Facultad de Matemática y Computación. Universidad de La Habana.
Métodos de Optimización.

Repositorio del Proyecto en GitHub

Evaluación de Algoritmos de Optimización en la Giunta Function.

Daniel Machado Pérez - daniel.machado.0206@gmail.com
September 4, 2024

Resumen

Este trabajo presenta un estudio comparativo de varios algoritmos de optimización aplicados a la función de Giunta, una función no lineal ampliamente utilizada en la evaluación del rendimiento de métodos de optimización. Se implementaron y probaron tanto métodos de optimización clásicos como avanzados, incluyendo el método de descenso máximo, métodos cuasi-Newton, algoritmos genéticos, optimización por enjambre de partículas (PSO), evolución diferencial, y otros métodos basados en la región de confianza y el recocido simulado. El objetivo principal fue evaluar la precisión de los algoritmos para aproximarse al mínimo global conocido de la función de Giunta, así como analizar el tiempo computacional requerido para alcanzar dicha solución. Los resultados obtenidos permiten identificar las fortalezas y debilidades de cada enfoque, proporcionando una guía sobre qué algoritmos son más adecuados para problemas similares en términos de eficiencia y precisión.

1 Introducción

La función de Giunta es una función matemática utilizada comúnmente en la evaluación de algoritmos de optimización, especialmente aquellos diseñados para abordar problemas multimodales. La función está definida como sigue:

$$f(\mathbf{x}) = 0.6 + \sum_{i=1}^{2} \left[\sin\left(\frac{16}{15}x_i - 1\right) + \sin^2\left(\frac{16}{15}x_i - 1\right) + \frac{1}{50}\sin\left(4\left(\frac{16}{15}x_i - 1\right)\right)\right]$$

donde $\mathbf{x} = (x_1, x_2)$ está sujeto a $-1 \leq x_i \leq 1$ para $i = 1, 2$. El mínimo global de esta función se encuentra en $\mathbf{x}^* = (0.45834282, 0.45834282)$, con un valor de $f(\mathbf{x}^*) = 0.060447$.
Esta función presenta varias características que la hacen un reto interesante para los algoritmos de optimización:

- **Continuidad:** La función es continua en todo su dominio, lo que permite la aplicación de una amplia gama de métodos de optimización.
- **Diferenciabilidad:** Es diferenciable, lo que permite el uso de algoritmos que dependen de la derivada, como los métodos de descenso de gradiente.
- **Separabilidad:** La función puede descomponerse en sumas de funciones más simples, lo que podría simplificar la optimización en ciertas circunstancias.
- **Escalabilidad:** Aunque en este trabajo se estudia en un espacio de dos dimensiones, la función puede generalizarse a dimensiones superiores.
- **Multimodalidad:** La presencia de múltiples óptimos locales hace que la búsqueda del mínimo global sea un desafío, especialmente para métodos que tienden a quedarse atrapados en mínimos locales.

Dado que la función de Giunta es multimodal y presenta múltiples óptimos locales, es crucial elegir algoritmos que no solo sean capaces de encontrar un mínimo, sino que también tengan la capacidad de explorar adecuadamente el espacio de búsqueda para evitar caer en mínimos locales. En este estudio, se han seleccionado y probado los siguientes algoritmos:

- **Métodos Clásicos:** Se han implementado métodos como el *descenso máximo*, el *método cuasi-Newton (BFGS)*, y el *método de región de confianza*. Estos métodos son apropiados para funciones diferenciables y proporcionan una buena convergencia local.
- **Algoritmos Evolutivos:** Se incluye la *evolución diferencial* y el *algoritmo genético*, que son conocidos por su capacidad para explorar grandes espacios de búsqueda y su eficacia en la identificación de óptimos globales.

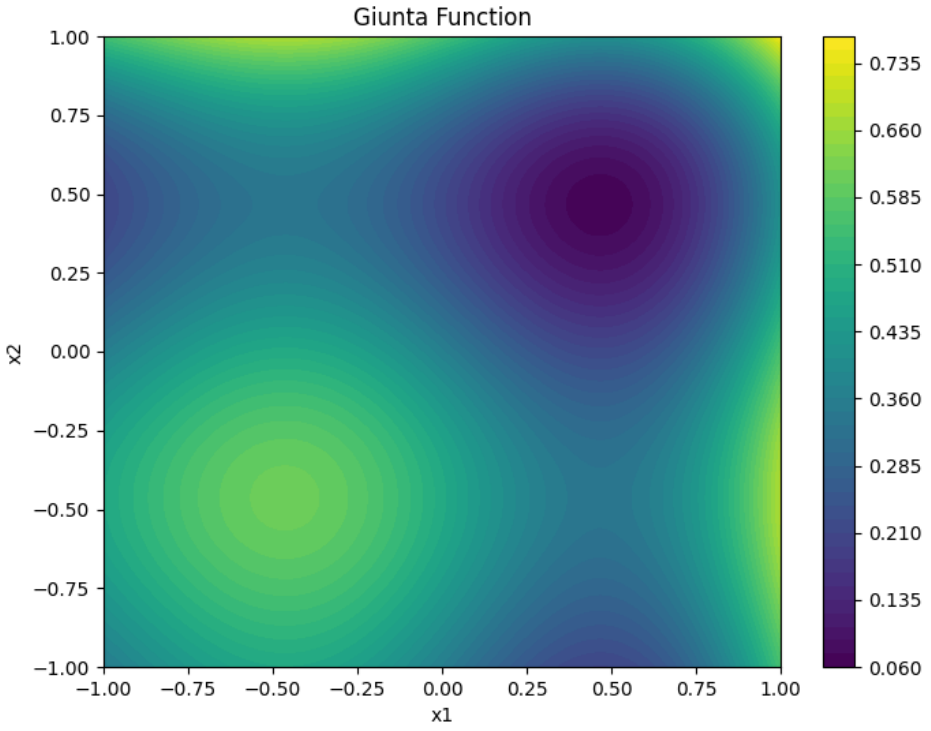

Figure 1: Giunta Function.

- **Optimización Basada en Población:** El *algoritmo de enjambre de partículas (PSO)* es otro enfoque utilizado, el cual es efectivo en la búsqueda global, aprovechando la cooperación entre partículas.
- **Algoritmos de Búsqueda Estocástica:** Finalmente, se ha probado el *recocido simulado* y el *método de salto de cuenca (Basin Hopping)*, que son métodos robustos frente a la multimodalidad, proporcionando una exploración más extensa del espacio de búsqueda.
- **Función `minimize` de `scipy.optimize`:** Esta función es un envoltorio general para la optimización de minimización de funciones en Python, que permite la aplicación de varios algoritmos subyacentes como BFGS, L-BFGS-B, SLSQP, entre otros. Es un enfoque flexible que puede adaptarse a diferentes tipos de problemas de optimización y ha sido incluido en este estudio para comparar su rendimiento frente a otros algoritmos especializados.

1.1 Objetivo del Estudio

El objetivo de este trabajo es evaluar la precisión y eficiencia de estos algoritmos al aproximarse al mínimo global de la función de Giunta. Se analizarán tanto los errores en la posición y el valor de la función respecto al óptimo global, como el tiempo computacional requerido para cada algoritmo.
Este análisis no solo proporcionará una comparación directa entre diferentes enfoques de optimización, sino que también permitirá identificar qué métodos son más adecuados para funciones multimodales como la de Giunta, en escenarios de optimización real.

2 Algoritmos de Optimización

2.1 Descenso Máximo

El método de *descenso máximo*, también conocido como *gradiente descendente*, es uno de los algoritmos de optimización más básicos y ampliamente utilizados. Este método se basa en la iteración hacia la dirección opuesta del gradiente de la función objetivo, con el objetivo de minimizarla.

2.1.1 Aplicaciones y Efectividad

El descenso máximo es especialmente efectivo en escenarios donde la función objetivo es convexa y diferenciable, ya que garantiza la convergencia hacia el mínimo global en tales casos. Es ampliamente utilizado en problemas de optimización continua y es la base para muchas variantes y métodos más sofisticados.

2.1.2 Ventajas y Desventajas Generales

- **Ventajas:**
  - Simplicidad de implementación.
  - Eficiencia en problemas convexos.
- **Desventajas:**
  - Sensibilidad a la elección del tamaño de paso (learning rate).
  - Propenso a quedar atrapado en óptimos locales en funciones no convexas.

2.1.3 Rendimiento en la Función de Giunta

En la optimización de la función de Giunta, el método de descenso máximo presenta limitaciones significativas. Dado que la función es multimodal, este algoritmo tiende a quedar atrapado en los numerosos óptimos locales, lo que impide alcanzar el mínimo global. Además, la elección del tamaño de paso es crucial; un tamaño de paso inadecuado puede ralentizar la convergencia o causar que el algoritmo oscile sin converger.

2.2 Método Cuasi-Newton (BFGS)

El *método cuasi-Newton*, y en particular el algoritmo BFGS (Broyden-Fletcher-Goldfarb-Shanno), es un método iterativo para encontrar el mínimo de una función diferenciable. Este método mejora el descenso máximo al utilizar una aproximación de la matriz Hessiana para ajustar el tamaño de paso y la dirección de búsqueda.

2.2.1 Aplicaciones y Efectividad

BFGS es ampliamente utilizado en problemas de optimización no lineal debido a su rapidez de convergencia y robustez, especialmente en funciones suaves y bien condicionadas. Es particularmente eficaz en escenarios donde se necesita un equilibrio entre la precisión y el costo computacional.

2.2.2 Ventajas y Desventajas Generales

- **Ventajas:**
  - Convergencia más rápida que el descenso máximo.
  - Menor dependencia en la elección del tamaño de paso.
- **Desventajas:**
  - Mayor complejidad computacional debido a la estimación de la matriz Hessiana.
  - Menor eficiencia en problemas de alta dimensionalidad o funciones mal condicionadas.

2.2.3 Rendimiento en la Función de Giunta

En el caso de la función de Giunta, BFGS también enfrenta dificultades para evitar los óptimos locales debido a la multimodalidad de la función. No obstante, la capacidad de ajustar la dirección de búsqueda usando una aproximación de la matriz Hessiana permite una convergencia más eficiente hacia un óptimo local en comparación con el descenso máximo.

2.3 Método de Región de Confianza

El *método de región de confianza* es un enfoque iterativo que ajusta el tamaño de la región en la que se confía que la aproximación cuadrática de la función es precisa. En cada iteración, se minimiza un modelo simplificado de la función dentro de esta región.

2.3.1 Aplicaciones y Efectividad

Este método es altamente efectivo en problemas donde la función objetivo es suavemente diferenciable y donde se requiere una robustez adicional frente a la posible mala condición de la matriz Hessiana. Es comúnmente utilizado en optimización no lineal y en aplicaciones que requieren una alta precisión en la convergencia.

2.3.2 Ventajas y Desventajas Generales

- **Ventajas:**
  - Alta robustez frente a variaciones en la curvatura de la función.
  - Ajuste dinámico de la región de búsqueda que mejora la convergencia.
- **Desventajas:**
  - Complejidad computacional elevada.
  - Puede ser ineficiente en problemas de alta dimensionalidad.

2.3.3 Rendimiento en la Función de Giunta

Aunque el método de región de confianza es robusto en términos de convergencia, su aplicación en la función de Giunta está limitada por la multimodalidad de la función. Similar a otros métodos locales, este método puede quedar atrapado en óptimos locales, especialmente si la región de confianza es inicialmente grande y abarca múltiples picos de la función.

2.4 Evolución Diferencial

El *algoritmo de evolución diferencial* es un método estocástico de optimización global que se basa en la mutación, cruce y selección para explorar el espacio de búsqueda. Es especialmente útil en problemas de optimización no convexos, discontinuos o multimodales.

2.4.1 Aplicaciones y Efectividad

Evolución diferencial es ampliamente utilizado en optimización global debido a su capacidad para explorar el espacio de búsqueda y evitar óptimos locales. Es eficaz en problemas donde otros métodos podrían fallar debido a la presencia de múltiples óptimos locales o superficies de respuesta irregulares.

2.4.2 Ventajas y Desventajas Generales

- **Ventajas:**
  - Alta capacidad de exploración global.
  - No requiere el cálculo de derivadas.
- **Desventajas:**
  - Convergencia más lenta en comparación con métodos de optimización local.
  - Depende de la elección de los parámetros de mutación y cruce.

2.4.3 Rendimiento en la Función de Giunta

En la optimización de la función de Giunta, la evolución diferencial destaca por su capacidad para explorar el espacio de búsqueda y escapar de óptimos locales. Sin embargo, su convergencia puede ser más lenta que la de otros métodos locales, lo que puede ser una desventaja si se requiere una solución rápida.

2.5 Algoritmo Genético

El *algoritmo genético* es un método de optimización inspirado en el proceso de selección natural. Utiliza operadores como la selección, cruce y mutación para evolucionar una población de soluciones y encontrar el óptimo global.

2.5.1 Aplicaciones y Efectividad

Los algoritmos genéticos son efectivos en problemas de optimización global, especialmente aquellos con múltiples óptimos locales, restricciones complejas o superficies de respuesta no suaves. Son aplicables en una variedad de dominios, desde el diseño de ingeniería hasta la inteligencia artificial.

2.5.2 Ventajas y Desventajas Generales

- **Ventajas:**
  - Capacidad para escapar de óptimos locales.
  - Flexibilidad para manejar problemas con restricciones no lineales y dominios discretos.
- **Desventajas:**
  - Convergencia relativamente lenta.
  - Dependencia en la configuración de parámetros como la tasa de mutación y cruce.

2.5.3 Rendimiento en la Función de Giunta

Para la función de Giunta, los algoritmos genéticos son altamente efectivos para evitar los óptimos locales, dada la multimodalidad de la función. No obstante, similar a la evolución diferencial, la convergencia puede ser lenta y depende de una buena configuración de los parámetros del algoritmo.

2.6 Algoritmo de Enjambre de Partículas (PSO)

El *algoritmo de enjambre de partículas (PSO)* es un método de optimización que simula el comportamiento de un grupo de partículas en movimiento dentro del espacio de búsqueda. Las partículas ajustan su posición en función de su experiencia personal y la del enjambre, buscando el óptimo global.

2.6.1 Aplicaciones y Efectividad

PSO es adecuado para problemas de optimización global, especialmente aquellos con múltiples óptimos locales. Es comúnmente utilizado en problemas de alta dimensionalidad y en aplicaciones donde se requiere una búsqueda global efectiva.

2.6.2 Ventajas y Desventajas Generales

- **Ventajas:**
  - Buena capacidad para encontrar el óptimo global.
  - Menor dependencia en la elección de parámetros en comparación con algoritmos genéticos.
- **Desventajas:**
  - Posibilidad de convergencia prematura.
  - Sensibilidad a la elección de ciertos parámetros como el coeficiente de inercia.

2.6.3 Rendimiento en la Función de Giunta

En la función de Giunta, PSO se muestra como un método eficaz para evitar los óptimos locales y encontrar el mínimo global. Sin embargo, existe el riesgo de convergencia prematura si las partículas se agrupan demasiado rápido en una región subóptima del espacio de búsqueda.

2.7 Recocido Simulado

El *recocido simulado* es un algoritmo estocástico de optimización global inspirado en el proceso de recocido en metalurgia. El algoritmo permite que el sistema explore soluciones subóptimas en etapas tempranas, reduciendo gradualmente la probabilidad de aceptar soluciones peores a medida que avanza la búsqueda.

2.7.1 Aplicaciones y Efectividad

Recocido simulado es útil en problemas de optimización global, especialmente aquellos con superficies de búsqueda complejas y múltiples óptimos locales. Es eficaz para evitar quedar atrapado en mínimos locales, aunque puede requerir un tiempo considerable para alcanzar la convergencia.

2.7.2 Ventajas y Desventajas Generales

- **Ventajas:**
  - Capacidad para escapar de mínimos locales.
  - No requiere el cálculo de derivadas, siendo útil en funciones no diferenciables.
- **Desventajas:**
  - Convergencia lenta.
  - Sensibilidad a la elección de la programación de enfriamiento.

2.7.3 Rendimiento en la Función de Giunta

Para la función de Giunta, el recocido simulado es un enfoque robusto que puede evitar los óptimos locales, pero su lenta convergencia puede ser una desventaja en escenarios donde el tiempo computacional es un factor crítico.

2.8 Método de Salto de Cuenca (Basin Hopping)

El *método de salto de cuenca* es un enfoque de optimización global que combina la búsqueda local con saltos aleatorios entre diferentes regiones del espacio de búsqueda. Este método es particularmente útil para funciones multimodales.

2.8.1 Aplicaciones y Efectividad

Basin Hopping es efectivo en problemas donde la función objetivo tiene múltiples mínimos locales. Es comúnmente utilizado en química computacional y biología estructural para encontrar las configuraciones energéticas más bajas.

2.8.2 Ventajas y Desventajas Generales

- **Ventajas:**
  - Capacidad para explorar múltiples cuencas de atracción.
  - Combinación efectiva de búsqueda local y global.
- **Desventajas:**
  - Puede requerir ajustes finos en los parámetros de salto.
  - La eficiencia depende del éxito del método local utilizado en cada cuenca.

2.8.3 Rendimiento en la Función de Giunta

En la función de Giunta, Basin Hopping muestra un rendimiento notable, ya que su combinación de saltos globales y búsqueda local permite evitar los óptimos locales. No obstante, la elección del método local puede influir significativamente en la efectividad del algoritmo.

2.9 Función `minimize` de `scipy.optimize`

La función `minimize` de `scipy.optimize` es un envoltorio que permite aplicar varios algoritmos de minimización a funciones continuas. Los métodos disponibles incluyen BFGS, L-BFGS-B, SLSQP, entre otros.

2.9.1 Aplicaciones y Efectividad

La función `minimize` es útil en una amplia gama de problemas de optimización debido a su flexibilidad y facilidad de uso. Es particularmente efectiva en funciones suavemente diferenciables donde se necesita una solución rápida y eficiente.

2.9.2 Ventajas y Desventajas Generales

- **Ventajas:**
  - Flexibilidad para seleccionar el método de optimización más adecuado.
  - Fácil de implementar y ajustar.
- **Desventajas:**
  - Limitada en su capacidad para manejar funciones altamente multimodales sin modificaciones.
  - La efectividad depende del método específico seleccionado dentro de la función.

2.9.3 Rendimiento en la Función de Giunta

En la función de Giunta, el rendimiento de `minimize` depende en gran medida del método subyacente seleccionado. Si bien los métodos como BFGS pueden ser rápidos y eficientes para encontrar óptimos locales, la función es propensa a quedarse atrapada en estos mínimos debido a la multimodalidad de la función de Giunta.

2.10 Inaplicabilidad del Algoritmo de Newton

El *algoritmo de Newton* es un método de optimización que utiliza la matriz Hessiana completa para ajustar la dirección y el tamaño de los pasos en la búsqueda del mínimo de una función. Aunque es un método muy eficaz para problemas convexos y bien condicionados, presenta limitaciones significativas cuando se aplica a funciones no lineales y multimodales como la función de Giunta.

2.10.1 Limitaciones en la Función de Giunta

La función de Giunta es altamente no lineal y presenta una estructura multimodal con múltiples mínimos locales. El algoritmo de Newton, al ser un método de optimización local, asume que la función objetivo puede ser bien aproximada por una parábola en torno a un punto de interés, lo cual es efectivo en funciones suaves y convexas. Sin embargo, debido a la complejidad y no linealidad de la función de Giunta, esta suposición falla, lo que puede llevar a resultados imprecisos o a una convergencia hacia óptimos locales no deseados.
Además, la no linealidad de la función de Giunta implica que la matriz Hessiana puede ser extremadamente difícil de calcular y manejar, ya que puede estar mal condicionada o variar drásticamente en diferentes regiones del espacio de búsqueda. Esto no solo incrementa el costo computacional del algoritmo, sino que también puede resultar en direcciones de descenso ineficaces, haciendo que el método sea poco fiable y susceptible a quedarse atrapado en mínimos locales.
Por estas razones, se ha decidido no implementar el algoritmo de Newton en el presente estudio, dado que los métodos seleccionados ofrecen una mejor capacidad para lidiar con las características no lineales y multimodales de la función de Giunta.

3 Resultados

3.1 Precisión en la Solución

La precisión de los algoritmos se evaluó en términos del error en la posición $\mathbf{x}$ encontrada en comparación con el mínimo global conocido $\mathbf{x}^*$, así como el valor de la función $f(\mathbf{x})$ en ese punto. La Figura 2 muestra los errores en la posición para cada uno de los algoritmos, mientras que la Tabla 1 resume los valores obtenidos.
Véase como los algoritmos mostraron rendimientos similares y relativamente buenos en la aproximación del mínimo global de la Giunta Function.

3.2 Tiempo Computacional

El tiempo computacional requerido por cada uno de los algoritmos para converger a una solución se muestra en la Figura 3. Es notable que algunos algoritmos como Genetic Algorithm, Simulated Annealing, y Basin Hopping requieren significativamente más tiempo en comparación con métodos como Minimize o Quasi-Newton.

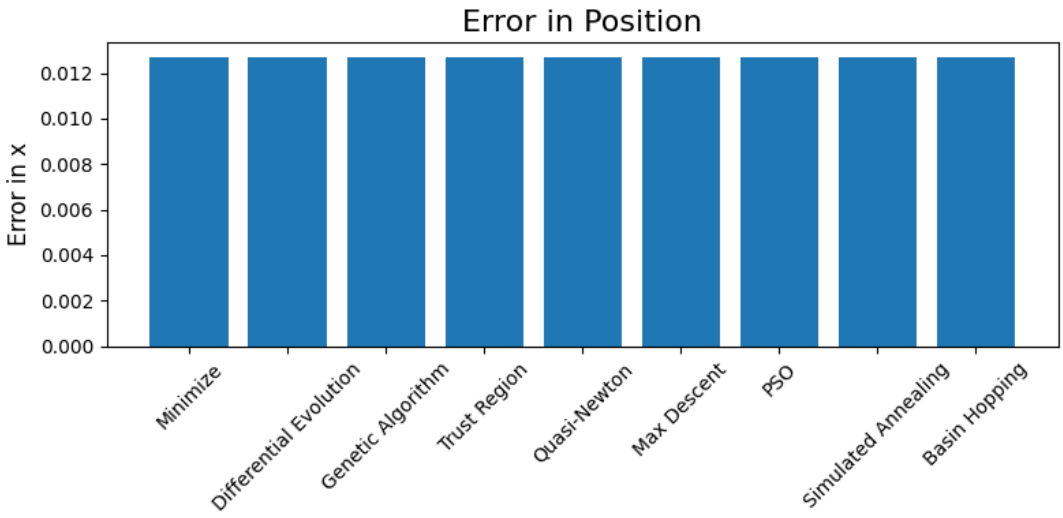

Figure 2: Error en la posición obtenida por cada algoritmo.

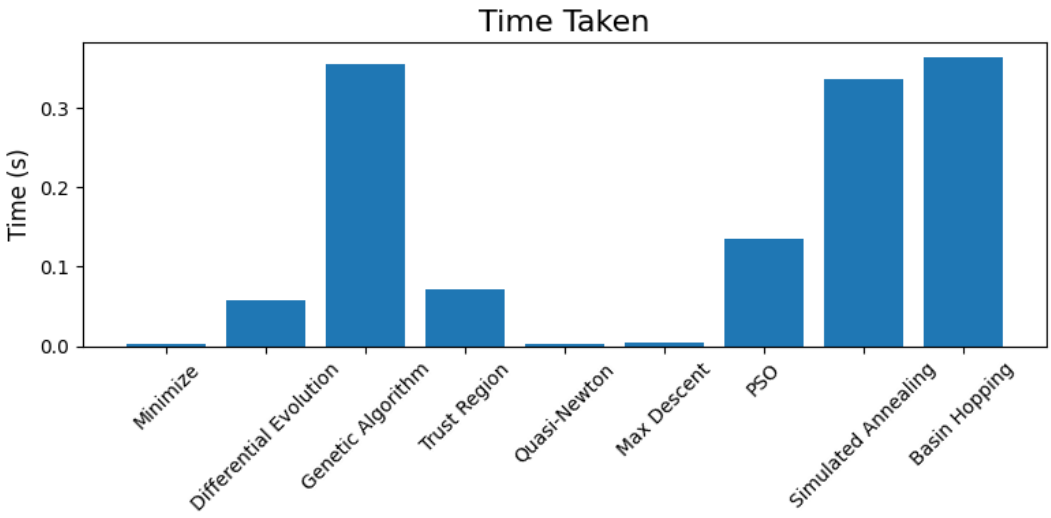

Figure 3: Tiempo de ejecución de cada algoritmo.

3.3 Resumen de Resultados

Los resultados específicos de cada algoritmo se presentan en la Tabla 1. Los valores de $\mathbf{x}$ obtenidos se redondearon a 8 lugares decimales para asegurar una comparación clara y concisa. Esta tabla permite observar las diferencias en precisión y tiempo de manera más detallada.

También en la tabla 2 podemos apreciar detalladamente los errores a la hora de encontrar el punto de mínimo y en la evaluación de la función. Como ilustraba la figura 2, los algoritmos alcanzaron soluciones significativamente similares.

3.4 Observaciones sobre el Comportamiento con Dimensiones Crecientes

El análisis del comportamiento de los algoritmos de optimización con dimensiones crecientes es crucial, dado que muchos problemas en la práctica no se limitan a espacios de baja dimensionalidad. En este contexto, se deben considerar varios factores que afectan el rendimiento de los algoritmos estudiados cuando se incrementa la dimensionalidad del problema, en este caso, la Giunta Function.

- **Métodos Clásicos:** Los métodos como el *descenso máximo*, *quasi-Newton* (BFGS), y el *método de región de confianza* tienden a enfrentar desafíos significativos en espacios de alta dimensionalidad. La principal dificultad

Optimization Results

Method	x	f(x)	Time (s)
Minimize	[0.46732119 0.46732119]	0.06447042	0.00221777
Differential Evolution	[0.46731737 0.46731733]	0.06447042	0.04906511
Genetic Algorithm	[0.46732002 0.46732003]	0.06447042	0.36350369
Trust Region	[0.46730182 0.46730182]	0.06447042	0.08343649
Quasi-Newton	[0.46732119 0.46732119]	0.06447042	0.00614452
Max Descent	[0.46732003 0.46732003]	0.06447042	0.0041368
PSO	[0.46725268 0.46732441]	0.06447043	0.13822508
Simulated Annealing	[0.46732001 0.46731989]	0.06447042	0.38621569
Basin Hopping	[0.46732003 0.46732002]	0.06447042	0.36054778

Table 1: Resumen de los resultados de optimización.

Errors Relative to Global Minimum

Method	Error in x	Error in f
Minimize	0.01269734	0.00402342
Differential Evolution	0.01269575	0.00402342
Genetic Algorithm	0.01269568	0.00402342
Trust Region	0.01266994	0.00402342
Quasi-Newton	0.01269733	0.00402342
Max Descent	0.01269569	0.00402342
PSO	0.01267554	0.00402342
Simulated Annealing	0.01269564	0.00402342
Basin Hopping	0.01269569	0.00402342

Table 2: Errores relativos al mínimo global.

radica en la complejidad computacional que aumenta con el número de dimensiones, especialmente para BFGS, donde la necesidad de calcular y almacenar la matriz Hessiana aproximada se vuelve prohibitiva. Además, estos métodos son propensos a quedar atrapados en mínimos locales, un problema que se acentúa a medida que aumenta la cantidad de dimensiones y, por lo tanto, el número de óptimos locales. Esto implica que, aunque estos algoritmos pueden ser eficientes en espacios de baja a mediana dimensionalidad, su aplicabilidad disminuye en problemas de alta dimensionalidad.

- **Algoritmos Evolutivos y Basados en Población:** Los algoritmos evolutivos, como la *evolución diferencial* y los *algoritmos genéticos*, así como el *algoritmo de enjambre de partículas* (PSO), son más adecuados para manejar el incremento dimensional debido a su capacidad intrínseca para explorar grandes espacios de búsqueda. Estos métodos tienden a mantener una buena capacidad exploratoria incluso en espacios de alta dimensionalidad, lo que les permite evitar quedarse atrapados en óptimos locales. Sin embargo, esto viene a costa de un aumento significativo en el tiempo computacional, ya que la evaluación de las funciones objetivo en un espacio de mayor dimensión es inherentemente más costosa.

- **Métodos Estocásticos:** El *recocido simulado* y el *salto de cuenca* (Basin Hopping) también son robustos frente al incremento dimensional, principalmente debido a su capacidad para realizar exploraciones globales del espacio de búsqueda. No obstante, su rendimiento en términos de tiempo puede ser considerablemente afectado, ya que la exploración estocástica de un espacio de búsqueda más grande generalmente requiere más iteraciones para alcanzar la convergencia. Además, la eficacia de estos métodos puede depender fuertemente de la parametrización del algoritmo, la cual puede necesitar ajustes más finos en espacios de alta dimensionalidad.

La selección del algoritmo de optimización más adecuado para problemas de alta dimensionalidad debe balancear la necesidad de precisión y la capacidad de evitar óptimos locales con el costo computacional. Algoritmos como la evolución diferencial, PSO, y recocido simulado son generalmente preferibles en escenarios de alta dimensionalidad,

a pesar de sus mayores requisitos computacionales, mientras que los métodos clásicos pueden verse limitados por la complejidad computacional y su tendencia a converger en mínimos locales.

4 Conclusiones

El presente estudio ha proporcionado un análisis exhaustivo del rendimiento de diversos algoritmos de optimización aplicados a la Giunta Function, una función multimodal que presenta desafíos significativos debido a la presencia de múltiples óptimos locales. Se evaluaron tanto algoritmos clásicos como avanzados, considerando su precisión para aproximarse al mínimo global y el tiempo computacional requerido para alcanzar dicha solución.
Los resultados obtenidos permiten extraer varias conclusiones clave:

- **Precisión y Convergencia:** Los métodos clásicos, como el *descenso máximo* y *quasi-Newton* (BFGS), demostraron una buena capacidad de convergencia en escenarios de baja dimensionalidad, sin embargo, su propensión a quedarse atrapados en óptimos locales limita su aplicabilidad en funciones multimodales como la Giunta Function. Estos métodos son más adecuados cuando se busca una solución rápida en un entorno donde la función objetivo es suave y diferenciable, pero son menos efectivos en la búsqueda global.

- **Exploración Global:** Los algoritmos evolutivos, como la *evolución diferencial* y los *algoritmos genéticos*, junto con el *algoritmo de enjambre de partículas* (PSO), mostraron una mayor capacidad para evitar quedar atrapados en óptimos locales, logrando una exploración más efectiva del espacio de búsqueda. Aunque estos algoritmos requieren un mayor tiempo computacional, son preferibles en problemas multimodales o de alta dimensionalidad, donde la exploración global es crucial.

- **Robustez en Dimensiones Crecientes:** Al aumentar la dimensionalidad del problema, los métodos estocásticos como el *recocido simulado* y el *salto de cuenca* (Basin Hopping) demostraron una robustez significativa. Estos métodos, aunque más lentos en convergencia, son capaces de explorar de manera efectiva grandes espacios de búsqueda y evitar la convergencia prematura en mínimos locales. Su capacidad para ajustar dinámicamente la búsqueda en función de la topología del espacio de búsqueda los hace especialmente útiles en problemas complejos y de alta dimensionalidad.

- **Tiempo Computacional:** Un aspecto crucial en la selección del algoritmo adecuado es el balance entre la precisión y el tiempo computacional. Los métodos como *BFGS* y el *método de región de confianza* son rápidos en su convergencia hacia un óptimo local, pero este beneficio se ve contrarrestado por su susceptibilidad a quedar atrapados en óptimos locales. Por otro lado, aunque los algoritmos evolutivos y estocásticos requieren más tiempo, ofrecen una mejor exploración global y son más fiables en problemas de mayor complejidad.

En resumen, la elección del algoritmo de optimización debe basarse en las características específicas del problema a resolver. Para funciones multimodales y problemas de alta dimensionalidad, los métodos estocásticos y evolutivos son generalmente más efectivos, a pesar de su mayor costo computacional. Los métodos clásicos, aunque rápidos y eficientes en escenarios más simples, deben usarse con precaución cuando se enfrentan a problemas con una topología compleja, como la Giunta Function. Este estudio proporciona una guía valiosa para seleccionar el método más apropiado en función de las características del problema y las limitaciones de tiempo computacional.