Std.	8.7911E+02	8.6506E+02	1.1163E+03	3.2326E+02
Test			=	+	-	Test			=	=	=
		CLS	SIN	OPP	ORT			CLS	SIN	OPP	ORT
c09	FR	0.12	0	0.04	0.08	c18	FR	0.92	0.96	0.8	1.0
	Mejor	6.9593E+08	-	5.7881E+13	1.0062E+09		Mejor	2.0829E+01	2.1975E+03	9.7238E+03	1.3143E+04
	Mediana	9.0384E+09	-	5.7881E+13	8.4853E+12		Mediana	2.9575E+04	3.6354E+04	3.3901E+04	2.9944E+04
	Media	2.2651E+13	-	5.7881E+13	8.4853E+12		Media	2.9972E+04	3.7242E+04	3.6997E+04	3.2944E+04
	Peor	6.7944E+13	-	5.7881E+13	1.6970E+13		Peor	8.6420E+04	6.7696E+04	8.2173E+04	5.1581E+04
	Std.	3.2027E+13	-	0.0000E+00	8.4843E+12		Std.	2.3748E+04	1.5886E+04	1.8234E+04	1.1246E+04
Test			+	=	=	Test			=	=	=

Tabla 6.4: Resultados DE_ecm con SIN,OPP, CLS y ORT en 30D.

6.2.2. Gráficas de diversidad

Otra forma de analizar los resultados es a través de la diversidad de la población de cada variante; permitiendo comparar no solo cuantas veces se resolvió exitosamente el problema en cuestión si no también qué tan cerca estuvo de llegar al mejor valor conocido, la dinámica de la población, la similitud con otras variantes, etc.

A continuación se muestran las gráficas de diversidad en 10 y 30 dimensiones de un conjunto representativo de funciones, en ellas se incluye el valor de aptitud $f(x)$ y la suma de violación de restricciones $cvs(x)$ del mejor individuo de la última población en la ejecución mediana de cada variante (resultado final de la ejecución en la mediana).

El total de gráficas de diversidad, junto con el resto de resultados del experimento, puede encontrarse en el Apéndice B.2.

Cada problema seleccionado representa los resultados obtenidos por las combinaciones de algoritmos:

- $c08$: Representa a los resultados donde no existen diferencias importantes entre ninguna variante de DE ni GA.
- $c05$: Es un ejemplo de la influencia en la diversidad del manejador ϵ constrained method sin conseguir llevar a las variantes que lo implementan al mejor valor conocido de manera contundente.
- $c16$: Muestra la influencia de ϵcm en las variantes que sí llegaron al mejor valor conocido.
- $c06$: Expone el papel de las técnicas de diversidad con mejor desempeño (CLS y ORT en conjunción con ϵcm).

En la sección de discusión de resultados se describen y analizan en profundidad los hallazgos del experimento.

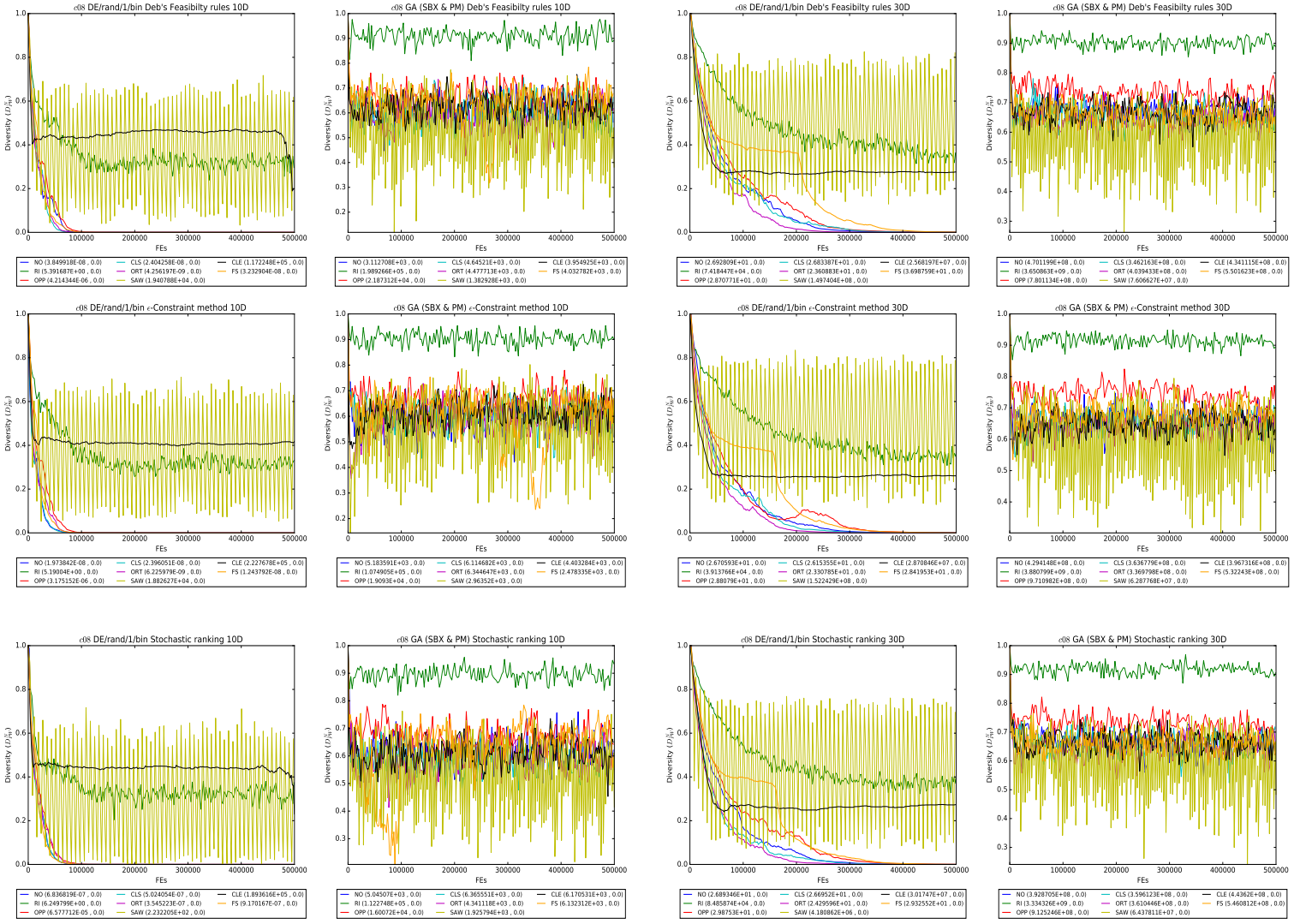

Figura 6.6: Gráficas de diversidad para el problema $c08$ en $10D$ y $30D$.

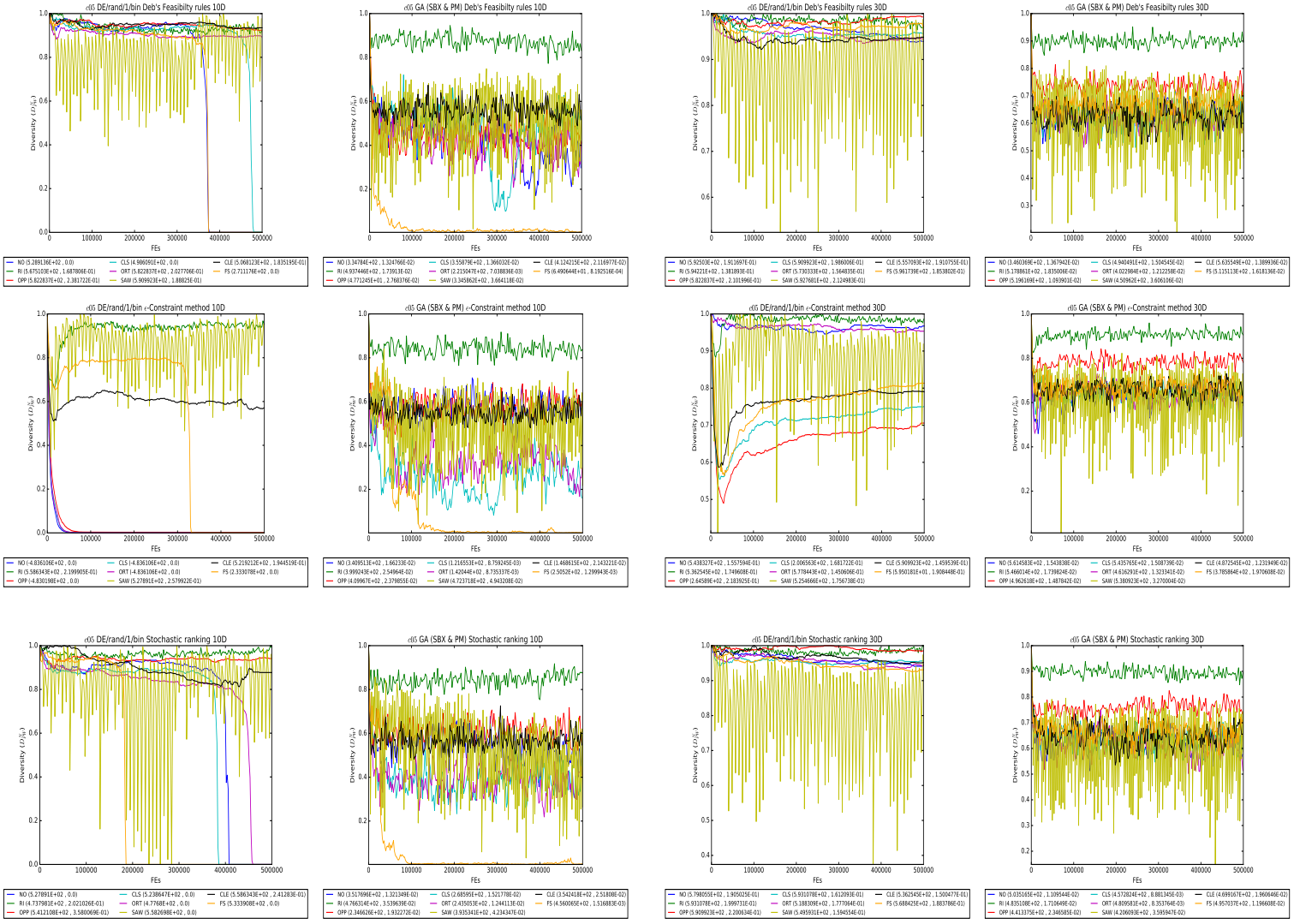

Figura 6.7: Gráficas de diversidad para el problema $c05$ en $10D$ y $30D$.

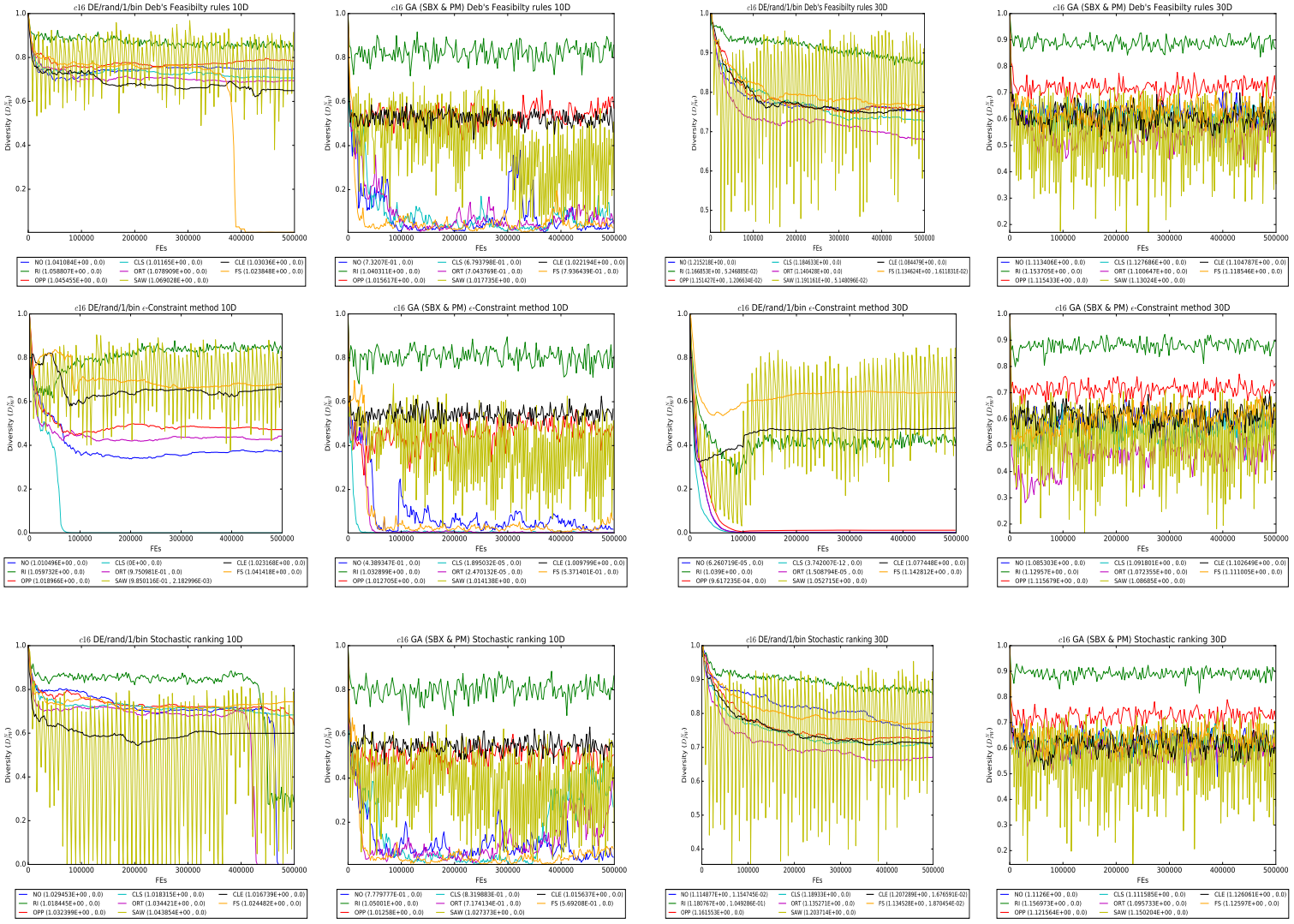

Figura 6.8: Gráficas de diversidad para el problema $c16$ en $10D$ y $30D$.

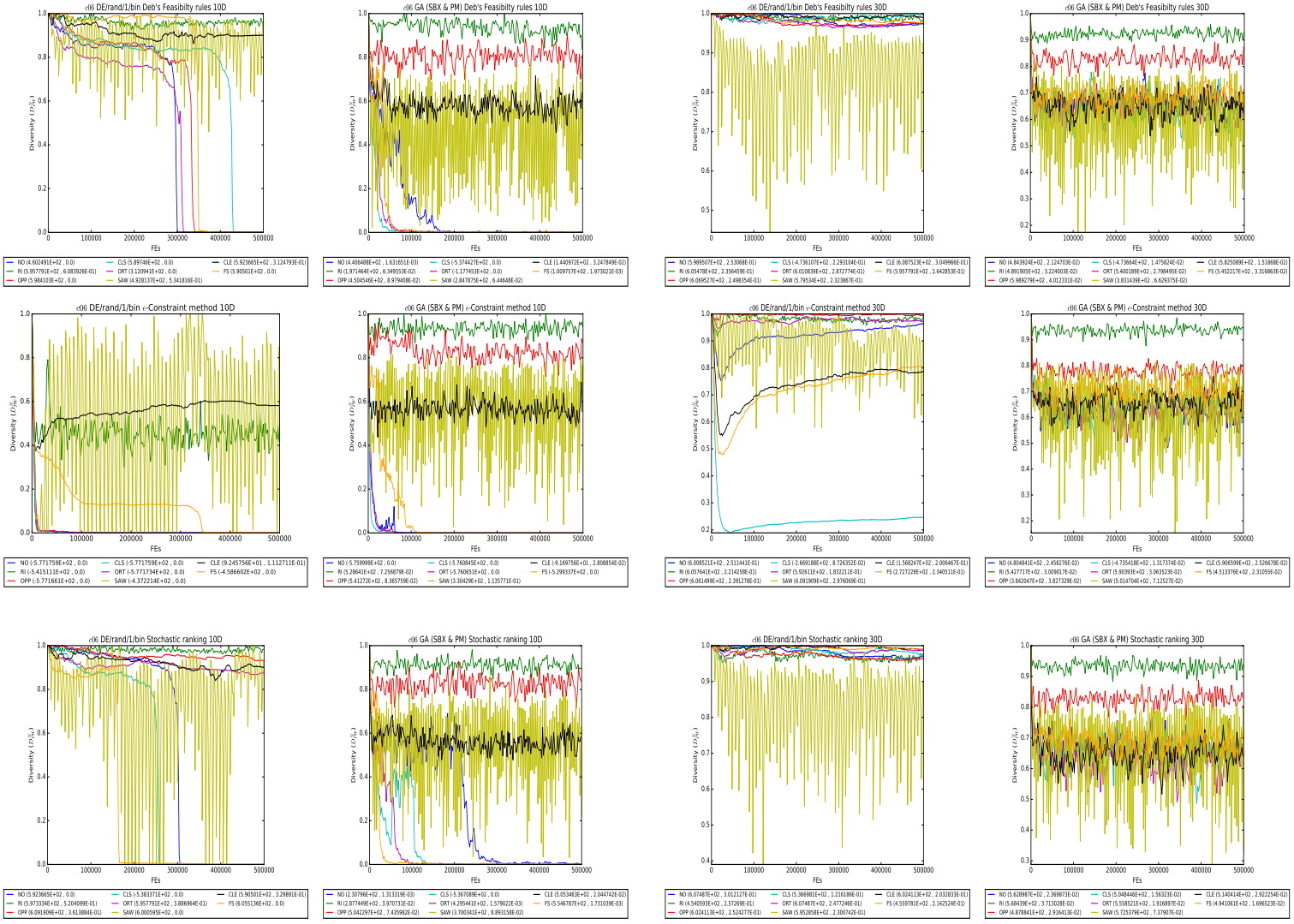

Figura 6.9: Gráficas de diversidad para el problema $c06$ en $10D$ y $30D$.

6.3. Discusión de resultados

6.3.1. Discusión de resultados finales

Al analizar los diagramas de cajas de la tasa éxito y factibilidad (Figuras 6.2, 6.3, 6.4 y 6.5) se puede apreciar que ninguna variante logra factibilidad en todos los problemas y que la mayoría de las variantes sólo consigue llegar al mejor valor conocido en muy pocas funciones. Esta situación era esperada ya que los algoritmos seleccionados no fueron diseñados para optimización con restricciones. Además de ser versiones básicas de algoritmos que han sido mejorados en distintas maneras para generar variantes competitivas en problemas de optimización de distinta índole. Por otro lado, es importante destacar que se usaron los parámetros sugeridos (a excepción de σ para CLE y FS) en las publicaciones originales de algoritmos, manejadores y técnicas. No se realizó una calibración que permitiera alcanzar mejores resultados.

A pesar de los malos resultados existen patrones identificables, los cuales se comentan a continuación.

La dimensionalidad es el factor que merma en mayor medida el desempeño de las variantes, como puede observarse en los diagramas de cajas de FR y SR de las Figuras 6.2, 6.3, 6.4 y 6.5. Algunas variantes de evolución diferencial (DE) logran resolver en al menos una ejecución la mayoría de problemas en 10D pero un reducido número de funciones en 30 dimensiones, mientras que el algoritmo genético (GA) únicamente consigue llegar al mejor valor reportado en pocas funciones con 10D y en ningún caso en 30D[1].

En segundo término se puede ver que todas las variantes del algoritmo genético (GA) presentan resultados considerablemente peores que las combinaciones que implementan evolución diferencial. Existen 20 variantes que parten de DE que consiguieron resolver al menos un problema en 10D y 12 que lo consiguieron en 30D, a diferencia de las 12 combinaciones del algoritmo genético (GA) que lo lograron en 10D mientras que ninguna variante en 30D. Lo anterior también se aprecia en el diagrama de cajas de la tasa de éxito de ambas dimensiones. En 10 dimensiones (Figura 6.4) el final de la línea punteada, correspondiente a los valores extremos, de las variantes de DE consiguen valores más altos que cualquier variante de GA. También al evaluar la Figura 6.5 se aprecia que los valores extremos en la mayoría de variantes de DE se aproximan a 1 , mientras que solo ciertas combinaciones del algoritmo genético consiguen que sus valores extremos se aproximen (en menor medida que DE) a 1.

El tercer elemento visible en los resultados presentados es la superioridad de ϵ constrained method con respecto a los otros dos manejadores: reglas de

[1]Los valores de las tasas de éxito y factibilidad en las 25 ejecuciones de cada problema de cada variante en el estudio se encuentran en el Apéndice B.1.

factibilidad de Deb y Stochastic ranking. Los diagramas de cajas de las Figuras 6.2, 6.3, 6.4 y 6.5 (los resultados de FR y SR completos se encuentran en el apéndice en las Figuras B.3 y B.4) muestran que, acorde a las diferencias en los resultados por parte de DE y GA, las variantes que implementan ϵcm consiguen llegar al mejor valor conocido en muchas más funciones que usando el resto de manejadores. Por ejemplo, las variantes de DE con ϵcm sólo fracasaron en 2 funciones en 10D, las versiones de DE con los demás manejadores de restricciones no consiguieron tasas de éxito con valores mayores a 0 en 12 de 18 problemas del benchmark. En 30D la diferencia es menor, pero se repite el mismo patrón 4 funciones resueltas por la mejor versión con ϵcm y 2 con la mejor variante que implementa tanto *deb* como *sr*.

A nivel de técnicas de promoción también se observa la influencia positiva en los resultados de ciertas técnicas sobre otras. Esta situación es notoria al analizar la tasa de éxito de las combinaciones que implementan ϵcm como manejador con las técnicas CLS, ORT, OPP y la versión sin técnica de promoción de la diversidad, donde se observan los valores más altos de la tasa de éxito (medianas más altas en 10D y valores extremos más altos en 30D). Al considerar el diagrama de cajas, las variantes DE_ϵcm_CLS, DE_ϵcm_SIN, DE_ϵcm_ORT y DE_ϵcm_OPP son las únicas que presentan cajas con volumen cuyos primeros cuartiles sean superiores a 0 (75 % de funciones resueltas en al menos una ejecución en 10D), lo cual contrasta fuertemente con el resto de variantes. De manera similar, entre los malos resultados de GA, las variantes GA_ϵcm_CLS, GA_ϵcm_SIN y GA_ϵcm_ORT son las únicas cuyos valores extremos se aproximan a 0.8, mientras que el resto variantes consigue como máximo 0.4.

A partir de las observaciones anteriores se determinó evaluar aquellas con mejores resultados para aplicar el test no paramétrico de hipótesis de Wilcoxon para determinar el mejor. Los resultados estadísticos de las variantes DE_ϵcm_SIN, DE_ϵcm_OPP, DE_ϵcm_CLS y DE_ϵcm_ORT se proporcionan en las Tablas 6.3 y 6.4, incluyendo la prueba de Wilcoxon entre la variante con mayor número de victorias ($DE\_\epsilon cm\_CLS$) y el resto para cada función en ambas dimensiones.

En primer lugar se observa que los resultados finales son muy similares entre las cuatro variantes, presentando diferencias significativas en muy pocas funciones, especialmente en 10D. Como puede apreciarse en la Tabla 6.2 la variante DE_ϵcm_CLS supera en al menos una función al resto de combinaciones en 10D, mientras en 30 dimensiones vence con un poco más de solvencia a $DE\_\epsilon cm\_SIN$ y $DE\_\epsilon cm\_OPP$. En cambio, existe un empate en 30 dimensiones entre $DE\_\epsilon cm\_CLS$ y $DE\_\epsilon cm\_ORT$.

Considerando todos los resultados se puede afirmar que de las 48 combinaciones de algoritmos, manejadores de restricciones y técnicas de promoción de la diversidad estudiadas las que mejor desempeño presentaron son $DE\_\epsilon cm\_CLS$ y $DE\_\epsilon cm\_ORT$.

6.3.2. Discusión de gráficas de diversidad

Como se ha mencionado en párrafos anteriores, este experimento no busca únicamente identificar las mejores variantes si no también analizar la influencia en la diversidad de los elementos evaluados: algoritmos de optimización, manejadores de restricciones y técnicas de promoción de la diversidad. A continuación se discuten los resultados del estudio de diversidad.

Las gráficas de diversidad agrupadas en las Figuras 6.6, 6.7 6.8 y 6.9, fueron obtenidas de la ejecución ubicada en la mediana de las 25 ejecuciones independientes realizadas por todas las variantes. Esta situación es importante porque la información que las gráficas proveen, a diferencia de la ofrecida por el análisis de la tasa de éxito y factibilidad, no corresponderá necesariamente al mejor resultado obtenido por cada variante. En su defecto, el resultado en la mediana se torna más representativo respecto al desempeño general de cada combinación en la función en turno. Para analizar la influencia de los distintos elementos evaluados es importante contemplar ambos casos, los resultados finales positivos que inspiran la nueva línea de investigación y la influencia de cada uno de estos elementos en el proceso evolutivo.

6.3.2.1. Análisis general

De manera general se observa una gran diferencia en las gráficas de diversidad de las variantes de DE y GA, donde las combinaciones que implementan GA presentan movimientos mucho más abruptos que aquellas variantes que parten de DE.

Este fenómeno es originado por la naturaleza de los algoritmos. En específico gracias a las diferencias en el reemplazo de ambos algoritmos, pues el algoritmo genético únicamente conserva al mejor individuo en cada generación y el resto es sustituido por los descendientes generados en dicha generación. En cambio, evolución diferencial únicamente sustituye individuos cuando el vector trial (producido de la mutación y cruza) supera al vector target, es decir, sólo si se encuentran mejores individuos existirán cambios. Otro motivo por el cual las gráficas de GA, son mucho más escarpadas que las correspondientes a las variantes que surgen de DE es el tipo de operadores seleccionados para GA, especialmente la mutación polinomial. La implementación de este operador se realizó aplicando la actualización de la probabilidad de mutación que describe Deb [19] para incrementar progresivamente dicho valor, de modo que las variaciones serán aún mayores conforme pasan las iteraciones.

Al igual que con los resultados de la tasa de éxito, en el caso de Evolución diferencial se aprecian comportamientos similares en variantes con el mismo manejador de restricciones, de tal suerte que es posible encontrar dos grupos claramente definidos. El primero formado por las reglas de factibilidad de Deb y

Stochastic ranking. Por otro lado se encuentra ϵ constrained method con un comportamiento muy distinto al grupo anterior, reflejando la superioridad encontrada en el análisis estadístico.

A nivel de técnicas de promoción de la diversidad nos encontramos con comportamientos muy similares a partir de las características de dichas técnicas. Las 7 técnicas empleadas en el estudio están catalogadas ya sea como técnicas de infusión (RI, OPP, CLS, ORT y SAW), o como mecanismos de nitching (CLE y FS) [37]. A pesar de esto, el grupo de técnicas de infusión no tiene el mismo nivel de perturbación en la población.

Por ejemplo, inmigrantes aleatorios (RI) y la reinicialización por diente de sierra (SAW) producen modificaciones en la diversidad mayores que el resto de técnicas de infusión, debido a que son aplicadas a nivel población: RI sustituye un porcentaje (30 % en este experimento) de la población cada iteración, lo cual evidentemente causa una constante fluctuación en los niveles de diversidad de la población. SAW tiene un efecto similar, al reducir el tamaño de la población en cada generación hasta llegar al punto de reinicio e introducir nuevamente a los individuos cesados anteriormente. De modo que su nombre no solo ejemplifica lo que sucede con el tamaño poblacional si no también con la diversidad. Ambas técnicas presentan las curvas más escarpadas, expresadas en color verde (RI) y amarillo (SAW).

Por otro lado, la búsqueda local caótica (CLS) y la cruza ortogonal (ORT) solo se aplican sobre un individuo en cada generación. Lo anterior reduce en primer lugar la complejidad del algoritmo y la perturbación en la diversidad de la población. Además, ambas técnicas de promoción tienen la característica de modificar dinámicamente su comportamiento: En el caso de CLS se incluye un factor de encogimiento, reduciendo progresivamente el espacio para la búsqueda local. En cuanto a ORT, propiamente no se define una estrategia de reducción del espacio de búsqueda; sin embargo, conforme la población comienza a converger el espacio entre los individuos involucrados en la cruza (la pareja de padres en GA o los vectores target y trial en DE) se reduce y por consiguiente el muestreo ortogonal se lleva a cabo progresivamente en espacios más pequeños. Las curvas de diversidad, expresadas en color azul claro (CLS) y morado (ORT), reflejan los patrones generales del experimento, así como algunos casos donde solo CLS y ORT consiguen convergencia y resultados cercanos al óptimo.

El caso con la técnica de oposición (OPP) tiene diferente efecto en los algoritmos. Por un lado, en variantes de DE es poco probable que en una generación existan muchos vectores opuestos que sustituyan a sus semejantes simétricos gracias al tipo de reemplazo que DE aplica. Por el contrario, en el algoritmo genético se puede dar el caso de que buena parte de la población opuesta sustituya a los peores individuos de la población. Aunado a ello, el conjunto de problemas utilizado en este estudio (CEC 2010) presenta rotaciones complicando aún más la

posibilidad de encontrar mejores zonas en la población opuesta. La gráfica de diversidad de OPP se denota en color rojo. En general esta técnica se ajusta a los comportamientos generales descritos anteriormente y profundizados a continuación.

Por último se encuentran las técnicas clearing (CLE) y fitness sharing (FS). Estos dos son ejemplos de métodos de nitching, los cuales generan grupos de individuos de acuerdo al radio de nicho seleccionado. Ambos evalúan a los individuos del nicho tomando en cuenta los valores de los miembros en conjunto, a través de la media de sus valores de aptitud. A pesar de ese punto en común, las técnicas tienen una diferencia significativa: mientras que en FS los nichos pueden tener tantos individuos como el radio de nicho alcance a cubrir, en CLE existe un número reducido de miembros permitidos y cuando este número se ve superado los peores elementos son eliminados, dejando su lugar a nuevos individuos aleatorios que son introducidos a la población. En este sentido CLE, cuyas curvas de diversidad se muestran en color negro, presentan dinámicas de diversidad similares a RI, ya que en ambas técnicas existe un reemplazo constante de un porcentaje alto de la población. Por el contrario, las variantes que implementan FS son capaces de converger en ocasiones, puesto que no existe límite de individuos dentro de un nicho que requiera de inyección de individuos aleatorios. El color para las gráficas de FS es el naranja.

A continuación se describe y discute con mayor profundidad las gráficas de diversidad de las 48 variantes estudiadas agrupadas por comportamientos, usando una función representativa de cada caso.

Función $c08$: Sin diferencias en diversidad.
Las gráficas de diversidad de la función $c08$ representan al conjunto de problemas en los que las distintas variantes no tuvieron diferencias respecto a la diversidad ($c01$, $c02$ $c07$, $c08$, $c11$, $c12$ y $c13$); es decir, si se analiza GA se observa el mismo comportamiento en todas las variantes del algoritmo genético. Del mismo modo todas las gráficas de diversidad de las variantes de Evolución Diferencial dibujan trayectorias similares. Esto no implica que GA y DE tengan los mismos resultados finales, simplemente que para este grupo de problemas no existe una influencia tan notable de los distintos manejadores de restricciones ni de las técnicas de promoción de la diversidad.

Una de las razones es que la mayoría de problemas carece de restricciones de igualdad, solamente las funciones $c02$, $c11$ y $c12$ presentan una restricción de igualdad. Las restricciones de igualdad reducen el espacio factible de manera considerable, por lo que el tener un número reducido y especialmente carecer de ellas, permite que las variantes consigan llegar a la zona factible más fácilmente. Una vez que la población ha llegado a la zona factible el papel de los manejadores de restricciones deja de ser tan importante, pues los tres manejadores evaluados difieren especialmente en el trato a soluciones no factibles y no cuando se compa-

ran individuos que no violan las restricciones, con el valor de aptitud $f(x)$ como único criterio.

A pesar de ello, encontrar el área factible no significa dar con el óptimo de la función, como puede apreciarse en las gráficas de diversidad de la función $c08$ (véase la Figura 6.6) de todas las variantes excepto las combinaciones de Evolución Diferencial con SIN, OPP, CLS, ORT y FS en 10D.

Función $c05$: $DE\_\epsilon cm$ y malos resultados finales.
Este es un caso común en los resultados, las funciones $c04$, $c05$, $c09$, $c10$ y $c18$ muestran una dinámica de diversidad de acuerdo al manejador de restricciones. Al igual que en los resultados estadísticos, se aprecia que las variantes de evolución diferencial presentan mayor influencia al usar ϵ constrained method que con el resto de manejadores.

Al observar las gráficas de la Figura 6.7, se aprecia que las correspondientes a ϵ constrained method son distintas que el resto. En el caso de las variantes de Evolución Diferencial este fenómeno es más acentuado, pues las curvas de diversidad de casi todas las combinaciones al usar reglas de Deb y Stochastic ranking como manejador se mantienen con valores de diversidad muy elevados (cercanos a 1), únicamente las curvas de CLS y ORT logran decrementar la diversidad alrededor de 300000 evaluaciones al resolver la función en 10D. En cambio, las gráficas de las variantes de DE con ϵcm presentan un decremento en la diversidad entre las evaluaciones 0 y 100000, siendo más pronunciadas en 10D que en 30D. Lo anterior sucede gracias al esquema del manejador ϵcm, el cual comienza siendo permisivo respecto al grado de violación de restricciones y progresivamente se vuelve más severo, castigando a las soluciones no factibles poco a poco. En los experimentos el umbral de tolerancia desaparece al finalizar el 20 % del total de evaluaciones.

Dado que únicamente utilizando este manejador se consigue un decremento en la diversidad por parte de las variante, se observan las bondades de permitir que soluciones con cierto grado de infactibilidad propongan direcciones de búsqueda en función de su valor de aptitud y se torna interesante estudiar a profundidad qué sucede cuando se permite que ϵcm trabaje por más iteraciones.

En las gráficas correspondientes al algoritmo genético la influencia de ϵcm es más discreta, como ya se comentó el algoritmo genético reemplaza a todos los individuos de la población (a excepción del mejor individuo, gracias al elitismo). La presión de selección se aplica durante la selección de padres para su posterior recombinación y no garantiza que los descendientes sean mejores que sus padres, de tal suerte que la población como conjunto no reduce el valor de $cvs(x)$ en relación al umbral ϵ de manera tan clara como en DE. A pesar de ello, en las gráficas de la función $c05$, y algunos casos del resto de funciones en este grupo, se aprecian ligeras diferencias en relación a los otros manejadores.

Función $c16$**:** $DE\_\epsilon cm$ **y buenos resultados finales.**
La función $c16$ representa al conjunto de problemas cuyas gráficas de diversidad son muy diferentes en las combinaciones que usan ϵcm como manejador y esta diferencia se traduce a mejores resultados. A pesar de que no todas las funciones en las que se identifica este comportamiento ($c03$, $c14$ $c15$, $c16$ y $c17$) finalizan en el mejor valor conocido, se aprecian resultados finales considerablemente mejores que al usar el resto de manejadores de restricciones.

En el caso de la función $c16$, las gráficas de diversidad de la Figura 6.8 de las variantes de Evolución diferencial que implementan ϵcm consiguen decrementar los niveles de diversidad alrededor de la evaluación 100000, es decir en el 20 % del total de iteraciones. En 10D únicamente la variante $DE\_\epsilon cm\_CLS$ consigue llegar al mejor valor reportado, mientras que en 30D también lo consigue la versión sin técnica de promoción y las combinaciones $DE\_\epsilon cm\_ORT$ y $DE\_\epsilon cm\_OPP$ obtienen valores muy cercanos. Es notoria la influencia positiva de ϵcm en el manejo de la diversidad y en los resultados, pues las variantes del resto de combinaciones presentan un estancamiento dado que la diversidad se mantiene casi estática en valores altos (excepto para SAW, cuyas gráficas siempre oscilarán debido a la periódica reducción del tamaño de población y reinicio). Existe un caso atípico en las variantes de DE con la técnica OPP, donde el algoritmo logra converger alrededor de la evaluación 400000 sin que esto lo lleve a mejores resultados finales.

En el algoritmo genético, al igual que en los otros grupos identificados, las disimilitudes en las curvas de diversidad son menos pronunciadas. En este caso se aprecia que las técnicas CLS, ORT, FS y la versión carente de mecanismo promoción tienen menor movimiento al aplicar ϵcm que el resto de manejadores al resolver la función $c16$ en 10D. En 30D las gráficas de diversidad difieren un poco más, especialmente para las técnicas RI, CLS y ORT, las cuales presentan un decremento en la diversidad hasta que se termina la tolerancia a violación de restricciones por parte del manejador ϵcm. El manejo de diversidad propiciado por ϵcm produjo resultados favorables en las variantes $GA\_\epsilon cm\_CLS$ y $GA\_\epsilon cm\_ORT$, pues logran encontrar el óptimo de la función $c16$ en 10D.

Función $c06$**:** $DE\_\epsilon cm\_CLS$ **y buenos resultados finales.**
Finalmente se describe el caso más prometedor: funciones donde la variante $DE\_\epsilon cm\_CLS$ obtuvo comportamientos de diversidad diferentes que resultaron en mejores resultados. No existen muchas funciones que demuestren tan claramente resultados favorables únicamente con esta combinación, a pesar de ello en la mayoría de funciones $DE\_\epsilon cm\_CLS$ presenta los mejores resultados finales junto con $DE\_\epsilon cm\_ORT$.

El caso más claro de este comportamiento lo demuestra la función $c06$, donde no solo utilizando ϵcm como manejador se observan un resultados finales favorables para las combinaciones que implementan CLS: En DE se consigue la violación de restricciones más pequeña entre todas las demás combinaciones que

usan reglas de Deb y Stochastic ranking al resolver la función en 10D y 30D. El caso de GA es similar, donde únicamente las variantes con CLS consiguen individuos factibles muy cerca del mejor valor reportado en 10D.

Una de las razones del éxito de esta técnica es que la perturbación se aplica en el mejor individuo de la población y sólo si éste fue superado se sustituye. Como su nombre lo indica es una búsqueda local, lo que tornaría a la variante $DE\_\epsilon cm\_CLS$ en una especie de algoritmo memético. El otro elemento que ayuda a conseguir mejores resultados es que la variación se vuelve más pequeña conforme avanzan las iteraciones, siendo especialmente útil cuando existen restricciones de igualdad que requieren de movimientos finos para aproximarse a la zona factible y posteriormente al óptimo.

6.4. Conclusiones

Reuniendo los resultados de las pruebas estadísticas y gráficas de diversidad del experimento se puede observar que todos los elementos estudiados tienen influencia en las dinámicas de diversidad y en los resultados finales.

Se observa que la dimensionalidad complica considerablemente los problemas de optimización: DE consigue resolver la mayoría de ellos en 10D y muy pocos en 30D usando las mismas estrategias y parámetros, el caso con GA es similar en el sentido de que puede resolver algunos de los problemas en 10D y ninguno en 30D.

Como se esperaba, los mecanismos que incluye cada algoritmo tienen una influencia muy grande. Los resultados son mejores con DE, en parte gracias a que su reemplazo asegura no perder buenas soluciones, o mantener la estrategia de severidad progresiva al usar ϵ constrained method como manejador de restricciones. A diferencia del algoritmo genético que pierde todas las soluciones a excepción de la mejor, sin que el reemplazo respete necesariamente la estrategia de ϵcm.

La tendencia más fuerte se encuentra en el manejador de restricciones, donde ϵ constrained method superó totalmente a las reglas de factibilidad de Deb y Stochastic ranking en los resultados. Así mismo el único manejador que propició cambios en la diversidad fue ϵcm, cuyo manejo coadyuvó a conseguir buenos resultados.

En cuanto a técnicas de promoción los resultados posicionan a la búsqueda local caótica y la cruza ortogonal como las más prometedoras. La prueba de Wilcoxon realizada coloca a las combinaciones $DE\_\epsilon cm\_CLS$ y $DE\_\epsilon cm\_ORT$, como las mejores variantes del estudio. Debido a que existen pocos problemas con resultados significativamente diferentes entre las variantes $DE\_\epsilon cm\_CLS$, $DE\_\epsilon cm\_SIN$, $DE\_\epsilon cm\_ORT$ y $DE\_\epsilon cm\_OPP$, el estudio de diversidad cobra importancia, pues permite observar más allá de los resultados finales. Ambos

resultados en conjunto permitieron corroborar que CLS y ORT tienen un efecto en la diversidad que se reflejó en los resultados finales.

Capítulo 7

Conclusiones y trabajo futuro

7.1. Conclusiones

La relación entre la diversidad y el desempeño de los algoritmos bio-inspirados es reconocida en el área, pues este tipo de algoritmos ha sido exitoso en buena parte gracias a su compromiso entre exploración y explotación. En este sentido, han surgido diversos trabajos relacionados con la diversidad; mecanismos en busca de promover la diversidad en la población y así aspirar a mejores resultados finales, medidas de diversidad que sirvan de indicadores del nivel de variación entre los individuos, entre otros. Desafortunadamente la mayoría de trabajos ha sido enfocado en optimización sin restricciones, multi-objetivo o dinámica, pocos trabajos existen acerca de la diversidad al tratar problemas restringidos. Bajo este panorama emerge la motivación para este trabajo.

Este documento presentó un estudio de diversidad en optimización evolutiva con restricciones desde distintas perspectivas, el cual comprende dos experimentos.

El primero de ellos consistió en un comparativo empírico de medidas de diversidad usando algoritmos del estado del arte para optimización con restricciones, para determinar el comportamiento de 6 medidas de diversidad. Se seleccionaron dos algoritmos de diferente naturaleza, CMODE[32] y SAM-PSO[30], uno basado en Evolución diferencial y otro en optimización por cúmulos de partículas (PSO). Las medidas de diversidad consideradas en el comparativo son catalogadas como medidas de diversidad genotípica, pues expresan la disimilitud de acuerdo a la posición de los individuos en el espacio de búsqueda.

De acuerdo a los resultados finales y las gráficas de diversidad se observa que las medidas de diversidad que mejor desempeño tienen son aquellas que contemplan a todos los individuos de la población para emitir un valor. Destacan las medidas D_{TAP}^{N2} y D_{PW}^{N}, donde al igual que en el trabajo de Corriveau en optimización sin restricciones[14], presentan los mejores resultados.

Del primer experimento se obtienen dos conclusiones:

- **Medidas con mayor poder de carterización:** Las medidas de diversidad D_{TAP}^{N2} y D_{PW}^{N}, son las que mejores resultados obtuvieron al tomar en cuenta a todos individuos en la misma proporción, mostrando los valores esperados cuando existe convergencia y estancamiento.

- **Se requiere un estudio con mayor control:** El manejo de la diversidad depende altamente de los mecanismos que implementa cada algoritmo, pues cada decisión que éstos toman tiene un efecto en el balance entre exploración y explotación. Por este motivo, para obtener conclusiones en cuanto a manejo de la diversidad en un comparativo entre algoritmos del estado del arte, cuyos operadores fueron diseñados específicamente para resolver problemas restringidos de manera competitiva, se requiere tener control sobre los distintos elementos involucrados en el proceso de optimización para encontrar en qué sentido y proporciones influyen en el manejo de diversidad.

Bajo esta óptica, se diseñó el segundo experimento, con el objetivo de contrastar de manera más controlada los distintos elementos que influyen en el manejo de diversidad: algoritmos de optimización con operadores específicos, manejadores de restricciones y técnicas de promoción de la diversidad. Los algoritmos seleccionados fueron las versiones básicas de dos algoritmos conocidos: Evolución diferencial [31] y el algoritmo genético, para permitir la integración con los otros elementos. Se seleccionaron 3 manejadores de restricciones distintos: Reglas de factibilidad de Deb [19], ϵ constrained method [34] y Stochastic ranking [35]. Además se incluyeron 7 técnicas de promoción de diversidad: inmigrantes aleatorios (RI)[42], oposición (OPP)[52], búsqueda local caótica (CLS)[53], cruza ortogonal (ORT)[54], reinicio por diente de sierra (SAW)[55], clearing[49] y fitness sharing (FS)[74].

En total se compararon 48 combinaciones de algoritmo, manejador y técnica de promoción de diversidad al resolver las funciones de prueba del benchmark CEC 2010, las cuales están disponibles en 10 y 30 dimensiones.

Los resultados estadísticos junto el estudio de diversidad demuestran el papel de cada elemento evaluado, listados en orden de relevancia en los resultados del experimento:

- **Dimensionalidad**: Es el elemento que mayor influencia tuvo en los resultados. La maldición de la dimensionalidad se ve latente al complicar que los individuos encuentren tan solo la zona factible, especialmente en problemas con restricciones de igualdad.

- **Algoritmo**: Los operadores específicos de cada algoritmo, al ser la estrategia base para encontrar el óptimo, dictan en gran medida el manejo de

la diversidad. En el experimento, las combinaciones que parten de DE consiguen mejores resultados y curvas de diversidad con movimientos menos abruptos que las variantes de GA, gracias a los operadores y decisiones puntuales que cada uno implementa. Un ejemplo es la estrategia de reemplazo de DE, el cual realiza el reemplazo de un individuo al terminar cada cruza, en vez del reemplazo generacional de GA.

- **Manejador de restricciones**: Al analizar los resultados se aprecia que el manejador de restricciones es un factor muy determinante en las curvas de diversidad, donde en ambos algoritmos se aprecia mayor juego en la dinámica de la diversidad utilizando ϵ constrained method como manejador. Debido a que ϵcm emplea una heurística donde se inicia el proceso de optimización y progresivamente es capaz de desplazar algunos individuos a zonas con cierto grado de infactibilidad pero con buenos valores de aptitud, con la posibilidad de que una de estas nuevas direcciones de búsqueda (las cuales no habrían sido visitadas usando otros manejadores que priorizan la factibilidad a lo largo de toda la búsqueda) produzcan movimientos hacia el óptimo de la función. En caso de no ser útiles, dichas direcciones de búsqueda serán sacrificadas al decrementarse el umbral ϵ.

- **Técnicas de promoción de la diversidad**: Dentro de las cuatro variables fue la que menor influencia tuvo en los resultados finales. Sin embargo, dos técnicas destacan del resto de mecanismos de promoción de la diversidad estudiados: la búsqueda local caótica [53] y la cruza ortogonal [54]. Las combinaciones de Evolución diferencial, ϵ contrained method y estas técnicas de promoción consiguieron los mejores resultados del estudio. El punto en común en estos mecanismos reside en que ambos trabajan una vez por iteración, modificando únicamente la ubicación de un individuo de la población, a diferencia del resto de técnicas, las cuales son de carácter más disruptivo. Por otro lado ambas técnicas reducen el espacio en el que trabajan conforme avanzan las generaciones del proceso evolutivo, de tal suerte que las combinaciones $DE\_\epsilon cm\_CLS$ y $DE\_\epsilon cm\_ORT$ se podrían catalogar como meméticas.

Finalmente, la diversidad en optimización evolutiva es un fenómeno complicado para estudiar, puesto que las dimensiones de los problemas no permiten categorizar las dinámicas de la población visualmente. En su defecto se recurre al uso de medidas de diversidad que a partir de un valor numérico presenten información de la proximidad de la población en determinado momento. Analizar el comportamiento de las curvas de diversidad en relación con los resultados finales parece ser la única manera posible para analizar la dinámica de la población y rastrear comportamientos en los procesos de optimización.

Estudios en el mismo sentido que los presentados en este trabajo, especialmente empleando una metodología similar al segundo experimento, son necesarios para comprender mejor la influencia de ciertas estrategias en la dinámica de la población y el manejo de diversidad.

7.2. Trabajo futuro

Los resultados de los experimentos realizados ofrecen nuevas líneas de investigación para enriquecer y corroborar los hallazgos en diferentes sentidos, algunas de estas opciones se comentan a continuación:

- Evaluar el desempeño de otras medidas de diversidad surge como una opción. En ambos experimentos únicamente se aplicaron medidas en el espacio genotípico. Existen otras medidas de diversidad, como las medidas del espacio fenotípico, las cuales interpretan la diversidad como la disimilitud respecto a los valores de aptitud de los individuos. Sería especialmente interesante incluir en este tipo de medidas el valor de la suma de violación de restricciones, pues este valor es en buena parte del proceso evolutivo el primer elemento a minimizar.

- Realizar un estudio de los parámetros de las combinaciones con mejores resultados del experimento 2. Como se mencionó anteriormente, el experimento utilizó los parámetros sugeridos en las publicaciones de cada técnica. Una calibración de parámetros adecuada puede otorgar a los algoritmos un desempeño competitivo.

- Considerar más algoritmos básicos que permitan un diseño similar al Experimento 2. Estrategias evolutivas o programación evolutiva son ejemplos de algoritmos bio-inspirados que podrían añadirse a un comparativo empírico similar al Experimento 2.

- Incluir otro tipo de mecanismos de promoción de la diversidad. En el Experimento 2 las 7 técnicas estudias caen en las categorías de infusión y nitching. Existen mecanismos de promoción de la diversidad como las restricciones de emparejamiento, la distancia de crowding, etc. Sería interesante evaluar su influencia en el manejo de la diversidad de los algoritmos y los resultados finales.

Bibliografía

[1] K. Deb, "Genetic algorithm in search and optimization: The technique and applications," in *Proc. of Int. Workshop on Soft Computing and Intelligent Systems*, 1997.

[2] A. E. Eiben and J. E. Smith, *Introduction to evolutionary computing*. Springer, 2003, vol. 53.

[3] S. Binitha and S. S. Sathya, "A survey of bio inspired optimization algorithms," *International Journal of Soft Computing and Engineering*, vol. 2, no. 2, pp. 137–151, 2012.

[4] M. L. Mauldin, "Maintaining diversity in genetic search." in *AAAI*, 1984, pp. 247–250.

[5] P. A. Diaz-Gomez and D. F. Hougen, "Empirical study: Initial population diversity and genetic algorithm performance." *Artificial Intelligence and Pattern Recognition*, vol. 2007, pp. 334–341, 2007.

[6] T. Friedrich, P. S. Oliveto, D. Sudholt, and C. Witt, "Analysis of diversity-preserving mechanisms for global exploration," *Evolutionary Computation*, vol. 17, no. 4, pp. 455–476, 2009.

[7] P. A. Bosman and D. Thierens, "The balance between proximity and diversity in multiobjective evolutionary algorithms," *IEEE transactions on evolutionary computation*, vol. 7, no. 2, pp. 174–188, 2003.

[8] B. Sareni and L. Krahenbuhl, "Fitness sharing and niching methods revisited," *IEEE Transactions on Evolutionary computation*, vol. 2, no. 3, pp. 97–106, 1998.

[9] D. Gupta and S. Ghafir, "An overview of methods maintaining diversity in genetic algorithms," *International journal of emerging technology and advanced engineering*, vol. 2, no. 5, pp. 56–60, 2012.

[10] R. K. Ursem, "Diversity-guided evolutionary algorithms," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2002, pp. 462–471.

[11] M. Yang, C. Li, Z. Cai, and J. Guan, "Differential evolution with auto-enhanced population diversity," *IEEE transactions on cybernetics*, vol. 45, no. 2, pp. 302–315, 2015.

[12] D. Zaharie, "Control of population diversity and adaptation in differential evolution algorithms," in *Proc. of MENDEL*, vol. 9, 2003, pp. 41–46.

[13] F. Herrera and M. Lozano, "Adaptation of genetic algorithm parameters based on fuzzy logic controllers," *Genetic Algorithms and Soft Computing*, vol. 8, pp. 95–125, 1996.

[14] G. Corriveau, R. Guilbault, A. Tahan, and R. Sabourin, "Review and study of genotypic diversity measures for real-coded representations," *IEEE transactions on evolutionary computation*, vol. 16, no. 5, pp. 695–710, 2012.

[15] V. Tirronen and F. Neri, "Differential evolution with fitness diversity self-adaptation," in *Nature-inspired algorithms for optimisation*. Springer, 2009, pp. 199–234.

[16] G. Corriveau, R. Guilbault, A. Tahan, and R. Sabourin, "Evaluation of genotypic diversity measurements exploited in real-coded representation," *arXiv preprint arXiv:1507.00088*, 2015.

[17] O. Olorunda and A. P. Engelbrecht, "Measuring exploration/exploitation in particle swarms using swarm diversity," in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*. IEEE, 2008, pp. 1128–1134.

[18] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization," *KanGAL report*, vol. 2005005, p. 2005, 2005.

[19] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer methods in applied mechanics and engineering*, vol. 186, no. 2, pp. 311–338, 2000.

[20] N. Andréasson, A. Evgrafov, and M. Patriksson, "An introduction to optimization: Foundations and fundamental algorithms," 2005.

[21] K. Deb, *Optimization for engineering design: Algorithms and examples*. PHI Learning Pvt. Ltd., 2012.

[22] X.-S. Yang, *Engineering optimization: an introduction with metaheuristic applications*. John Wiley & Sons, 2010.

[23] W. Spears, K. De Jong, T. Bäck, D. Fogel, and H. De Garis, "An overview of evolutionary computation," in *Machine Learning: ECML-93*. Springer, 1993, pp. 442–459.

[24] K. A. De Jong, "Analysis of the behavior of a class of genetic adaptive systems," 1975.

[25] L. B. Booker, "Intelligent behavior as an adaptation to the task environment; part ii." 1982.

[26] J. E. Baker, "Reducing bias and inefficiency in the selection algorithm," in *Proceedings of the second international conference on genetic algorithms*, 1987, pp. 14–21.

[27] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," *Foundations of genetic algorithms*, vol. 1, pp. 69–93, 1991.

[28] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4. IEEE, Nov. 1995, pp. 1942–1948. [Online]. Available: http://dx.doi.org/10.1109/icnn.1995.488968

[29] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on.* IEEE, 1995, pp. 39–43.

[30] S. M. Elsayed, R. A. Sarker, and E. Mezura-Montes, "Self-adaptive mix of particle swarm methodologies for constrained optimization," *Information sciences*, vol. 277, pp. 216–233, 2014.

[31] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[32] Y. Wang and Z. Cai, "Combining multiobjective optimization with differential evolution to solve constrained optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 1, pp. 117–134, 2012.

[33] C. A. C. Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art," *Computer methods in applied mechanics and engineering*, vol. 191, no. 11, pp. 1245–1287, 2002.

[34] T. Takahama and S. Sakai, "Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation," in *Evolutionary Computation (CEC), 2010 IEEE Congress on.* IEEE, 2010, pp. 1–9.

[35] T. P. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *IEEE Transactions on evolutionary computation*, vol. 4, no. 3, pp. 284–294, 2000.

[36] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs (2Nd, Extended Ed.).* New York, NY, USA: Springer-Verlag New York, Inc., 1994.

[37] M. Črepinšek, S.-H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: A survey," *ACM Computing Surveys (CSUR)*, vol. 45, no. 3, p. 35, 2013.

[38] Y.-Y. Wong, K.-H. Lee, K.-S. Leung, and C.-W. Ho, "A novel approach in parameter adaptation and diversity maintenance for genetic algorithms," *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, vol. 7, no. 8, pp. 506–515, 2003.

[39] K. A. De Jong and W. M. Spears, "A formal analysis of the role of multi-point crossover in genetic algorithms," *Annals of mathematics and Artificial intelligence*, vol. 5, no. 1, pp. 1–26, 1992.

[40] H.-G. Beyer and K. Deb, "On self-adaptive features in real-parameter evolutionary algorithms," *IEEE Transactions on evolutionary computation*, vol. 5, no. 3, pp. 250–270, 2001.

[41] N. F. McPhee and N. J. Hopper, "Analysis of genetic diversity through population history," in *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 2*. Morgan Kaufmann Publishers Inc., 1999, pp. 1112–1120.

[42] J. J. Grefenstette *et al.*, "Genetic algorithms for changing environments," in *PPSN*, vol. 2, 1992, pp. 137–144.

[43] J. Zhang and A. C. Sanderson, "Jade: adaptive differential evolution with optional external archive," *IEEE Transactions on evolutionary computation*, vol. 13, no. 5, pp. 945–958, 2009.

[44] K. Matsui, "New selection method to improve the population diversity in genetic algorithms," in *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on*, vol. 1. IEEE, 1999, pp. 625–630.

[45] L. J. Eshelman and J. D. Schaffer, "Preventing premature convergence in genetic algorithms by preventing incest." in *ICGA*, vol. 91, 1991, pp. 115–122.

[46] E. Ronald, "When selection meets seduction," in *Proceedings of the 6th International Conference on Genetic Algorithms*. Morgan Kaufmann Publishers Inc., 1995, pp. 167–173.

[47] K. Deb and D. E. Goldberg, "An investigation of niche and species formation in genetic function optimization," in *Proceedings of the 3rd international conference on genetic algorithms*. Morgan Kaufmann Publishers Inc., 1989, pp. 42–50.

[48] H. G. Cobb, "An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments," NAVAL RESEARCH LAB WASHINGTON DC, Tech. Rep., 1990.

[49] A. Pétrowski, "A clearing procedure as a niching method for genetic algorithms," in *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*. IEEE, 1996, pp. 798–803.

[50] W. Sheng, P. Shan, S. Chen, Y. Liu, and F. E. Alsaadi, "A niching evolutionary algorithm with adaptive negative correlation learning for neural network ensemble," *Neurocomputing*, vol. 247, pp. 173–182, 2017.

[51] S. Rahnamayan, H. R. Tizhoosh, and M. M. Salama, "Opposition versus randomness in soft computing techniques," *Applied Soft Computing*, vol. 8, no. 2, pp. 906–918, 2008.

[52] ——, "Opposition-based differential evolution," *IEEE Transactions on Evolutionary computation*, vol. 12, no. 1, pp. 64–79, 2008.

[53] D. Jia, G. Zheng, and M. K. Khan, "An effective memetic differential evolution algorithm based on chaotic local search," *Information Sciences*, vol. 181, no. 15, pp. 3175–3187, 2011.

[54] Y. Wang, Z. Cai, and Q. Zhang, "Enhancing the search ability of differential evolution through orthogonal crossover," *Information Sciences*, vol. 185, no. 1, pp. 153–177, 2012.

[55] V. K. Koumousis and C. P. Katsaras, "A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 19–28, 2006.

[56] W. B. Langdon, "Data structures and genetic programming: genetic programming+ data structures= automatic programming," 1998.

[57] A. L. Barker and W. N. Martin, "Dynamics of a distance-based population diversity measure," in *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, vol. 2. IEEE, 2000, pp. 1002–1009.

[58] L. Masisi, V. Nelwamondo, and T. Marwala, "The use of entropy to measure structural diversity," in *Computational Cybernetics, 2008. ICCC 2008. IEEE International Conference on*. IEEE, 2008, pp. 41–45.

[59] M. Hutter and S. Legg, "Fitness uniform optimization," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 5, pp. 568–589, 2006.

[60] E. Burke, S. Gustafson, G. Kendall, and N. Krasnogor, "Advanced population diversity measures in genetic programming," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2002, pp. 341–350.

[61] I. Paenke, Y. Jin, and J. Branke, "Balancing population-and individual-level adaptation in changing environments," *Adaptive Behavior*, vol. 17, no. 2, pp. 153–174, 2009.

[62] F. Rothlauf, "Representations for genetic and evolutionary algorithms," in *Representations for Genetic and Evolutionary Algorithms*. Springer, 2006, pp. 9–32.

[63] E. Galván-López, J. McDermott, M. O'Neill, and A. Brabazon, "Towards an understanding of locality in genetic programming," in *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. ACM, 2010, pp. 901–908.

[64] R. W. Morrison and K. A. De Jong, "Measurement of population diversity," in *International Conference on Artificial Evolution (Evolution Artificielle)*. Springer, 2001, pp. 31–41.

[65] M. Wineberg and F. Oppacher, "The underlying similarity of diversity measures used in evolutionary computation," in *Genetic and Evolutionary Computation—GECCO 2003*. Springer, 2003, pp. 206–206.

[66] B. Lacevic and E. Amaldi, "Ectropy of diversity measures for populations in euclidean space," *Information Sciences*, vol. 181, no. 11, pp. 2316–2339, 2011.

[67] J. J. Liang and P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism," in *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*. IEEE, 2006, pp. 9–16.

[68] J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, P. Suganthan, C. C. Coello, and K. Deb, "Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization," *Journal of Applied Mechanics*, vol. 41, no. 8, 2006.

[69] R. B. Agrawal, K. Deb, and R. Agrawal, "Simulated binary crossover for continuous search space," *Complex systems*, vol. 9, no. 2, pp. 115–148, 1995.

[70] K. Deb and S. Agrawal, "A niched-penalty approach for constraint handling in genetic algorithms," in *Proceedings of the international conference on artificial neural networks and genetic algorithms (ICANNGA-99)*, 1999, pp. 235–243.

[71] C. Segura, A. Hernández-Aguirre, F. Luna, and E. Alba, "Improving diversity in evolutionary algorithms: New best solutions for frequency assignment," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 4, pp. 539–553, 2017.

[72] R. Mallipeddi and P. N. Suganthan, "Problem definitions and evaluation criteria for the cec 2010 competition on constrained real-parameter optimization," *Nanyang Technological University, Singapore*, 2010.

[73] S. Elsayed, R. Sarker, and C. C. Coello, "Enhanced multi-operator differential evolution for constrained optimization," in *Evolutionary Computation (CEC), 2016 IEEE Congress on*. IEEE, 2016, pp. 4191–4198.

[74] D. E. Goldberg, J. Richardson *et al.*, "Genetic algorithms with sharing for multimodal function optimization," in *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*. Hillsdale, NJ: Lawrence Erlbaum, 1987, pp. 41–49.

Apéndices

Apéndice A

Detalle de funciones del benchmark CEC 2010

Problem/Search Range	Type of Objective	Number of Constraints		Feasibility Region (ρ)	
		E	*I*	10D	30D
C01 $[0,10]^D$	Non Separable	0	2 Non Separable	0.997689	1.000000
C02 $[-5.12,5.12]^D$	Separable	1 Separable	2 Separable	0.000000	0.000000
C03 $[-1000,1000]^D$	Non Separable	1 Non Separable	0	0.000000	0.000000
C04 $[-50,50]^D$	Separable	4 2 Non Separable, 2 Separable	0	0.000000	0.000000
C05 $[-600,600]^D$	Separable	2 Separable	0	0.000000	0.000000
C06 $[-600,600]^D$	Separable	2 Rotated	0	0.000000	0.000000
C07 $[-140,140]^D$	Non Separable	0	1 Separable	0.505123	0.503725
C08 $[-140,140]^D$	Non Separable	0	1 Rotated	0.379512	0.375278
C09 $[-500500]^D$	Non Separable	1 Separable	0	0.000000	0.000000
C10 $[-500,500]^D$	Non Separable	1 Rotated	0	0.000000	0.000000
C11 $[-100,100]^D$	Rotated	1 Non Separable	0	0.000000	0.000000
C12 $[-1000,1000]^D$	Separable	1 Non Separable	1 Separable	0.000000	0.000000
C13 $[-500,500]^D$	Separable	0	3 2 Separable, 1 Non Separable	0.000000	0.000000
C14 $[-1000,1000]^D$	Non Separable	0	3 Separable	0.003112	0.006123
C15 $[-1000,1000]^D$	Non Separable	0	3 Rotated	0.003210	0.006023
C16 $[-10,10]^D$	Non Separable	2 Separable	2 1 Separable, 1 Non Separable	0.000000	0.000000
C17 $[-10,10]^D$	Non Separable	1 Separable	2 Non Separable	0.000000	0.000000
C18 $[-50,50]^D$	Non Separable	1 Separable	1 Separable	0.000010	0.000000

Figura A.1: Definiciones de los problemas del benchmarck CEC 2010.

Apéndice B

Resultados completos experimento 2

B.1. Estadísticas experimento 2

Evolución diferencial (DE)

	Reglas de Deb (deb)								**ϵ constrained method (ϵcr)**								**Stochastic ranking (sr)**							
	SIN	RI	OPP	CLS	ORT	SAW	CLE	FS	SIN	RI	OPP	CLS	ORT	SAW	CLE	FS	SIN	RI	OPP	CLS	ORT	SAW	CLE	FS
c01	1	1	1	1	1	1	1	1	1	1	0.92	1	0.96	1	1	1	1	1	1	1	1	1	1	1
c02	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c03	1	1	1	1	1	0	1	0.44	1	1	1	1	1	0	1	1	1	0.72	1	1	1	0.32	1	0.44
c04	1	1	1	1	1	1	0	1	1	1	1	1	1	0.92	0	1	1	1	1	1	1	1	0	0.96
c05	0.04	0	0	0.04	0	0	0	0.08	0.2	0	0.32	0.36	0.32	0	0	0.08	0.08	0	0	0.04	0.04	0.04	0	0.24
c06	0.08	0	0.12	0.36	0.08	0	0	0.12	0.8	0.68	0.88	1	0.84	0.12	0	0.04	0.12	0	0	0.36	0	0.32	0	0.08
c07	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
c08	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
c09	0.16	0.04	0.04	0.04	0	0	0.08	0.04	0.88	0.56	0.64	0.8	0.8	0.08	0.12	0.64	0.04	0.04	0	0	0.08	0.08	0.04	0
c10	0.12	0.04	0.08	0.04	0	0	0	0.04	0.76	0.44	0.8	0.72	0.64	0.08	0.24	0.8	0.04	0	0.08	0	0	0	0.16	0
c11	1	0	1	1	1	0	0	0.92	0.96	0.04	0.96	1	1	0	0	0.92	1	0	1	1	1	0	0	0.88
c12	1	0.96	1	1	1	0.12	0	1	0.84	1	0.92	0.84	0.92	0.24	0	1	1	0.96	1	1	1	0.68	0	0.92
c13	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
c14	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
c15	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
c16	0.08	0.12	0.32	0.2	0.08	0.08	0.56	0.4	0.48	0.08	0.4	0.88	0.56	0	0.64	0.32	0.32	0.12	0.4	0.44	0.36	0.8	0.44	0.44
c17	0.36	0.44	0.56	0.56	0.4	0.4	0.56	0.64	0.64	0.84	0.88	0.84	0.92	0.56	0.64	0.72	0.56	0.6	0.52	0.24	0.44	0.48	0.4	0.52
c18	0.84	0.72	0.84	0.8	0.88	0.88	0.48	0.76	0.92	0.76	0.88	0.92	0.92	0.76	0.76	0.76	0.76	0.84	0.68	0.76	0.8	0.84	0.92	0.76

Algoritmo genético (GA)

	Reglas de Deb (deb)								**ϵ constrained method (ϵcr)**								**Stochastic ranking (sr)**							
	SIN	RI	OPP	CLS	ORT	SAW	CLE	FS	SIN	RI	OPP	CLS	ORT	SAW	CLE	FS	SIN	RI	OPP	CLS	ORT	SAW	CLE	FS
c01	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
c02	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c03	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c04	0.56	0	0.32	0.48	0.36	0	0	0.44	1	0	0.64	1	1	0	0	1	0.36	0	0.28	0.52	0.48	0	0	0.16
c05	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c06	0	0	0	0.2	0.04	0	0	0	0.44	0	0	0.6	0.44	0	0	0.04	0	0	0	0.12	0	0	0	0
c07	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
c08	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
c09	0.24	0	0.12	0.08	0.24	0.12	0.08	0.24	0.48	0.04	0.16	0.44	0.32	0.12	0.08	0.52	0.08	0	0.08	0.12	0.08	0.16	0.04	0.24
c10	0.2	0.04	0.12	0.12	0.04	0.16	0.04	0.48	0.64	0.04	0.24	0.68	0.6	0.12	0	0.68	0.04	0.12	0.12	0.12	0.12	0.08	0.12	0.4
c11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c13	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
c14	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
c15	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
c16	1	0.72	0.88	1	1	0.84	0.88	1	1	0.8	0.92	1	1	1	1	1	1	0.76	0.88	1	1	0.92	1	1
c17	1	0.84	1	1	1	0.92	1	1	1	0.84	0.96	1	1	0.88	0.96	1	1	0.84	1	1	1	0.88	0.96	1
c18	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Tabla B.1: Tasa de factibilidad (FR) para cada problema del CEC 2010 en 10D.

Evolución diferencial (DE)

	Reglas de Deb (deb)								**ϵ constrained method (ϵcr)**								**Stochastic ranking (sr)**							
	SIN	RI	OPP	CLS	ORT	SAW	CLE	FS	SIN	RI	OPP	CLS	ORT	SAW	CLE	FS	SIN	RI	OPP	CLS	ORT	SAW	CLE	FS
c01	1	1	1	1	1	1	1	1	0.96	1	0.92	1	1	1	0.96	1	1	1	1	1	1	1	1	1
c02	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c03	0.04	0	0	0.04	0.4	0	0.04	0	0.68	0	0.16	0.76	1	0	0	0	0	0	0	0	0.08	0	0	0
c04	1	0	1	0.92	1	0	0	0.64	1	0	1	1	1	0	0	1	0.96	0	1	1	0.88	0	0	0.56
c05	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c06	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c07	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
c08	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
c09	0	0	0.04	0	0.04	0	0	0	0	0.04	0.04	0.12	0.08	0	0.04	0	0	0	0.08	0	0.08	0.04	0	0
c10	0.08	0.08	0	0	0.04	0	0	0	0.08	0.04	0.08	0.08	0.08	0	0	0	0	0	0	0	0	0.12	0	0
c11	0	0	0	0	0.12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c12	0.8	0	0.8	0.68	0.76	0	0	0.92	0.88	0	0.64	0.88	0.8	0	0	0.76	0.76	0	0.84	0.8	0.64	0	0	0.8
c13	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
c14	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
c15	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
c16	0.04	0	0	0.04	0.08	0	0.04	0	0.56	0.08	0.68	1	0.56	0.08	0.08	0.04	0	0	0.04	0.08	0.12	0.04	0	0
c17	0.32	0.48	0.12	0.36	0.32	0.2	0.4	0.28	0.32	0.24	0.4	0.68	0.44	0.36	0.44	0.24	0.4	0.24	0.36	0.32	0.44	0.24	0.28	0.32
c18	0.8	1	0.96	0.96	0.92	1	0.96	0.92	0.96	0.88	0.8	0.92	1	0.96	1	0.92	0.96	1	0.96	0.96	0.96	0.88	1	0.96

Algoritmo genético (GA)

	Reglas de Deb (deb)								**ϵ constrained method (ϵcr)**								**Stochastic ranking (sr)**							
	SIN	RI	OPP	CLS	ORT	SAW	CLE	FS	SIN	RI	OPP	CLS	ORT	SAW	CLE	FS	SIN	RI	OPP	CLS	ORT	SAW	CLE	FS
c01	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
c02	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c03	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c04	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c05	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c06	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c07	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
c08	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
c09	0.08	0	0.08	0.04	0.04	0.04	0.12	0.12	0	0	0	0.12	0.04	0.08	0.04	0.04	0.12	0	0	0.04	0.04	0.08	0.2	0.08
c10	0	0	0	0.08	0.08	0.04	0.12	0.04	0	0.04	0.12	0	0.08	0.04	0.08	0.04	0	0.12	0.12	0.04	0.04	0	0.08	0.08
c11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c13	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
c14	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
c15	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
c16	0.72	0.28	0.44	0.64	0.76	0.28	0.52	0.56	0.56	0.32	0.56	0.68	0.6	0.36	0.56	0.44	0.48	0.6	0.48	0.64	0.6	0.52	0.64	0.6
c17	0.64	0.52	0.8	0.64	0.72	0.52	0.8	0.8	0.88	0.68	0.76	0.8	0.88	0.64	0.8	0.8	0.84	0.52	0.76	0.8	0.8	0.68	0.72	0.88
c18	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Tabla B.2: Tasa de factibilidad (FR) para cada problema del CEC 2010 en 30D.

Evolución diferencial (DE)

	Reglas de Deb (deb)								**ϵ constrained method (ϵcr)**								**Stochastic ranking (sr)**							
	SIN	RI	OPP	CLS	ORT	SAW	CLE	FS	SIN	RI	OPP	CLS	ORT	SAW	CLE	FS	SIN	RI	OPP	CLS	ORT	SAW	CLE	FS
c01	0.64	0.76	0.68	0.56	0.64	0	0	0	0.64	0.56	0.48	0.64	0.72	0	0	0	0.6	0.48	0.52	0.68	0.64	0.04	0	0
c02	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c03	1	0.6	1	1	1	0	0	0.28	0.96	0.68	0.92	1	1	0	0	1	1	0.16	0.96	1	1	0	0	0.16
c04	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c05	0	0	0	0	0	0	0	0	0.16	0	0.12	0.32	0.28	0	0	0	0	0	0	0	0	0	0	0
c06	0	0	0	0	0	0	0	0	0.04	0	0	0.04	0	0	0	0	0	0	0	0	0	0	0	0
c07	1	0	1	1	1	0	0	0.88	1	0	1	1	1	0	0	0.92	1	0	1	1	0.92	0	0	0.96
c08	1	0	0.96	1	1	0	0	0.72	1	0	0.96	1	1	0	0	0.72	0.92	0	0.52	0.96	0.92	0	0	0.56
c09	0	0	0	0	0	0	0	0	0.76	0	0.48	0.64	0.68	0	0	0.04	0	0	0	0	0	0	0	0
c10	0	0	0	0	0	0	0	0	0.04	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c11	1	0	1	1	1	0	0	0.92	0.96	0.04	0.96	1	1	0	0	0.92	1	0	1	1	1	0	0	0.88
c12	0	0	0	0	0	0	0	0	0.08	0	0.12	0.04	0.04	0	0	0	0	0	0	0	0	0	0	0
c13	0.96	0.68	1	0.96	1	0	0	0	1	0.24	0.88	0.96	1	0	0	0	1	0.72	1	1	1	0	0	0
c14	0	0	0	0	0	0	0	0	0.96	0	0.8	0.92	0.88	0	0	0.6	0	0	0	0	0	0	0	0
c15	0	0	0	0	0	0	0	0	0.04	0.2	0	0.04	0.04	0	0	0.08	0	0	0	0	0	0	0	0
c16	0	0	0	0	0	0	0	0	0.04	0	0.04	0.6	0.12	0	0	0	0	0	0	0	0	0	0	0
c17	0	0	0	0	0	0	0	0	0.28	0	0.32	0.36	0.36	0	0	0.08	0	0	0	0	0	0	0	0
c18	0	0	0	0	0	0	0	0	0.72	0	0.48	0.52	0.56	0	0	0.12	0	0	0	0	0	0	0	0

Algoritmo genético (GA)

	Reglas de Deb (deb)								**ϵ constrained method (ϵcr)**								**Stochastic ranking (sr)**							
	SIN	RI	OPP	CLS	ORT	SAW	CLE	FS	SIN	RI	OPP	CLS	ORT	SAW	CLE	FS	SIN	RI	OPP	CLS	ORT	SAW	CLE	FS
c01	0.32	0	0	0.32	0.28	0	0	0	0.36	0	0.04	0.16	0.12	0	0	0	0.4	0	0	0.36	0.24	0	0	0
c02	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c03	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c04	0	0	0	0	0	0	0	0	0.04	0	0	0.08	0	0	0	0	0	0	0	0	0	0	0	0
c05	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c06	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c07	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c08	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c09	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c16	0	0	0	0	0	0	0	0	0.32	0	0.04	0.72	0.52	0	0	0.04	0	0	0	0	0	0	0	0
c17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tabla B.3: Tasa de éxito (SR) para cada problema del CEC 2010 en 10D.

Evolución diferencial (DE)

	Reglas de Deb (deb)								**ϵ constrained method (ϵcr)**								**Stochastic ranking (sr)**							
	SIN	RI	OPP	CLS	ORT	SAW	CLE	FS	SIN	RI	OPP	CLS	ORT	SAW	CLE	FS	SIN	RI	OPP	CLS	ORT	SAW	CLE	FS
c01	0.04	0	0	0.08	0	0	0	0	0.12	0	0	0.04	0	0	0	0	0.08	0	0.08	0.04	0.08	0	0	0
c02	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c03	0	0	0	0	0	0	0	0	0	0	0	0.04	0.04	0	0	0	0	0	0	0	0	0	0	0
c04	0	0	0	0	0	0	0	0	0	0	0.04	0	0	0	0	0	0	0	0	0	0	0	0	0
c05	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c06	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c07	0	0	0	0	0.04	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c08	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c09	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c12	0.68	0	0.48	0.44	0.72	0	0	0	0.52	0	0.24	0.64	0.72	0	0	0	0.4	0	0.28	0.56	0.64	0	0	0
c13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c16	0	0	0	0	0	0	0	0	0.48	0	0	0.84	0.36	0	0	0	0	0	0	0	0	0	0	0
c17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Algoritmo genético (GA)

	Reglas de Deb (deb)								**ϵ constrained method (ϵcr)**								**Stochastic ranking (sr)**							
	SIN	RI	OPP	CLS	ORT	SAW	CLE	FS	SIN	RI	OPP	CLS	ORT	SAW	CLE	FS	SIN	RI	OPP	CLS	ORT	SAW	CLE	FS
c01	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c02	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c03	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c04	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c05	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c06	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c07	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c08	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c09	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tabla B.4: Tasa de éxito (SR) para cada problema del CEC 2010 en 30D.

B.2. Gráficas de diversidad experimento 2

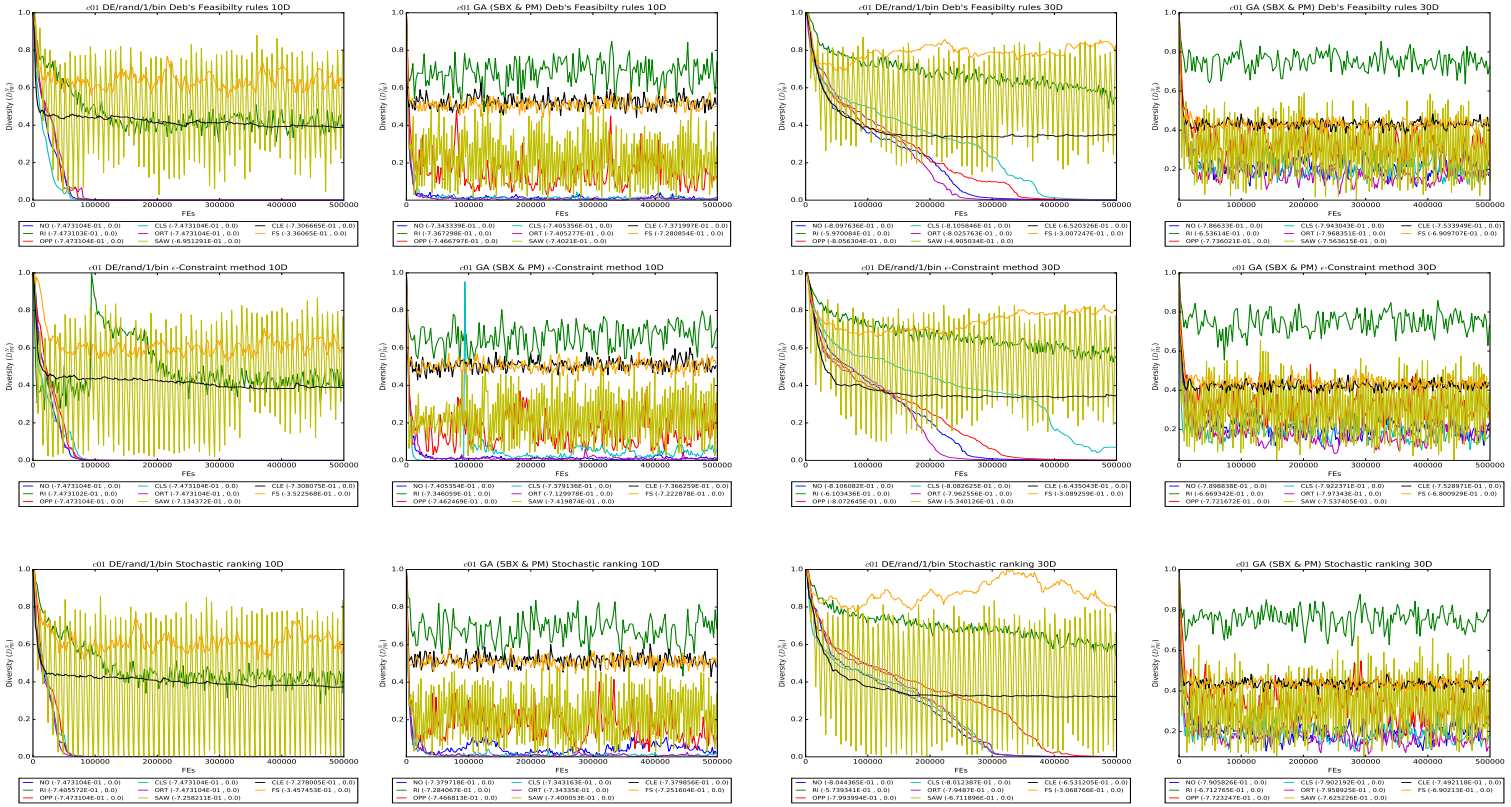

Figura B.1: Gráficas de diversidad para el problema $c01$ en $10D$ y $30D$.

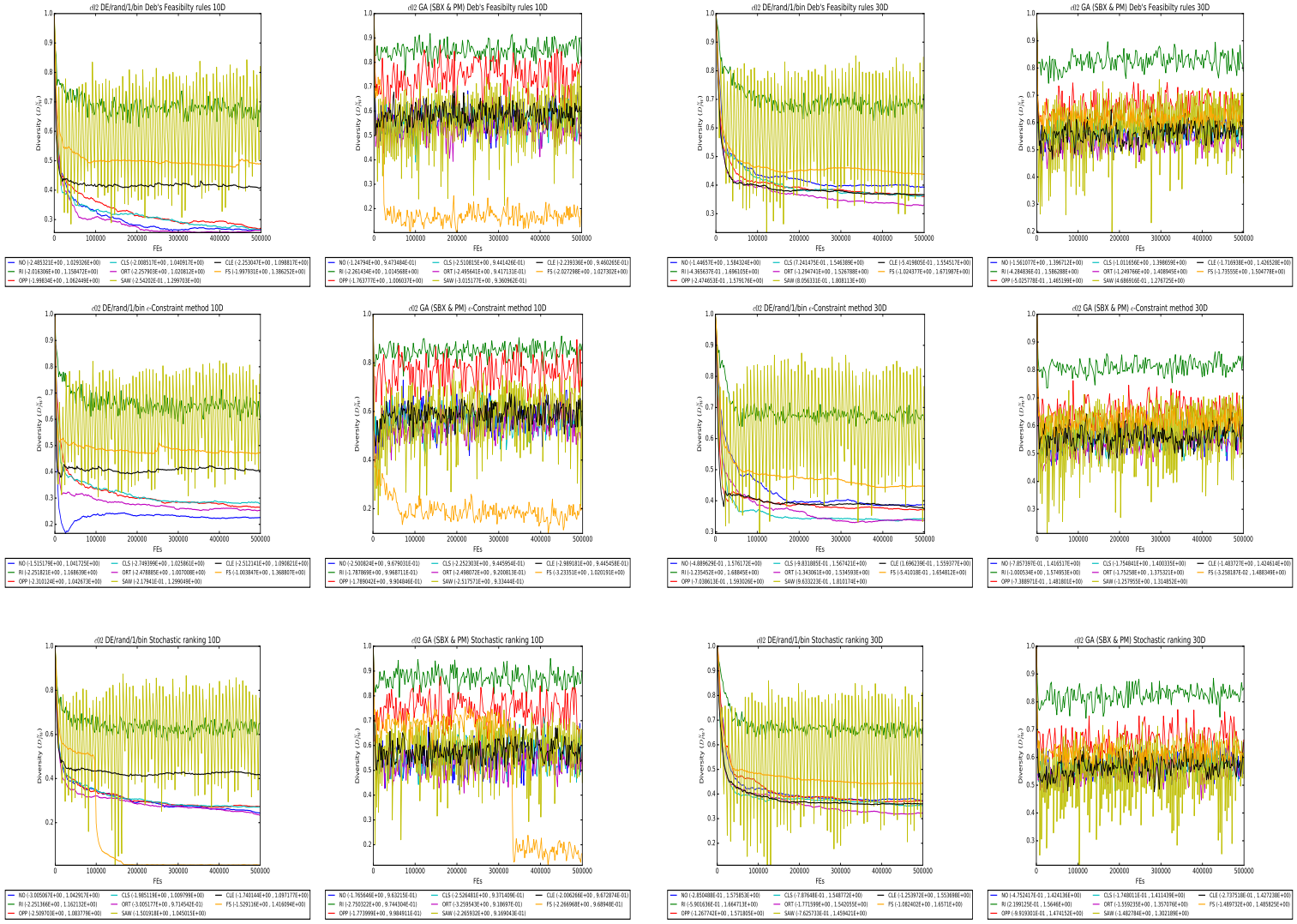

Figura B.2: Gráficas de diversidad para el problema $c02$ en $10D$ y $30D$.

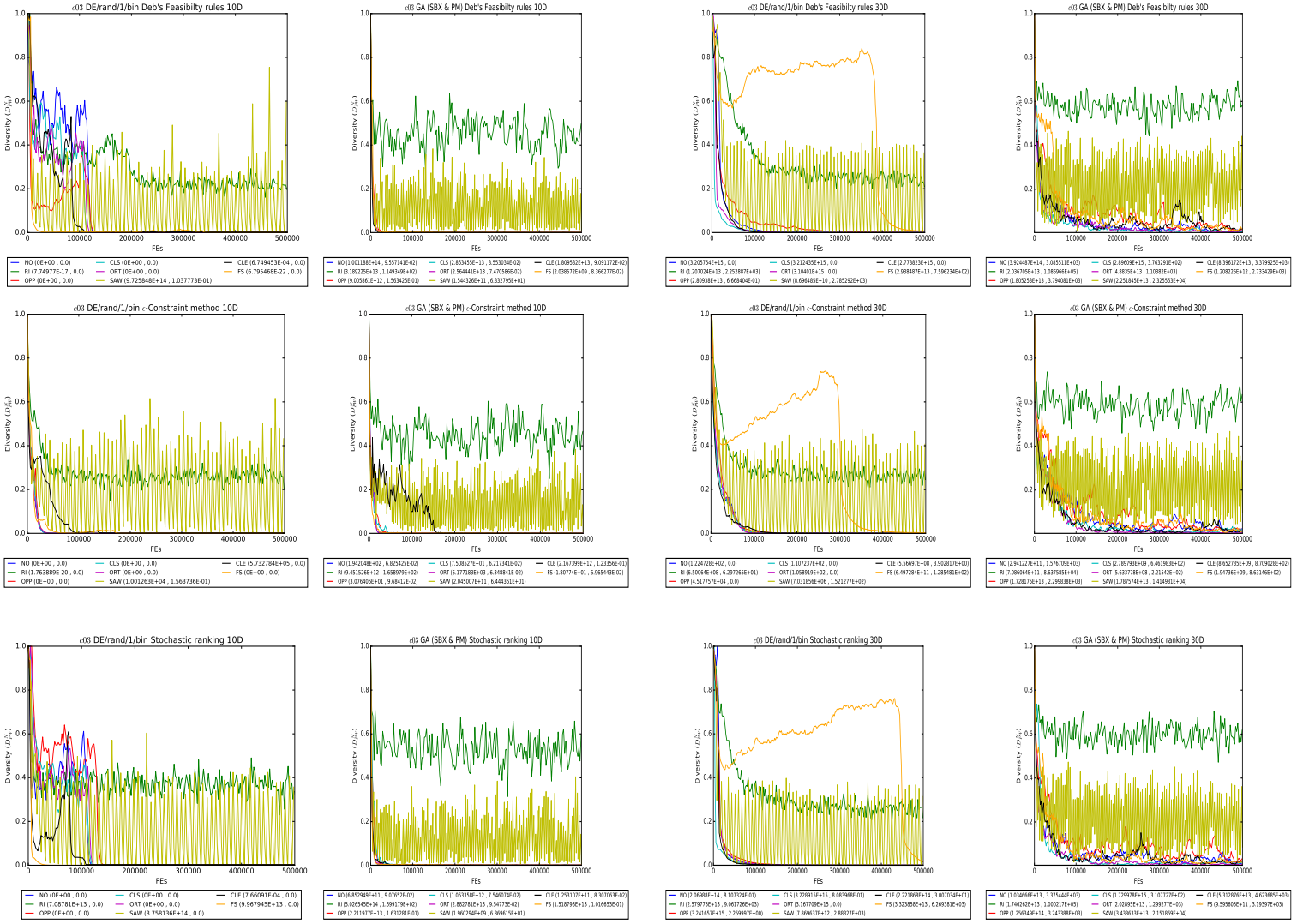

Figura B.3: Gráficas de diversidad para el problema $c03$ en $10D$ y $30D$.

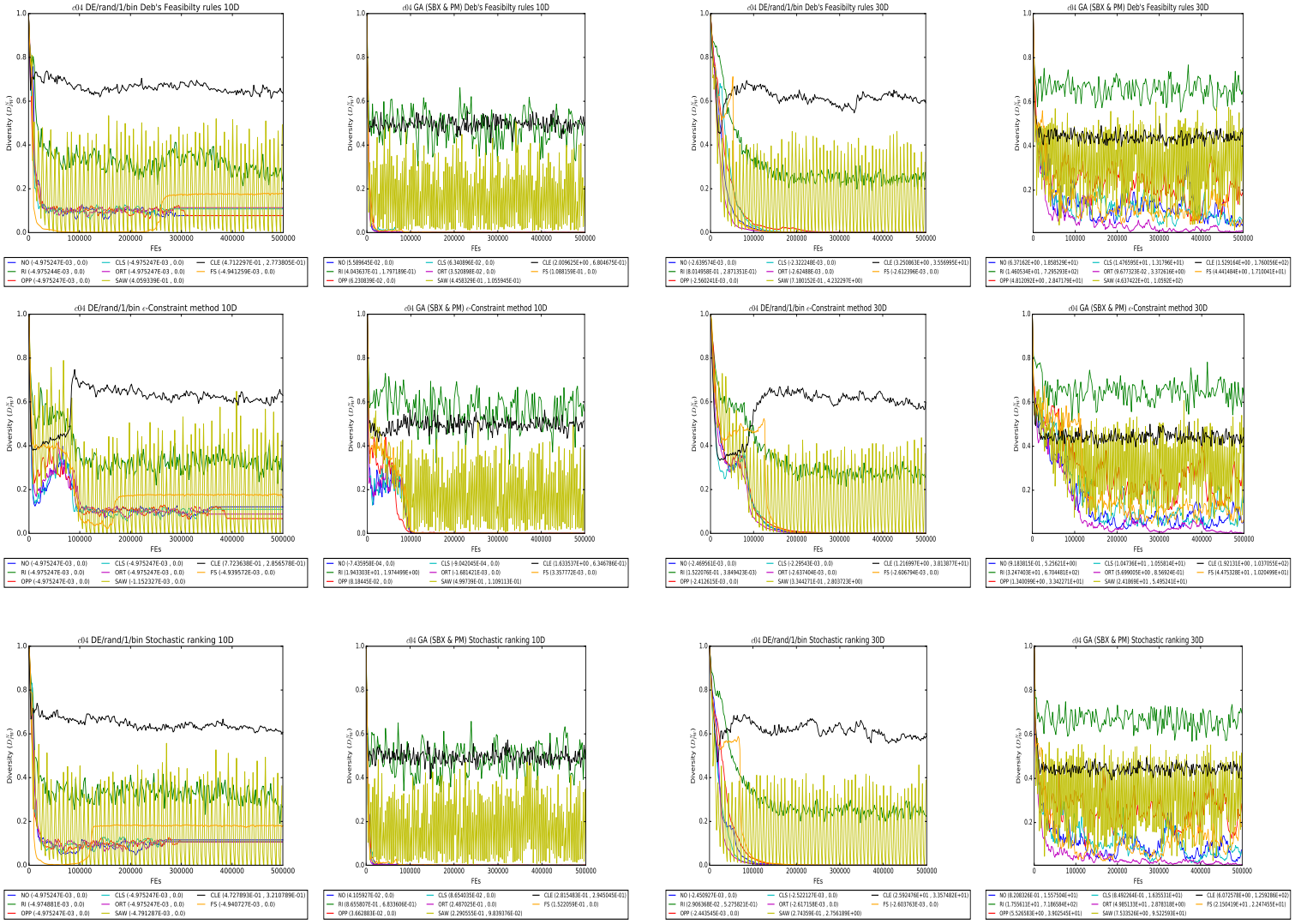

Figura B.4: Gráficas de diversidad para el problema $c04$ en $10D$ y $30D$.

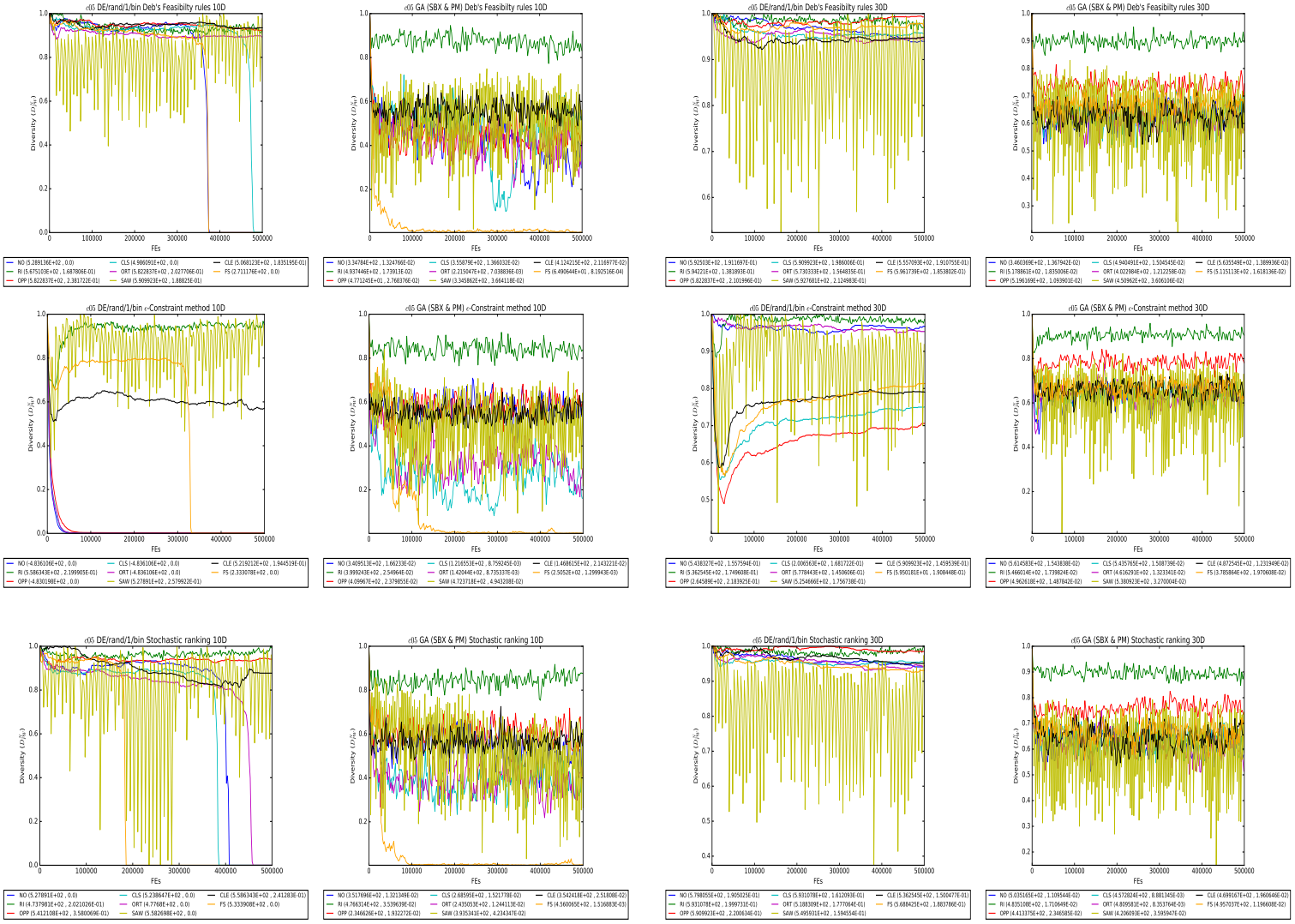

Figura B.5: Gráficas de diversidad para el problema $c05$ en $10D$ y $30D$.

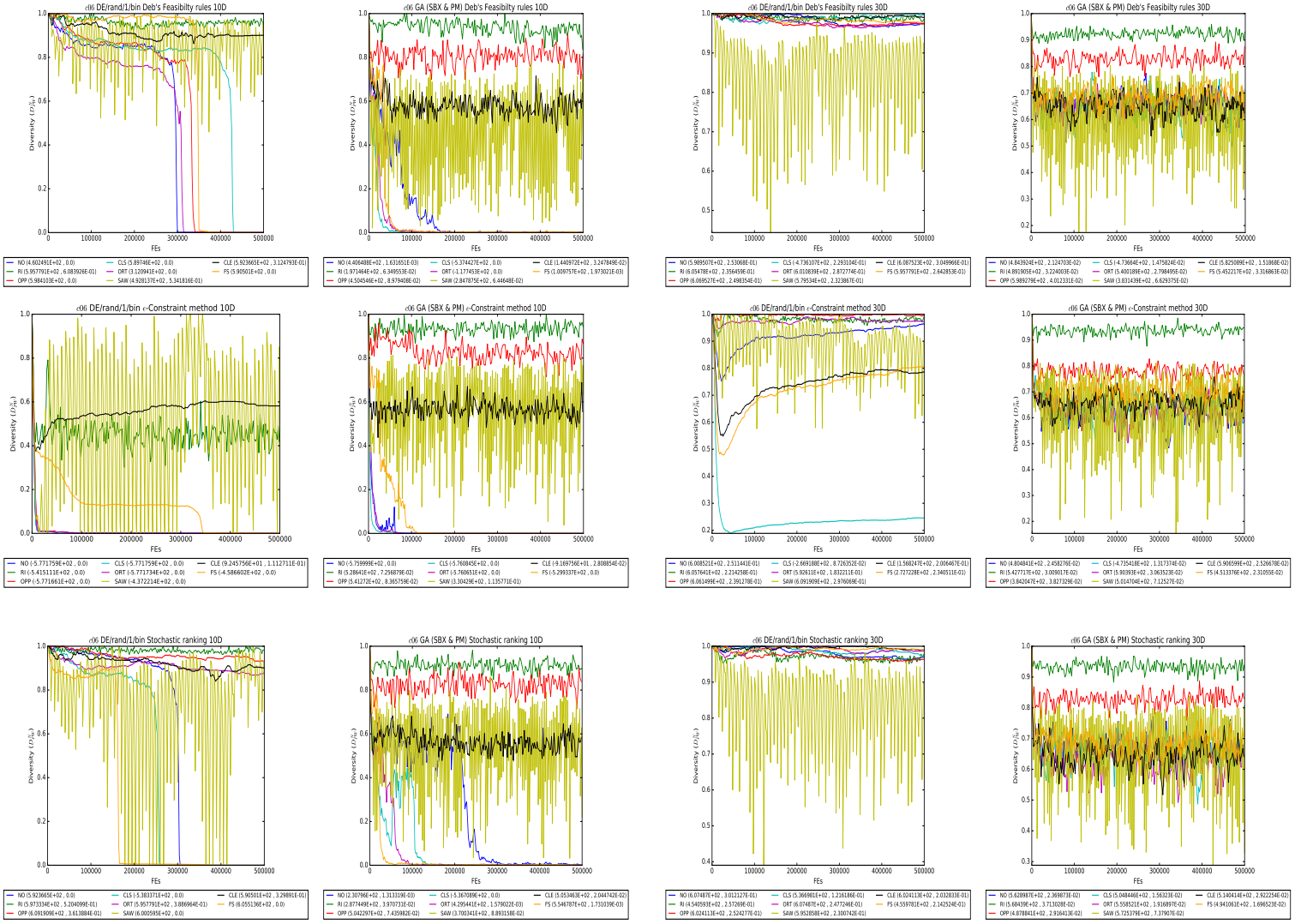

Figura B.6: Gráficas de diversidad para el problema $c06$ en $10D$ y $30D$.

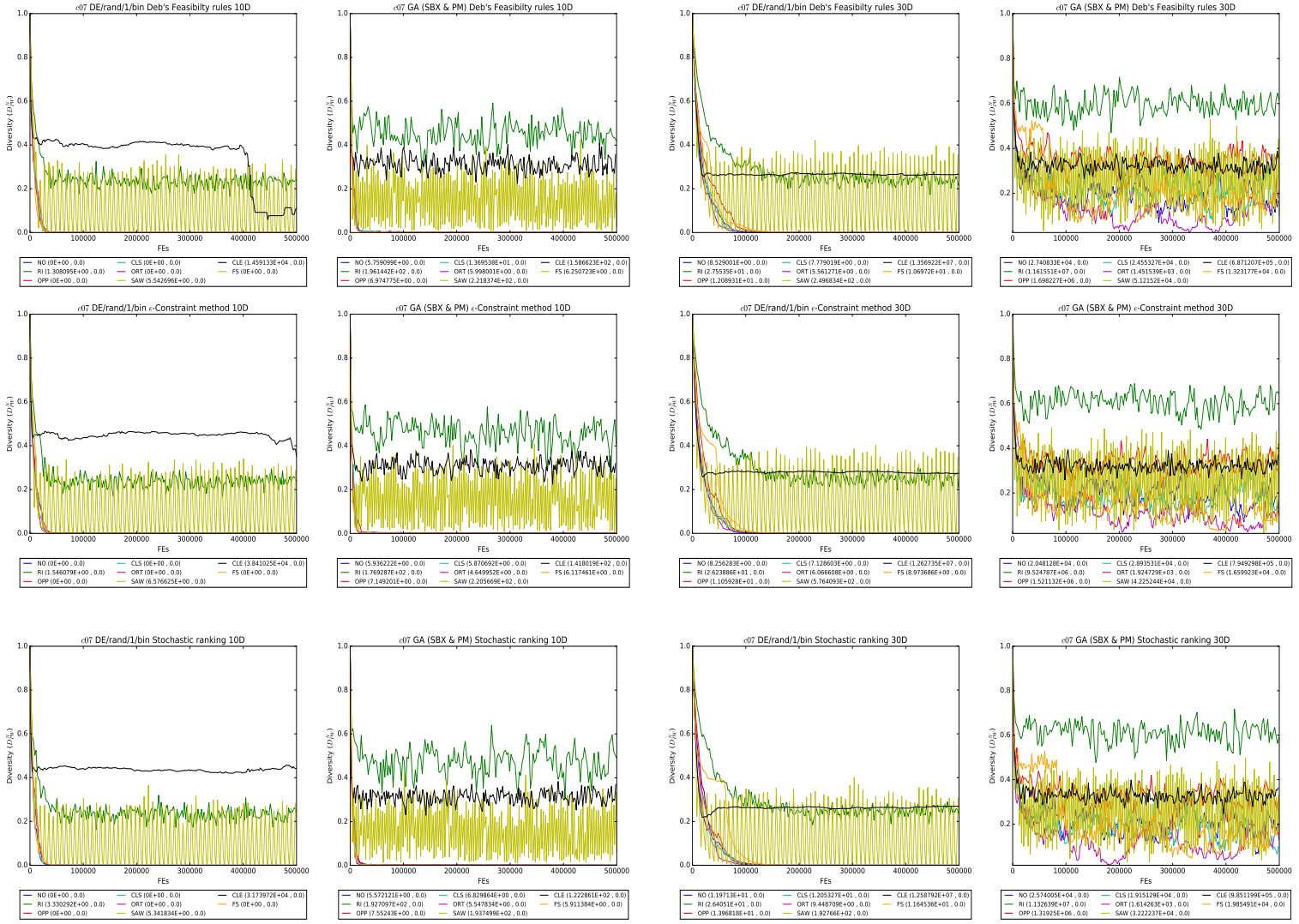

Figura B.7: Gráficas de diversidad para el problema $c07$ en $10D$ y $30D$.

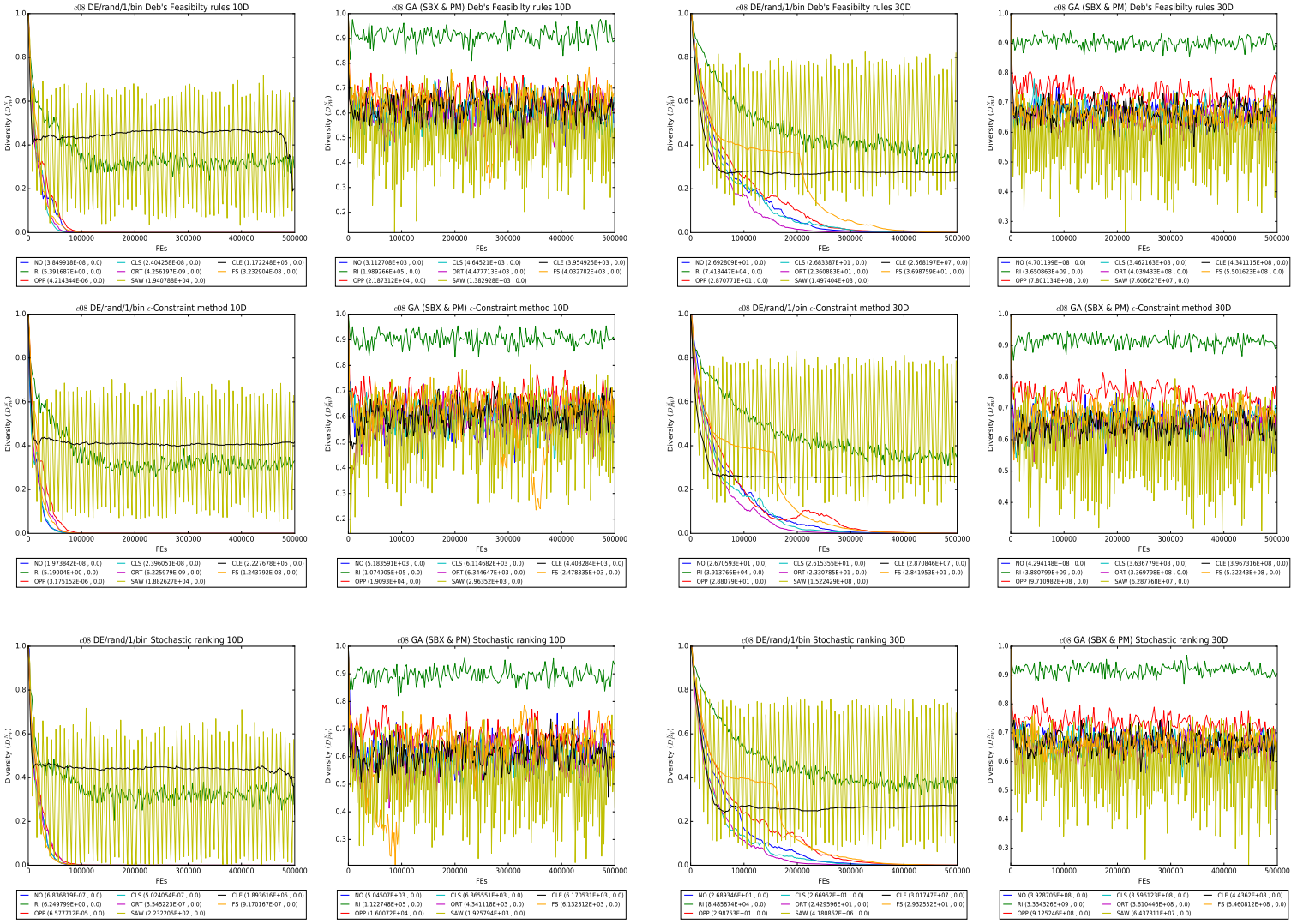

Figura B.8: Gráficas de diversidad para el problema $c08$ en $10D$ y $30D$.

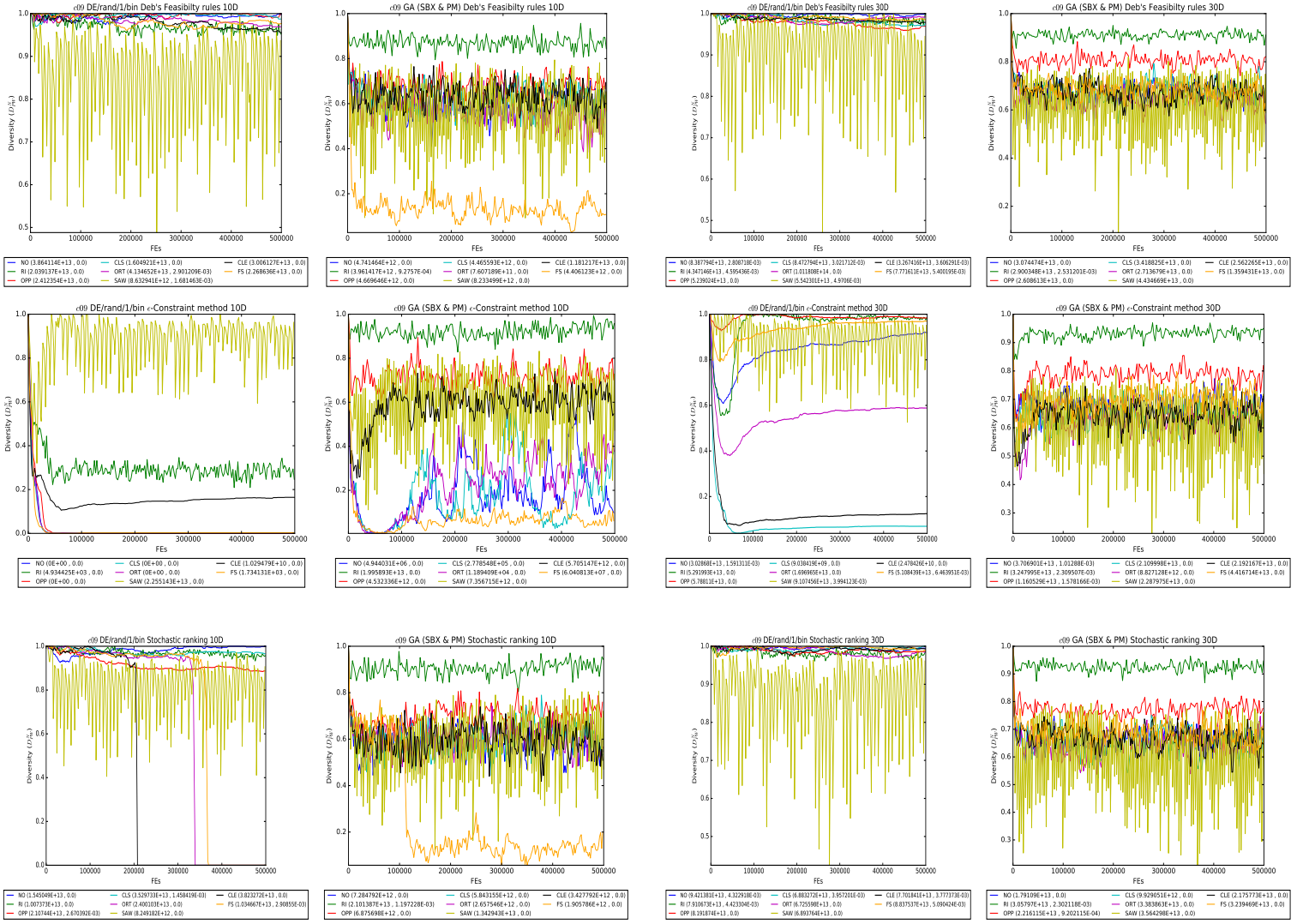

Figura B.9: Gráficas de diversidad para el problema $c09$ en $10D$ y $30D$.

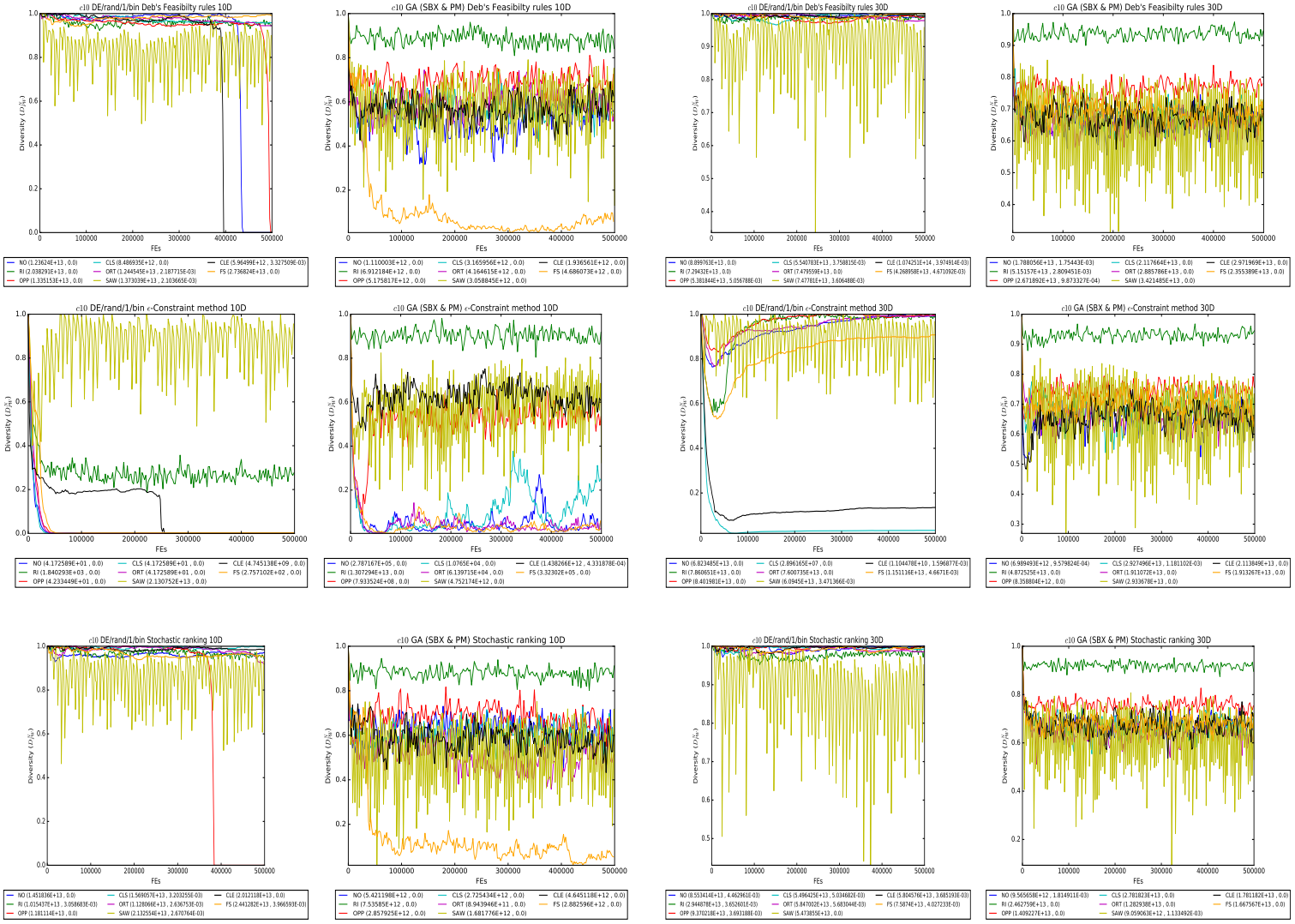

Figura B.10: Gráficas de diversidad para el problema $c10$ en $10D$ y $30D$.

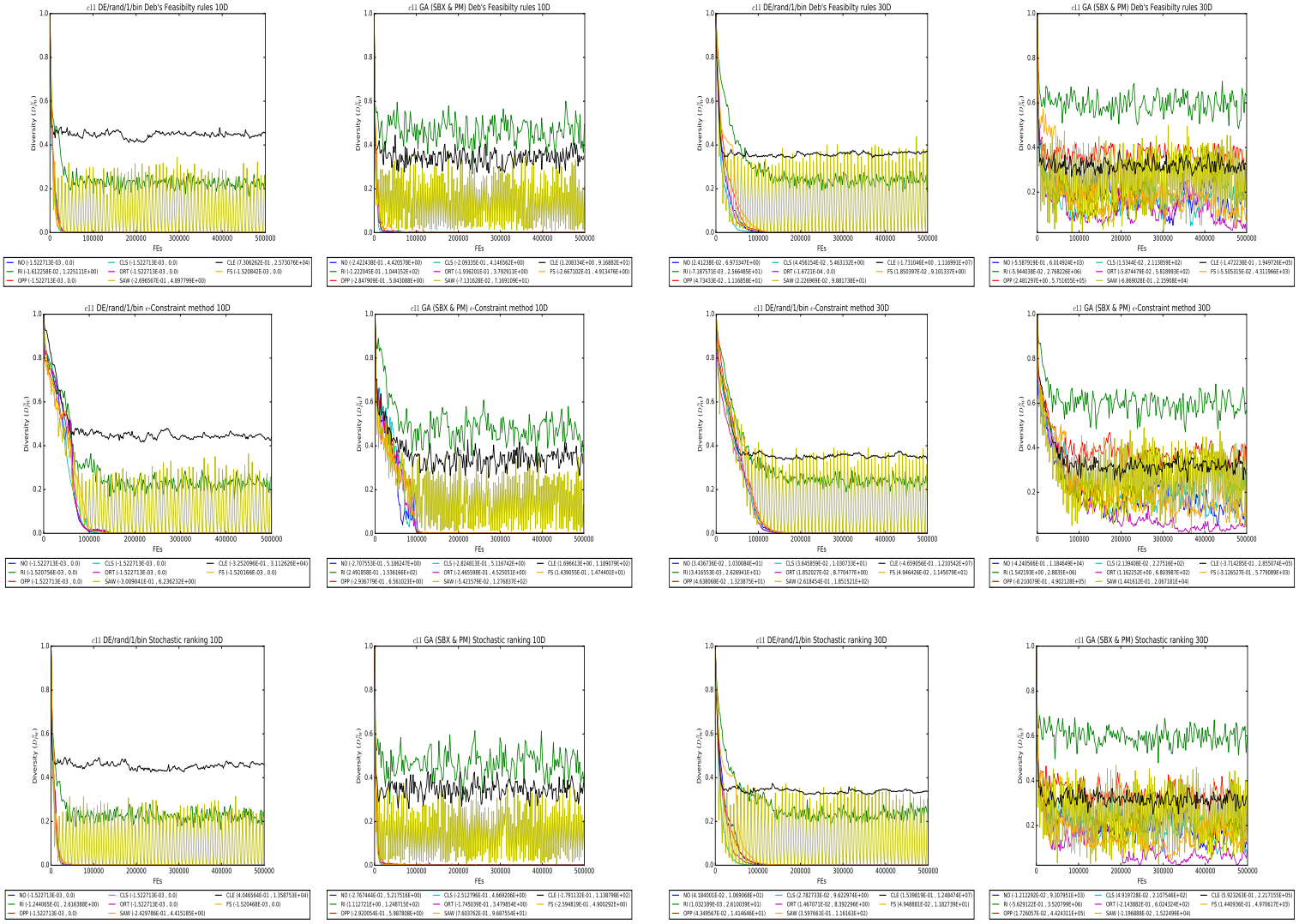

Figura B.11: Gráficas de diversidad para el problema $c11$ en $10D$ y $30D$.

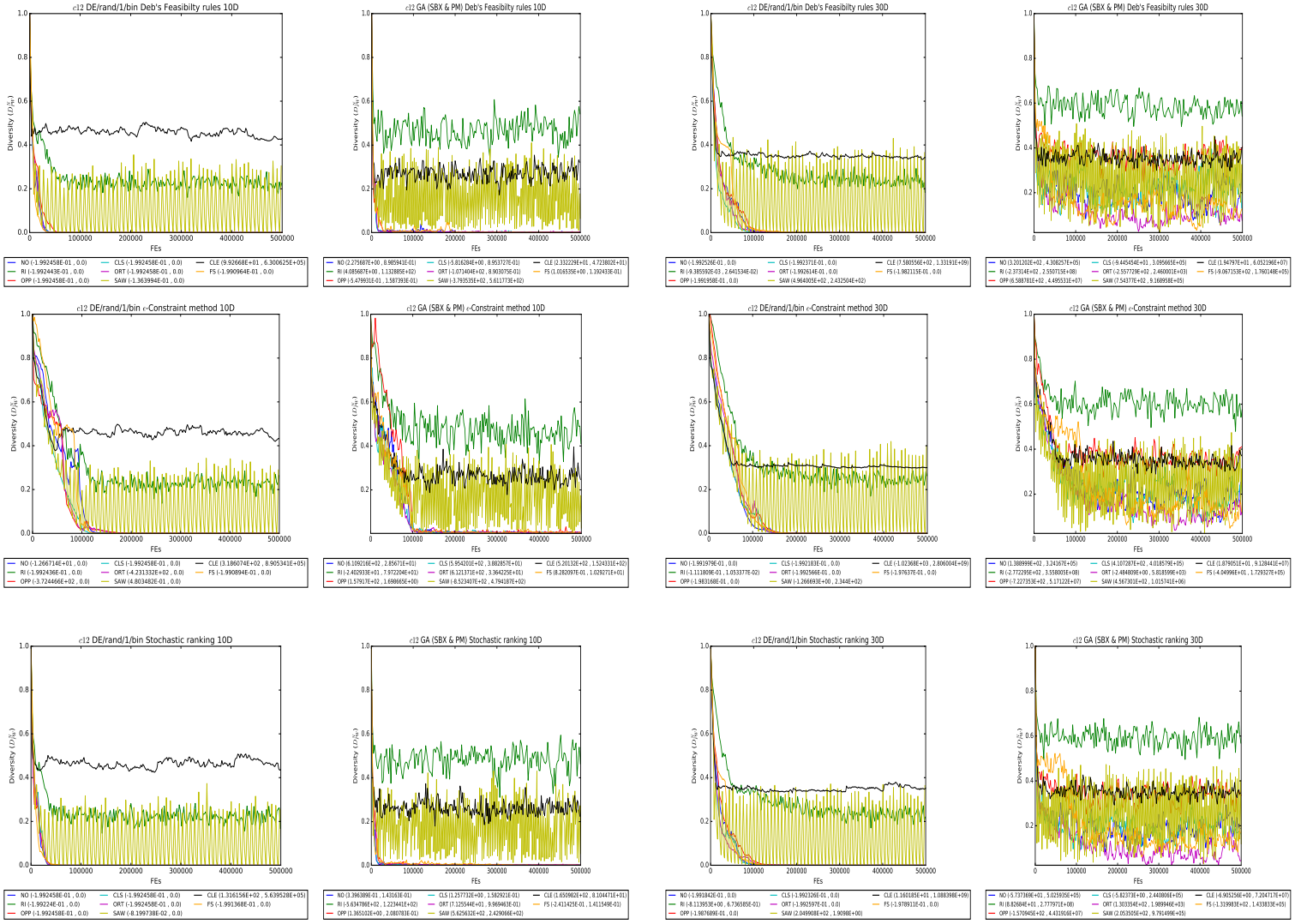

Figura B.12: Gráficas de diversidad para el problema $c12$ en $10D$ y $30D$.

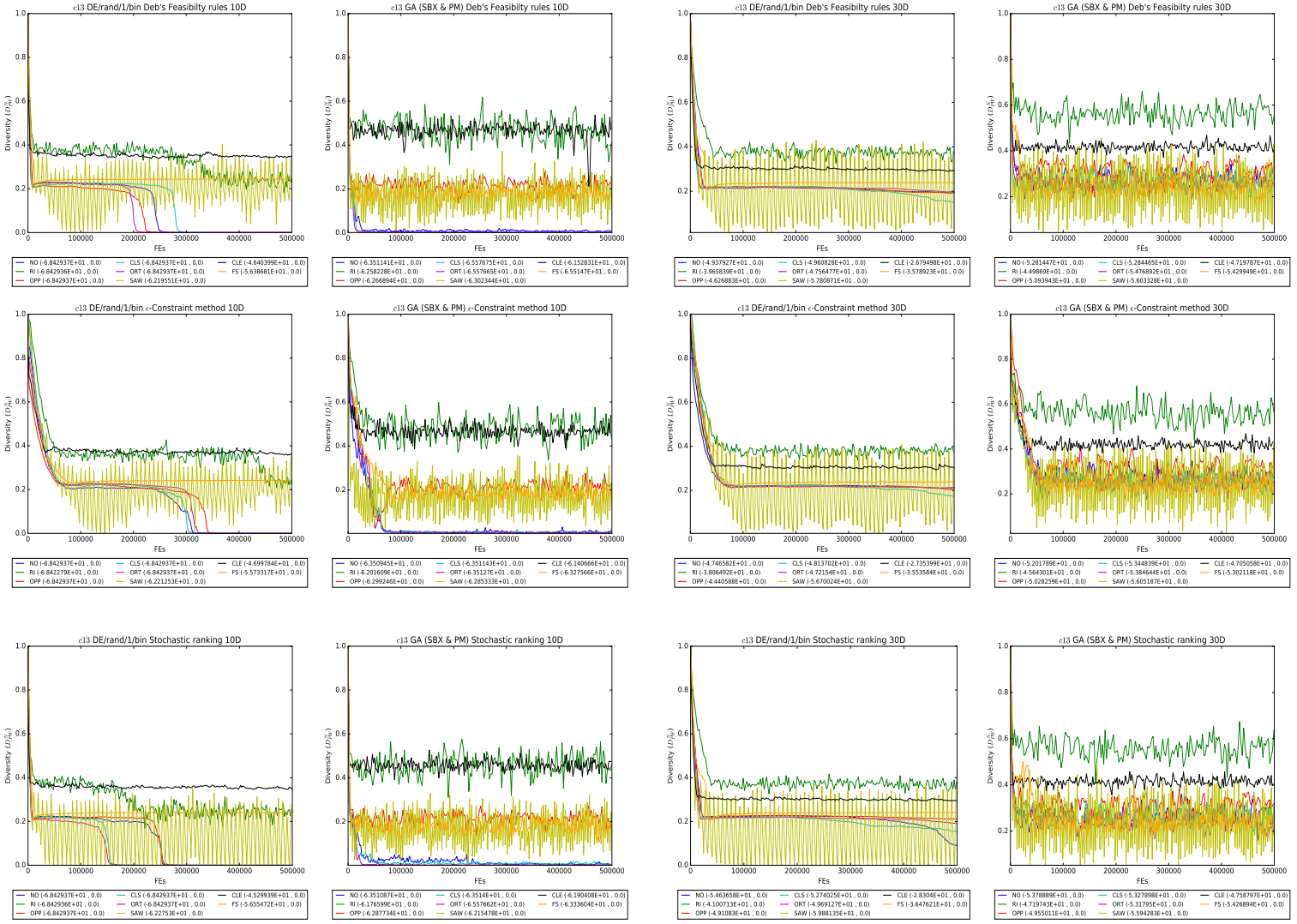

Figura B.13: Gráficas de diversidad para el problema $c13$ en $10D$ y $30D$.

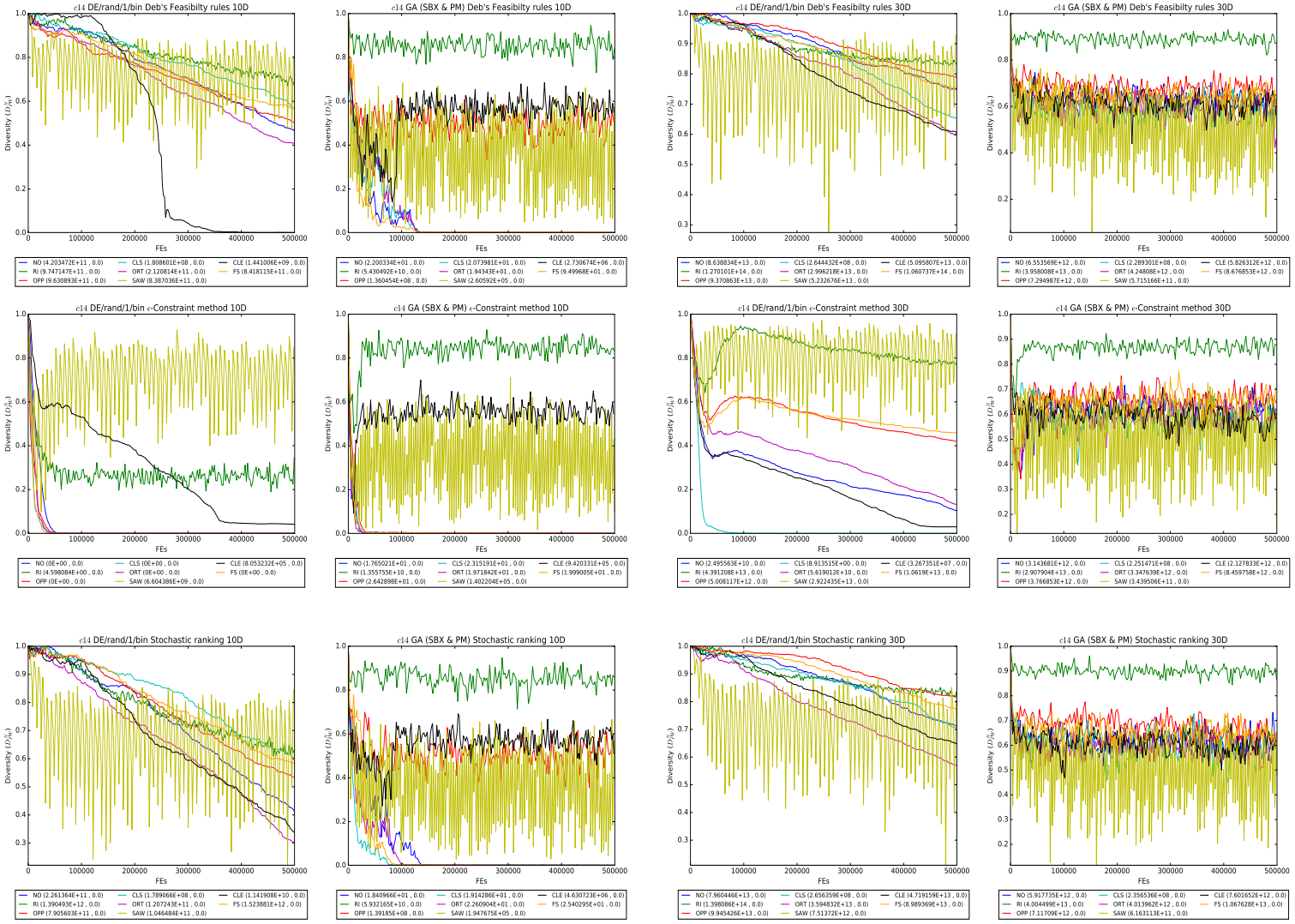

Figura B.14: Gráficas de diversidad para el problema $c14$ en $10D$ y $30D$.

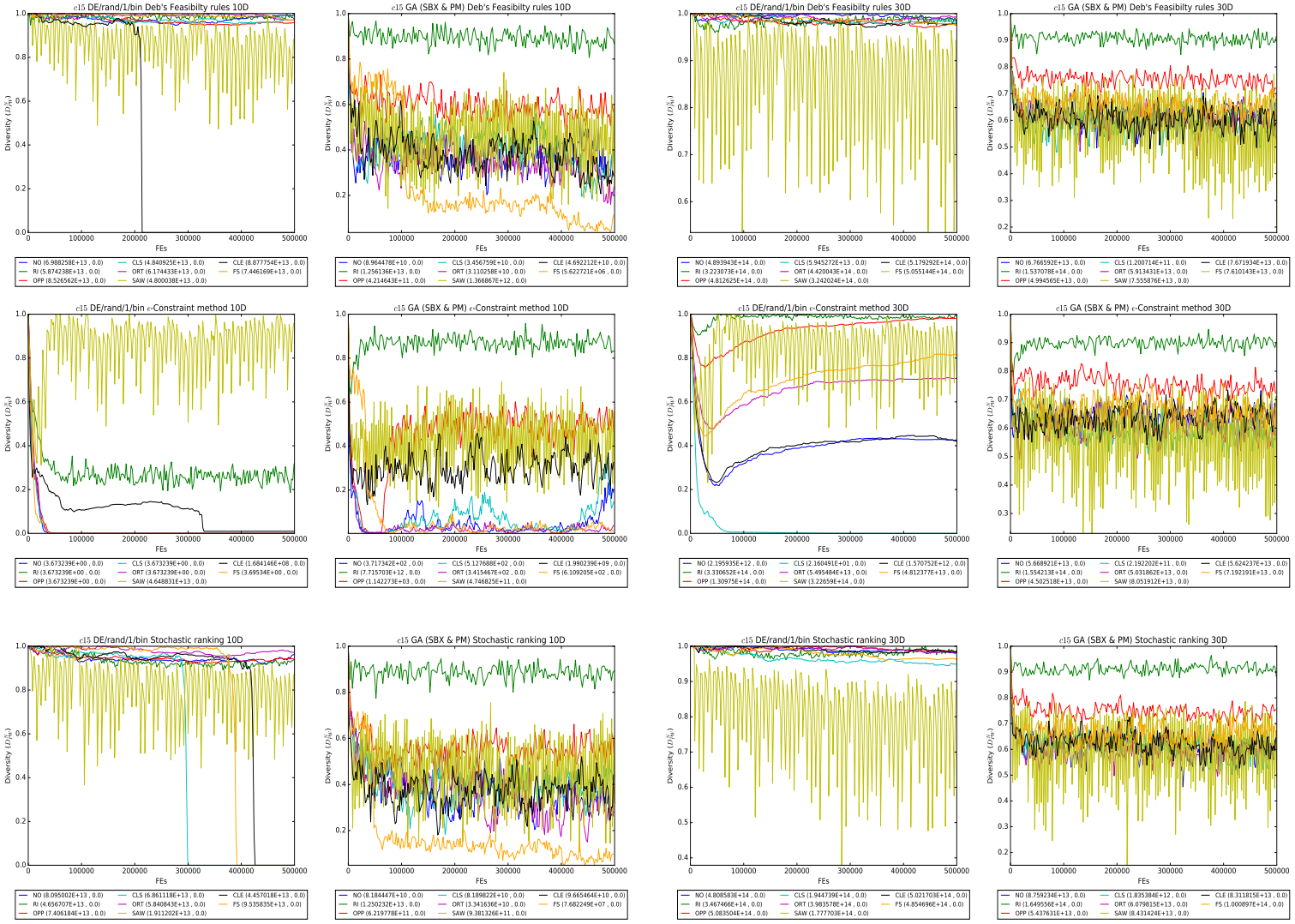

Figura B.15: Gráficas de diversidad para el problema $c15$ en $10D$ y $30D$.

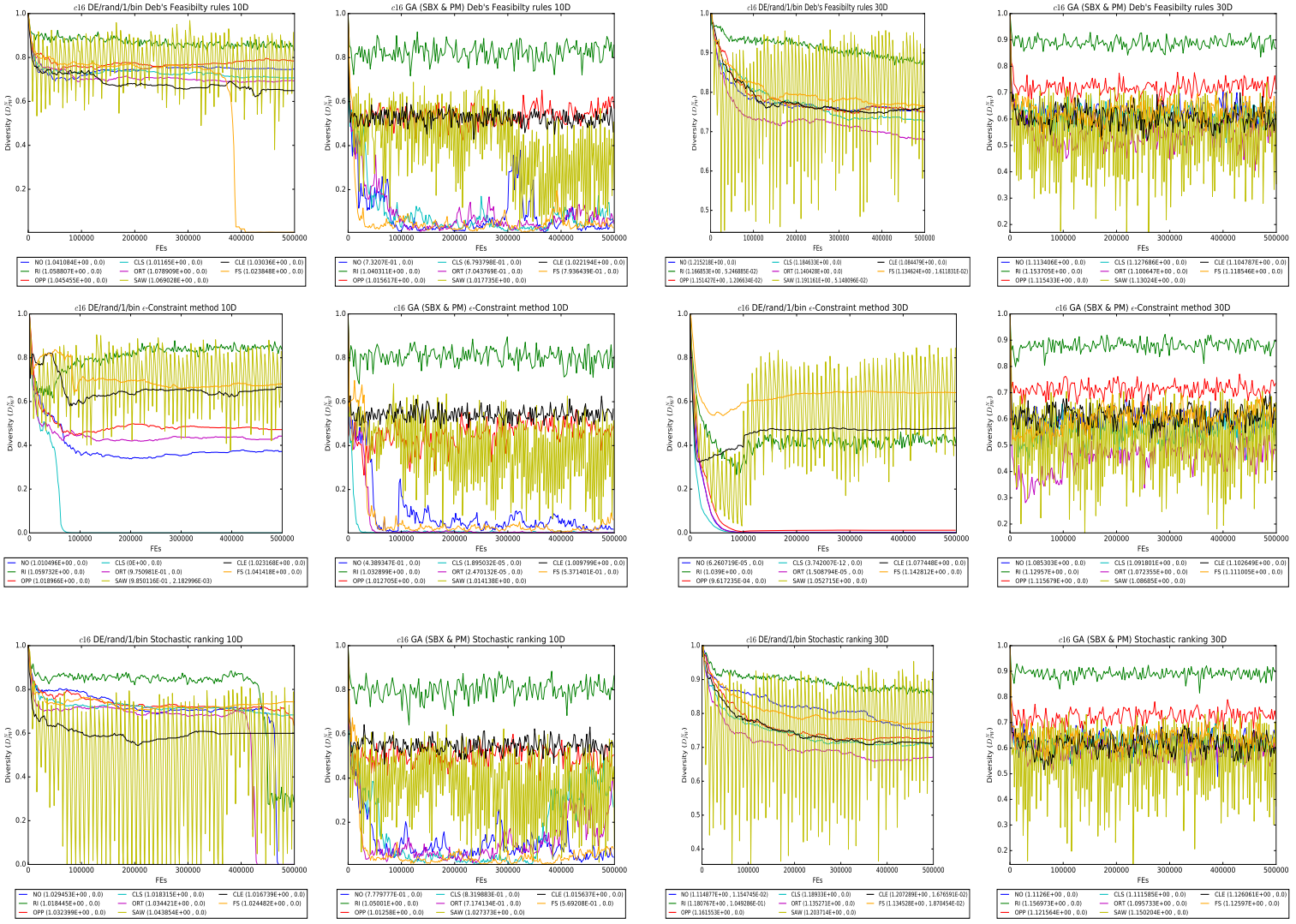

Figura B.16: Gráficas de diversidad para el problema $c16$ en $10D$ y $30D$.

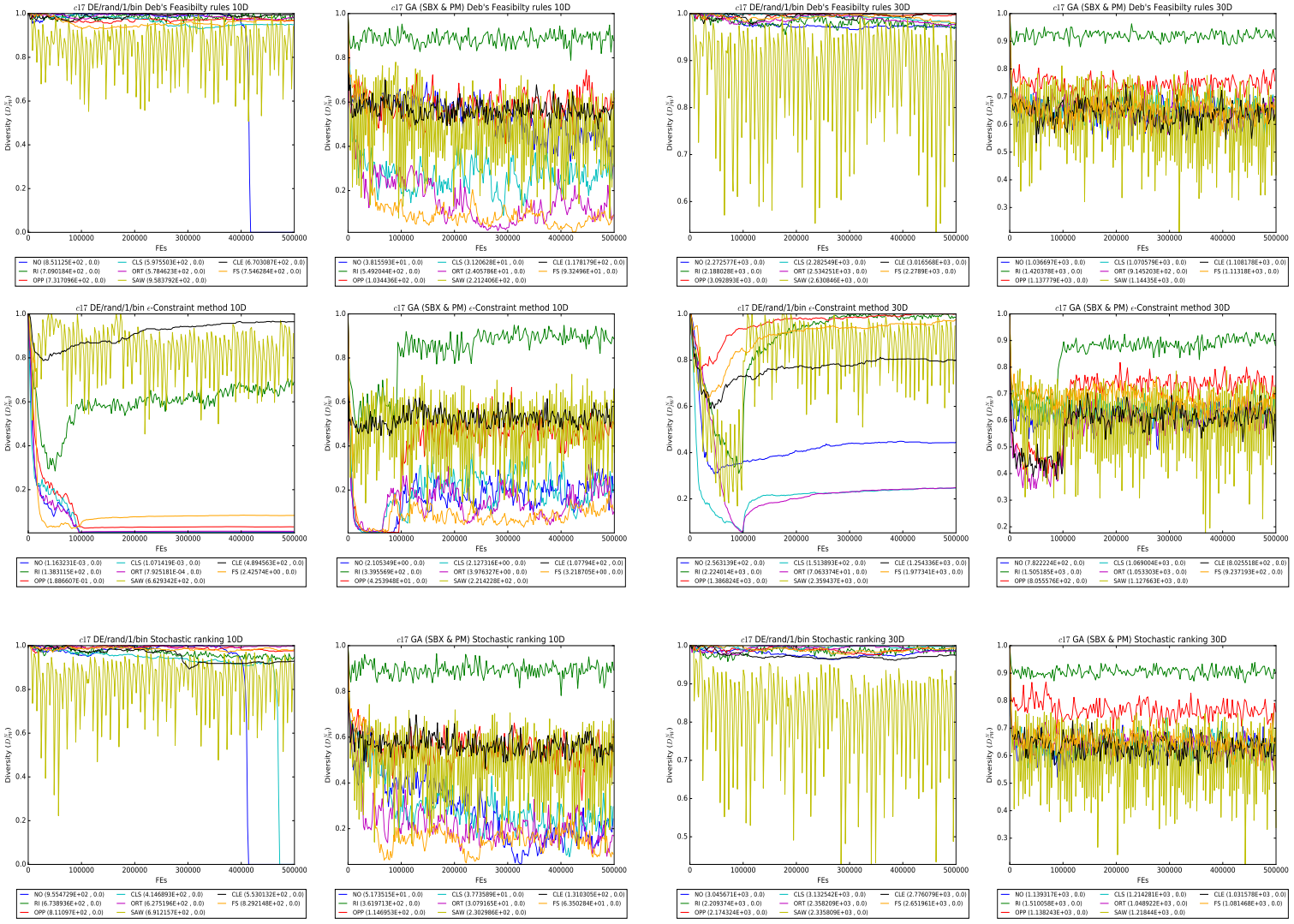

Figura B.17: Gráficas de diversidad para el problema $c17$ en $10D$ y $30D$.

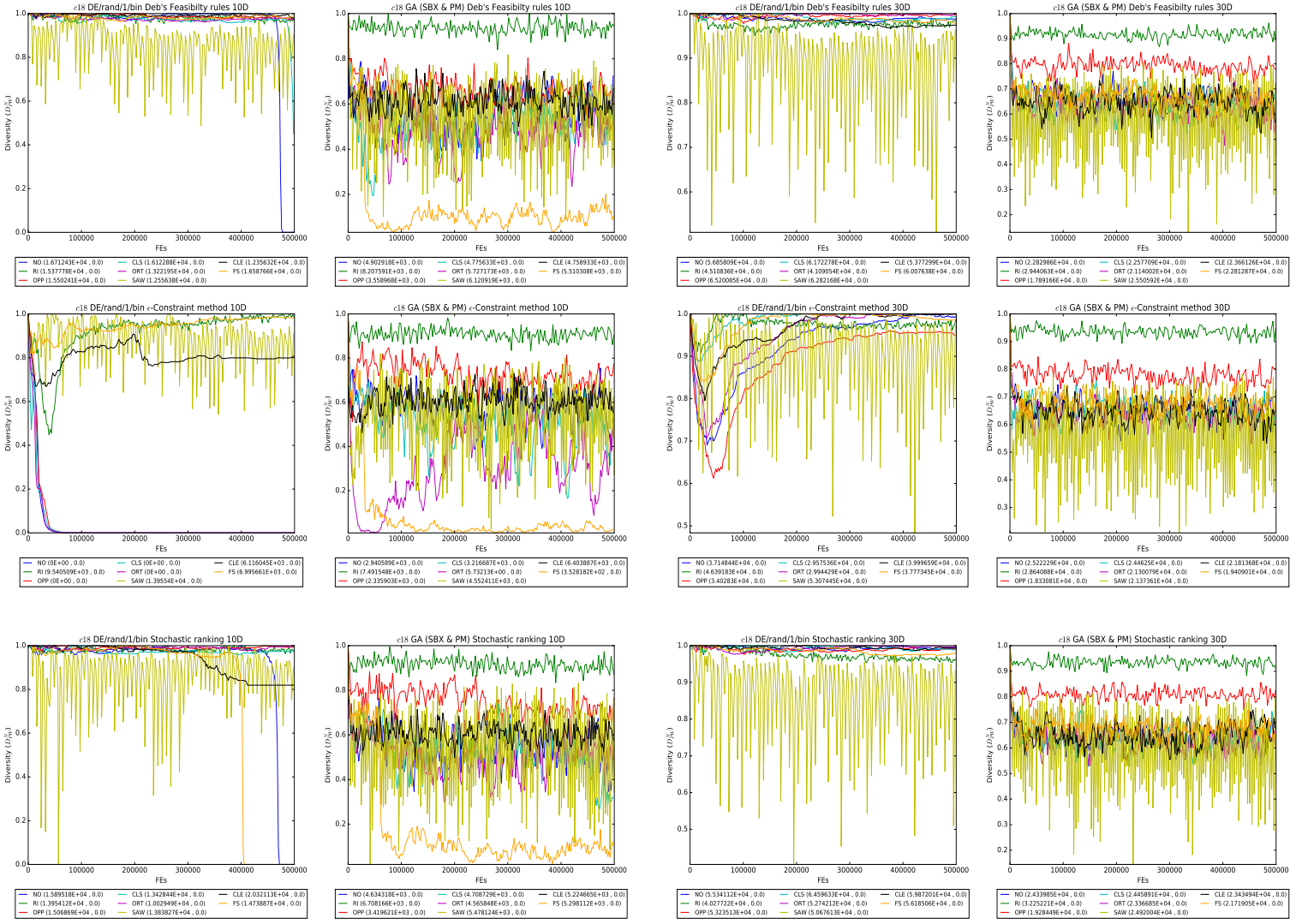

Figura B.18: Gráficas de diversidad para el problema $c18$ en $10D$ y $30D$.