Convergencia de Algoritmos de Optimización en Problema de Alta Dimensión: Un Análisis Comparativo

Eduar Castrillo Velilla[1] and Lifeth Alvarez Camacho[2]

Universidad Nacional de Colombia, Bogotá D.C., Colombia
emcastrillov@unal.edu.co[1], lalvarezc@unal.edu.co[2]

Abstract. En este artículo se desarrolla un análisis comparativo de la convergencia de varios algoritmos de optimización mono-objetivo aplicados en un problema de alta dimensión. El análisis de convergencia se ha realizado sobre los resultados obtenidos al optimizar la función Rastrigin en 10 dimensiones mediante los algoritmos Ascenso a la Colina, Enjambre de Partículas, Evolución Diferencial y Estrategias Evolutivas.

Keywords: Optimización Mono-Objetivo, Análisis Comparativo, Rastrigin.

1 Introducción

La optimización está presente en la mayoría de las cosas que hacemos. Optimización de horarios de trabajo, optimización de estrategias de juego, optimización de planes de transporte, entre muchas otras. La optimización es un área de estudio ampliamente investigado debido a su aplicabilidad universal, pero también debido a su fuerte contenido teórico y algorítmico [1].

A grandes rasgos un problema de optimización consiste en encontrar soluciones en un dominio (espacio de soluciones) que minimicen o maximicen una o varias funciones objetivos. Dependiendo de si la función es continua o discreta, el problema de optimización se clasifica como continuo o discreto respectivamente. Además, si el problema cuenta con una o varias funciones objetivo, dicho problema se clasifica como mono-objetivo o multi-objetivo respectivamente.

Cuando la solución analítica de una función objetivo que modela un problema no puede ser calculada, es necesario recurrir a técnicas de búsqueda de soluciones exactas o aproximadas. Una posible estrategia solución es recorrer el espacio de soluciones para encontrar el minimizador o maximizador de la función, pero debido a la potencial infinitud de posibles soluciones o a la dificultad para recorrerlas todas, esta estrategia es computacionalmente intratable. Por ello, se han desarrollado métodos y algoritmos de optimización que permitan explorar el dominio de la función de manera eficiente, para encontrar soluciones exactas o aproximadas. Existen gran variedad de algoritmos para la resolución de problemas de optimización [1], entre ellos están los algoritmos clásicos de Ascenso a la Colina, Estrategias Evolutivas, Evolución Diferencial, y Enjambre de Partículas. Estos algoritmos han demostrado eficiencia y eficacia al

resolver problemas de optimización, alcanzando buenos tiempos de convergencia hacia soluciones exactas o aproximadas.

Cabe notar que las funciones objetivo que modelan problemas reales y que necesitan ser optimizadas, usualmente son complejas, su solución analítica es desconocida, poseen muchos óptimos locales, o están definidas en altas dimensiones. Para tales tipos de funciones (especialmente en altas dimensiones), los algoritmos empiezan a presentar problemas de convergencia hacia buenas soluciones. En este artículo, realizamos un análisis comparativo de la convergencia de algunos algoritmos clásicos de optimización mono-objetivo aplicados a una función continua definida en un espacio de alta dimensión.

Este artículo está dividido en 4 secciones. La sección 2 presenta una descripción de los algoritmos clásicos utilizados en el estudio comparativo y de la función objetivo. La sección 3 presenta los detalles de la experimentación y análisis de los resultados. Finalmente la sección 4 presenta algunas conclusiones.

2 Marco Teórico

2.1 Problema de Optimización

Matemáticamente un problema de optimización mono-objetivo puede definirse como la minimización o la maximización de una función objetivo. La minimización y la maximización pueden verse como problemas equivalentes si se plantean de la siguiente manera.

$$\min_{x} f(x) \Leftrightarrow \max_{x} [-f(x)]$$

$$\max_{x} f(x) \Leftrightarrow \min_{x} [-f(x)] \tag{1}$$

La función $f(x)$ es llamada la función objetivo y el vector x es llamado vector independiente o variable de decisión (algunas veces llamado vector solución). El número de elementos en el vector x, es llamado la dimensión del problema. Como se muestra en la ecuación (1) un algoritmo diseñado para maximizar una función puede ser fácilmente utilizados para minimizar la misma función. El vector x que minimiza el valor (costo) $f(x)$ se conoce como minimizador, mientras que el vector x que maximiza el valor (Costo) $f(x)$ se conoce como maximizador.

2.2 Función Rastrigin

En optimización matemática la función Rastrigin es una función non-convexa, no lineal y multi-modal [5]. En n dimensiones la función está definida como

$$f(x) = An + \sum_{i=1}^{n} [x_i^2 - A\cos(2\pi x_i)] \tag{2}$$

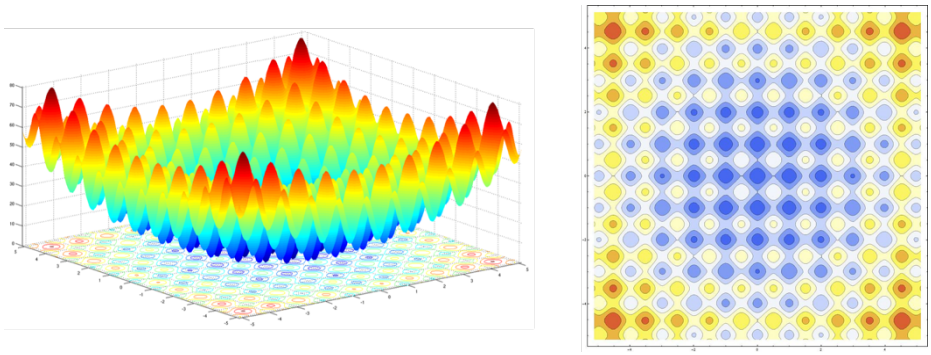

Fig. 1. (Izquierda) Gráfica 3-D de la función Rastrigin de 2 dimensiones. (Derecha) diagrama de contorno de la función Rastrigin.

Donde $A = 10$ y $x_i \in \{-5.12, 5.12\}$. La función tiene un mínimo global en $x = 0$, donde $f(x) = 0$. La Fig. 1 muestra la gráfica 3-D de la función Rastrigin de 2 dimensiones y su respectivo diagrama de contorno.

2.3 Ascenso a la Colina

La idea de Ascenso a la Colina (AC) es tan simple que puede haber sido descubierto múltiples veces, por lo que no es fácil establecer el origen del algoritmo. Si se quiere llegar al punto más alto de una montaña, una posible estrategia dar un paso en dirección de máxima pendiente, luego se reevalúa la pendiente y se ejecuta el siguiente paso. El algoritmo itera hasta que no es posible ejecutar un paso hacia una dirección con pendiente mayor a la actual.

```
Initialize p_m ∈ [0, 1] as the probability of mutation
x_0 ← randomly generated individual
While not(termination criterion)
    Compute the fitness f(x_0) of x_0
    x_1 ← x_0
    For each solution feature q = 1, ··· , n
        Generate a uniformly distributed random number r ∈ [0, 1]
        If r < p_m then
            Replace the q-th solution feature of x_1 with a random mutation
        End if
    Next solution feature
    Compute the fitness f(x_1) of x_1
    If f(x_1) > f(x_0) then
        x_0 ← x_1
    End if
Next generation
```

Fig. 2. Pseudo-código del algoritmo Ascenso a la Colina [1]

2.4 Estrategia Evolutiva

La Estrategia Evolutiva en su versión (μ+λ)-EE, genera una descendencia de tamaño λ, donde cada nuevo individuo es generado por recombinación de dos padres escogidos aleatoriamente y una mutación. Luego, de entre el conjunto de padres e hijos se escogen los mejores μ individuos. El método selección por lo tanto es elitista [2].

Initialize constants τ and τ' as shown in Equation (6.28).
$\{(x_k, \sigma_k)\} \leftarrow$ randomly generated individuals, $k \in [1, \mu]$.
Each x_k is a candidate solution, and each σ_k is a standard deviation vector.
Note that $x_k \in R^n$ and $\sigma_k \in R^n$.
While not(termination criterion)
  For $k = 1, \cdots, \lambda$
    Randomly select two parents from $\{(x_k, \sigma_k)\}$
    Use a recombination method to combine the two parents and obtain
      a child, which is denoted as (x'_k, σ'_k)
    Generate a random scalar ρ_0 from $N(0, 1)$
    Generate a random vector $\begin{bmatrix} \rho_1 & \cdots & \rho_n \end{bmatrix}$ from $N(0, I)$
    $\sigma'_{ki} \leftarrow \sigma'_{ki} \exp(\tau' \rho_0 + \tau \rho_i)$ for $i \in [1, n]$
    $\Sigma'_k \leftarrow \text{diag}((\sigma'_{k1})^2, \cdots, (\sigma'_{kn})^2) \in R^{n \times n}$
    Generate a random vector r from $N(0, \Sigma'_k)$
    $x'_k \leftarrow x'_k + r$
  Next k
  If this is a $(\mu + \lambda)$-ES then
    $\{(x_k, \sigma_k)\} \leftarrow$ the best μ individuals from $\{(x_k, \sigma_k)\} \cup \{(x'_k, \sigma'_k)\}$
  else if this is a (μ, λ)-ES then
    $\{(x_k, \sigma_k)\} \leftarrow$ the best μ individuals from $\{(x'_k, \sigma'_k)\}$
  End if
Next generation

Fig. 3. Pseudo-código del algoritmo Estrategia Evolutiva [1]

2.5 Enjambre de Partículas

Enjambre de Partículas (EP) está basado en la observación de que los individuos trabajan en grupo no sólo para mejorar su aptitud colectiva en alguna tarea, sino también la individual. El principio fundamental de EP se ven tanto en comportamientos animales como humanos [3].

2.6 Evolución Diferencial

En Evolución Diferencial (ED) cada individuo está representado como un vector real dimensión n. ED está basado en la idea de tomar dos vectores individuos x_1 y x_2, calcular su diferencia $d = x_1$-x_2 y sumarle una versión escalada del vector d a un tercer vector x_3 ($y = x_3 + Fd$), para generar un nuevo individuo y [4].

Initialize a random population of individuals $\{x_i\}$, $i \in [1, N]$
Initialize each individual's n-element velocity vector v_i, $i \in [1, N]$
Initialize the best-so-far position of each individual: $b_i \leftarrow x_i$, $i \in [1, N]$
Define the neighborhood size $\sigma < N$
Define the maximum influence values $\phi_{1,\max}$ and $\phi_{2,\max}$
Define the maximum velocity $v_{\max}$
While not(termination criterion)
    For each individual x_i, $i \in [1, N]$
        $H_i \leftarrow \{\sigma$ nearest neighbors of $x_i\}$
        $h_i \leftarrow \arg\min_x\{f(x) : x \in H_i\}$
        Generate a random vector ϕ_1 with $\phi_1(k) \sim U[0, \phi_{1,\max}]$ for $k \in [1, n]$
        Generate a random vector ϕ_2 with $\phi_2(k) \sim U[0, \phi_{2,\max}]$ for $k \in [1, n]$
        $v_i \leftarrow v_i + \phi_1 \circ (b_i - x_i) + \phi_2 \circ (h_i - x_i)$
        If $|v_i| > v_{\max}$ then
            $v_i \leftarrow v_i v_{\max} / |v_i|$
        End if
        $x_i \leftarrow x_i + v_i$
        $b_i \leftarrow \arg\min\{f(x_i), f(b_i)\}$
    Next individual
Next generation

Fig. 4. Pseudo-código del algoritmo Enjambre de Partículas [1]

$F =$ stepsize parameter $\in [0.4, 0.9]$
$c =$ crossover rate $\in [0.1, 1]$
Initialize a population of candidate solutions $\{x_i\}$ for $i \in [1, N]$
While not(termination criterion)
    For each individual x_i, $i \in [1, N]$
        $r_1 \leftarrow$ random integer $\in [1, N] : r_1 \neq i$
        $r_2 \leftarrow$ random integer $\in [1, N] : r_2 \notin \{i, r_1\}$
        $r_3 \leftarrow$ random integer $\in [1, N] : r_3 \notin \{i, r_1, r_2\}$
        $v_i \leftarrow x_{r1} + F(x_{r2} - x_{r3})$ (mutant vector)
        $\mathcal{J}_r \leftarrow$ random integer $\in [1, n]$
        For each dimension $j \in [1, n]$
            $r_{cj} \leftarrow$ random number $\in [0, 1]$
            If $(r_{cj} < c)$ or $(j = \mathcal{J}_r)$ then
                $u_{ij} \leftarrow v_{ij}$
            else
                $u_{ij} \leftarrow x_{ij}$
            End if
        Next dimension
    Next individual
    For each population index $i \in [1, N]$
        If $f(u_i) < f(x_i)$ then $x_i \leftarrow u_i$
    Next population index
Next generation

Fig. 5. Pseudo-código del algoritmo Enjambre de Partículas [1]

2.7 Representación de los Individuos

Cada individuo en la población se representa como un vector de reales de dimensión 10.

2.8 Operadores Genéticos

Se utilizan los operadores genéticos propuestos por cada algoritmo. (Ver Fig. 2, 3, 4, 5).

3 Experimentos

Versiones de los algoritmos mencionados en la sección anterior fueron implementados en C++ y fueron aplicados para optimizar la función Rastrigin de 10 dimensiones. Los parámetros de los experimentos para cada algoritmo están listados en la Tabla 1.

Tabla 1. Parámetros de los experimentos para cada algoritmo

Algoritmo	Población	Iteraciones	Ejecuciones
AC	1	1,000,000	30
$(\mu+\lambda)$-EE	$\mu = 100\ \lambda = 100$	1,000,000	30
EP	100	1,000,000	30
ED	100	1,000,000	30

3.1 Resultados

Por cada algoritmo se ha graficado el mejor, peor y la mediana del costo de la población de entre todas las ejecuciones, en función de la iteración. Fig. 6 a 9 muestran gráficamente los resultados de la optimización realizada por cada algoritmo en la función Rastrigin de 10 dimensiones (2).

Fig. 6. muestra el resultado obtenido por AC. En este caso, el costo disminuyó progresivamente al aumentar el número de iteraciones, hasta alcanzar un costo máximo de 38.6, un mínimo de 11 y una mediana de 21.6. Fig. 7. muestra el resultado obtenido por EE. Al igual que AC, el costo disminuyó progresivamente al aumentar el número de iteraciones, hasta alcanzar un costo máximo de 130.7, un mínimo de 8.8 y una mediana de 47.6. Fig. 8. muestra el resultado obtenido por EP. EP entró rápidamente en un óptimo local donde alcanzó un costo máximo de 34.8, un mínimo de 4.3 y una mediana de 12.5. Fig. 9. muestra el resultado obtenido por EP. Al igual que EP, ED entró rápidamente en un óptimo local donde alcanzó un costo máximo de 45.7, un mínimo de 6.6 y una mediana de 28.12.

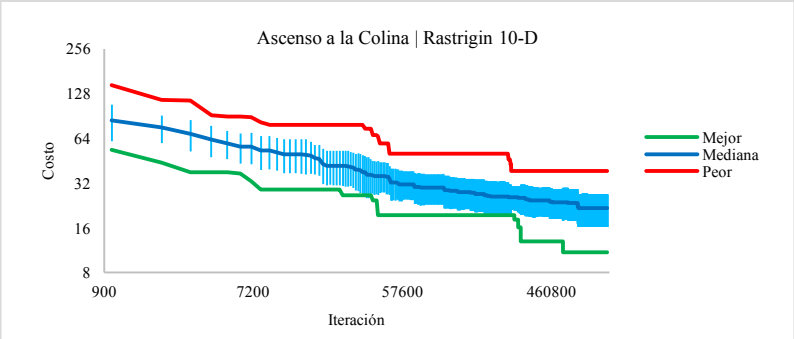

Fig. 6. Mejor, peor y mediana del costo (con la desviación estándar) de la población de entre todas las ejecuciones, en función de la iteración (costo bajo es mejor)

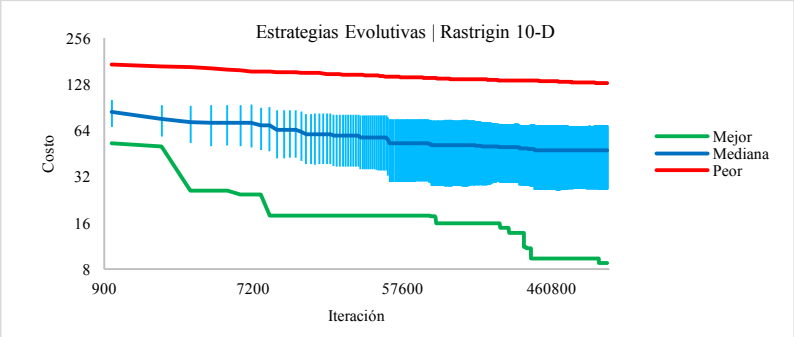

Fig. 7. Mejor, peor y mediana del costo (con la desviación estándar) de la población de entre todas las ejecuciones, en función de la iteración (costo bajo es mejor)

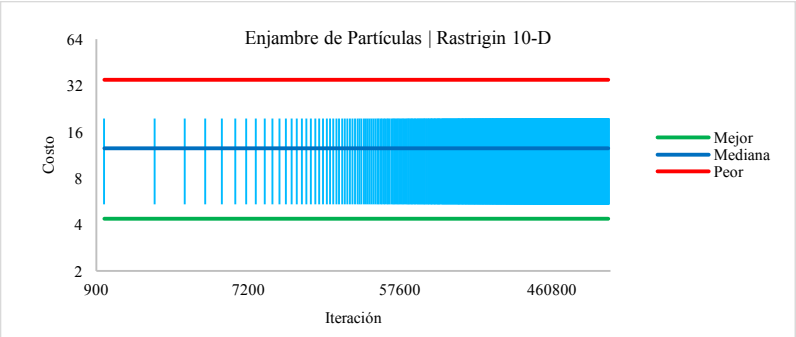

Fig. 8. Mejor, peor y mediana del costo (con la desviación estándar) de la población de entre todas las ejecuciones, en función de la iteración (costo bajo es mejor)

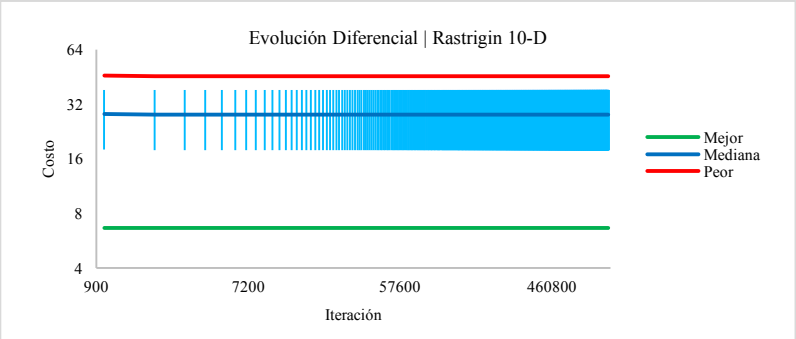

Fig. 9. Mejor, peor y mediana del costo (con la desviación estándar) de la población de entre todas las ejecuciones, en función de la iteración (costo bajo es mejor)

Como se puede notar, todos los algoritmos implementados son de estado estable, pues los resultados obtenidos únicamente mejoran al incrementar el número de iteraciones, hasta que se llega al óptimo local. Desafortunadamente ninguno de los algoritmos pudo encontrar el óptimo global cuyo costo es $f(x) = 0$. La Tabla 2. resume las estadísticas obtenidas por cada algoritmo en la última iteración. Estadísticamente el algoritmo Enjambre de Partículas obtuvo el mejor desempeño.

Tabla 2. Estadísticas de los algoritmos en la última iteración

Algoritmo	Costo Min.	Costo Max.	Mediana	Desviación Estándar
AC	11.03	38.67	21.66	**+/- 5.33**
(μ+λ)-EE	8.83	130.59	47.66	+/- 21.09
EP	**4.36**	**34.84**	**12.50**	+/- 7.10
ED	6.64	45.73	28.12	+/- 10.26

4 Conclusiones

En el presente artículo se realizó un análisis comparativo de la convergencia de algoritmos clásicos de optimización mono-objetivo en la función Rastrigin de 10 dimensiones. Se pudo observar que las implementaciones realizadas son de estado estable, pues sus aproximaciones únicamente mejoran al aumentar el número de iteraciones. A pesar de que ningún algoritmo logró encontrar el óptimo global de la función objetivo, pudieron encontrar "buenos" óptimos locales. El algoritmo Enjambre de Partículas mostró estadísticamente el mejor desempeño en el problema de optimización propuesto.

Referencias

1. Simon, D. (2013). Evolutionary optimization algorithms. John Wiley & Sons.

2. Rechenberg, I. (1989). Evolution strategy: Nature's way of optimization. In Optimization: Methods and applications, possibilities and limitations (pp. 106-126). Springer Berlin Heidelberg.
3. Eberhart, R., & Kennedy, J. (1995, October). A new optimizer using particle swarm theory. In Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on (pp. 39-43). IEEE.
4. Storn, R., & Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization, 11(4), 341-359.
5. Mühlenbein, H., Schomisch, M., & Born, J. (1991). The parallel genetic algorithm as function optimizer. Parallel computing, 17(6-7), 619-632.