Revisión del método Hybrid Bat Algorithm para la minimización de diversas funciones

Antoni Mauricio
Centro de Investigación e Innovación en Ciencias de la Computación
Universidad Católica San Pablo
manasses.mauricio@ucsp.edu.pe

Jorge López
Centro de Investigación e Innovación en Ciencias de la Computación
Universidad Católica San Pablo
jorge.lopez.caceres@usp.pe

***Abstract*—Los métodos basados en inteligencia de enjambre son técnicas muy poderosas para fines de optimización, con el paso de los años se han desarrollado metaheurísticas sobre estos métodos para mejorar la eficiencia en las etapas de exploración y profundización. En este trabajo se presenta la implementación y aplicación directa del algoritmo Hybrid Bat Algorithm (HBA), basado en el Bat Algorithm (BA). El HBA es la combinación de BA con estrategias de evolución diferencial.**

Keywords: Hybrid Bat Algorithm, estrategias de evolucin diferencial.

1. Introducción

La ecolocación es una característica importante del comportamiento de los murciélagos. Eso significa que los murciélagos emiten un pulso de sonido y escucha el eco que rebota contra en los obstáculos mientras vuela. Este fenómeno ha inspirado a Yang [1] a desarrollar el BA. El algoritmo muestra buenos resultados cuando se trata de problemas de optimización de poca dimensión, pero puede volverse problemático para problemas de dimensiones superiores debido a que tienden a converger muy rápido inicialmente. Por otro lado, la evolución diferencial [2] es un algoritmo evolutivo tpico con mutación, crossover y seleccin diferencial que se aplica con éxito a la función de optimización continua.

Para mejorar el comportamiento del BA para problemas de dimensiones superiores. [3] hibrída el algoritmo original de murciélagos usando estrategias de evolución diferencial.

El presente trabajo se divide en secciones, siendo que cada una abarca el desarrollo de un punto del trabajo replicado. La primera describe la matemática del método y las variantes realizadas sobre el mismo. La segunda presenta los detalles del modelado de las funciones *benchmark* solicitadas. La tercera contiene el principio de funcionamiento del código implementado y los procesos que se tomarón en cuenta para la sintonización de parámetros. Finalmente, la sección cuarta presenta el ranking de desempeo obtenido por la *prueba de suma de rangos de Wilcoxon*.

2. Descripción del algoritmo

Para explicar el algoritmo HBA, primero debemos explicar los elementos que lo componen. Primero el BA original y despúes el módelo de evolución diferencial.

2.1. Bat Algorithm

El algoritmo explota el módelo de ecolocación de los murciélagos. Los murciélagos usan ecos de sonar para detectar y evitar obstáculos. En general, se sabe que los pulsos de sonido se transforman en una frecuencia que se refleja en los obstáculos. Los murciélagos navegan usando el tiempo de retraso de la emisión a la reflexión. Ellos típicamente emiten impulsos de sonido cortos y fuertes. La frecuencia del pulso es generalmente de 10 a 20 veces por segundo. Después de golpear y reflexionar, los murciélagos transforman su propio pulso en informacin útil para medir la distancia a la presa. Los murciélagos usan longitudes de onda que varan en el rango de 0.7 a 17 mm o frecuencias de entrada de 20-500 kHz.

Para implementar el algoritmo, la frecuencia del pulso y tasa deben estar definidos. La frecuencia del pulso puede ser simplemente determinado en el rango de 0 a 1, donde 0 significa que no hay emisión y 1 significa que los murciélagos emiten sus máximos.

El comportamiento de los murciélagos se puede utilizar para formular el BA. Yang [35] us tres reglas generalizadas al implementar su algoritmo:

- Todos los murciélagos usan una ecolocación para detectar la distancia y también adivinan la diferencia entre el objetivo/ presa y las barreras de fondo.
- Al buscar su objetivo, los murciélagos vuelan al azar con velocidad V_i en la posicin X_i con frecuencia fija f_{min}, longitud de onda variable λ y volumen de emisión A_0. Pueden automáticamente ajustar la longitud de onda (o frecuencia) de sus pulsos emitidos y ajustar la tasa de emisión de impulsos $r \in [0, 1]$, dependiendo de la proximidad de su objetivo.
- Aunque el volumen puede variar de muchas maneras, suponemos que vara de un variable A_0 a un valor constante mínimo A_{min}.

El siguiente pseudocódigo presenta los pasos antes descritos en orden:

Algorithm Bat Algorithm:
Entrada: función objetivo $f(X)$, $X = (x_1, ..., x_d)$.
Inicializar los X_i y V_i para todos los murciélagos.
Definir los rangos de las frecuencias $Q_i \in [Q_{min}, Q_{max}]$.
Inicializar la tasa de emisión de impulsos r_i.
Inicializar el volumen A_i.
while t $< T_{max}$ **do**
  Generar nuevas soluciones ajustando la frecuencia
  y actualizando los X_i y V_i.
  if rand(0, 1) $> r_i$ **then**
    Selecciona una solución vecina a la mejor solución.
  end if
  Genera una nueva solución volando aleatoriamente.
  if rand(0, 1) $< A_i$ y $f(x_i) < f(x)$ **then**
    Aceptar la nueva solución.
    Aumentar r_i y reducir A_i
  end if
  Rankear las soluciones y encontrar la mejor actual
end while
return Mejor solución X_{best}
.

La inicialización de los X_i y V_i se realiza de manera random. Para la actualización de ambos valores se utilizan las ecuaciones presentadas en la ecuación 1.

$$\begin{aligned} Q_i^t &= Q_{min} + (Q_{max} - Q_{min})U(0,1) \\ V_i^{t+1} &= V_i^t + (X_i^t - best)Q_i^t \\ X_i^{t+1} &= X_i^t + V_i^t \end{aligned} \tag{1}$$

Donde $U(0,1)$ es una distribución. Una caminata aleatoria con explotación directa es usada para la búsqueda local, lo cual modifica la mejor solución actual ($best$) de acuerdo a la ecuación 3.

$$X_i^t = best + eA_i^t(2U(0,1) - 1) \tag{2}$$

Donde e es el factor de escala y A_i^t es el volumen. La búsqueda local se realiza con una proximidad dependiendo de la tasa de emisión r_i. Uno de los pasos mencionados en el algoritmo es aumentar la velocidad de emisión del pulso r_i y reducir el volumen A_i cuando el murcilago encuentra su presa, algo que ocurre en la naturaleza de la misma forma. Matemáticamente, estas características se capturan con las siguientes ecuaciones:

$$\begin{aligned} A_i^{t+1} &= \alpha A_i^t \\ r_i^t &= r_i^0[1 - exp(-\gamma e)] \end{aligned} \tag{3}$$

Donde α y γ.

2.2. Evolución diferenciada

La evolución diferencial (DE) es una técnica para la optimización introducida por Storn y Price en 1995 [4]. DE optimiza un problema manteniendo una poblacin de soluciones candidatas y crea nuevas soluciones candidatas combinando las existentes de acuerdo a sus reglas simples, y luego mantiene la solucin candidata que tenga la mejor puntuacin o aptitud en el problema de optimizacin disponible.

La búsqueda de soluciones en la vecindad de la mejor solución ($best$) es el punto sobre el que se realiza la mejora con DE, dado que es el proceso de búsqueda local. Para ello se realiza una mutación, cruce y selección diferenciada. En particular, la mutación diferencial selecciona al azar dos soluciones y agrega una diferencia escalada entre estas a la tercera solución. Esta mutación se puede expresar de la siguiente manera:

$$u_i^t = w_{r0}^t + F * (w_{r1}^t - w_{r2}^t) \tag{4}$$

Donde $F \in [0.1, 1.0]$ es el factor de escalamiento, mientras que r_0, r_1 y r_2 son soluciones seleccionadas de forma aleatoria en el espacio de soluciones.

Un cruce uniforme se emplea como cruce diferencial. El vector de prueba está formado por valores de parámetros copiados de dos soluciones diferentes. Matemáticamente, este cruce se puede expresar de la siguiente manera:

$$z_{i,j} = \begin{cases} u_{i,j}^t & rand_j(0,1) \leq CR \vee j = j_{rand} \\ w_{i,j}^t & otherwise \end{cases} \tag{5}$$

Donde CR $\in [0.0, 1.0]$ controla la fracción de parámetros que se copian a la solución de prueba. Siendo que $j = j_{rand}$ asegura que el vector de prueba es diferente a la solución original Y^t.

Matemáticamente, la selección diferencial se puede expresar de la siguiente manera:

$$w_i^{t+1} = \begin{cases} z_i^t & iff(Z^t) \leq f(Y_i^t) \\ w_i^t & otherwise \end{cases} \tag{6}$$

En el sentido técnico, el cruce y la mutación se pueden realizar de muchas maneras en DE. Por lo tanto, se usó una notación específica para describir la variedad de estos métodos (también estrategias) en general. Por ejemplo, "**DE / rand / 1 / bin**" denota que el vector base se selecciona al azar, se le agrega una diferencia de vector y el número de parámetros modificados en el vector de mutación sigue la distribución binomial.

2.3. Hybrid Bat Algorithm

Al utilizar DE en la búsqueda local del BA, se obtiene el siguiente pseudo-código:

Algorithm Hybrid Bat Algorithm:
Entrada: función objetivo $f(X)$, $X = (x_1, ..., x_d)$.
Inicializar los X_i y V_i para todos los murciélagos.
Definir los rangos de las frecuencias $Q_i \in [Q_{min}, Q_{max}]$.
Inicializar la tasa de emisión de impulsos r_i.
Inicializar el volumen A_i.
while t $< T_{max}$ **do**
  Generar nuevas soluciones ajustando la frecuencia
  y actualizando los X_i y V_i.

if rand(0, 1) $> r_i$ **then**
Modifica la solución uzando **DE / rand / 1 / bin**.
end if
Genera una nueva solución volando aleatoriamente.
if rand(0, 1) $< A_i$ y $f(x_i) < f(x)$ **then**
Aceptar la nueva solución.
Aumentar r_i y reducir A_i
end if
Rankear las soluciones y encontrar la mejor actual
end while
return Mejor solución X_{best}

3. Funciones *benchmark*

Las funciones *benchmark* solicitadas fueron modeladas como las funciones de costo a minimizar, Por lo que en la evaluación del fitness de cada murciélago se tomo en cuenta el valor de salida en cada función. Estas funciones son:

3.1. Función Ackley

La función Ackley es ampliamente utilizada para probar algoritmos de optimización. En su forma bidimensional, se caracteriza por una región exterior casi plana y un gran agujero en el centro. La función plantea el riesgo para los algoritmos de optimización, particularmente los algoritmos *hill climbing*, queden atrapados en uno de sus muchos mínimos locales. Su ecuación es:

$$f_1(x) = -a * exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - exp\left(\frac{1}{d}\sum_{i=1}^{d} cos(cx_i)\right) + a + exp(1)$$

Siendo su dominio $x_i \in [-32.768, 32.768]\ \forall i = 1, ..., d$ y su mínimo global $f_1(x^*) = 0$, $x^* = (0; ...; 0)$.

Para la implementación se utilizarón los valores a = 20, b = 0.2 y c = 2π.

3.2. Función Schwefel

La función Schwefel es compleja, con muchos locales mínimos. Su ecuación es:

$$f_2(x) = 418,9829d - \sum_{i=1}^{d} x_i sin(\sqrt{|x_i|})$$

Siendo su dominio $x_i \in [-500, 500]\ \forall i = 1, ..., d$ y su mínimo global $f_2(x^*) = 0$, $x^* = (420,9687; ...; 420,9687)$.

3.3. Función 3

Esta función se caracteriza por formar ondas de máximos y mínimos locales a partir de la siguiente ecuacin:

$$f_3(x) = 0.5 - \frac{\left(sin\left(\sqrt{\sum_{i=1}^{d} x_i^2}\right)\right)^2 - 0.5}{\left(1.0 + 0.001\left(\sum_{i=1}^{d} x_i^2\right)\right)^2}$$

Siendo su dominio $x_i \in [-100, 100]\ \forall i = 1, ..., d$ y su máximo global $f_2(x^*) = 1$, $x^* = (0; ...; 0)$.

Para la implementación se evaluo la función $-f_3(x)$ de modo que el algoritmo siga buscando el mnimo local en lugar del máximo.

4. Detalles de la implementación

En esta sección se presentan los detalles de la implementación incluyendo la sintonización de parámetros y la paralelización.

4.1. Sintonización de parámetros

Para la implementación del HBA se tiene que tener en cuenta que este tiene valores a sintonizar, los cuales son: A_o, r_o, Q_{min}, Q_{max}, c, α, γ y NP en el BA y, F y CR en el DE. Lo que hace un total de 10 valores a sintonizar.

Los primeros 4 (A_o, r_o, Q_{min} y Q_{max}) son sugeridos en varios artículos [3] [5], siendo estos A_o = 0.5, r_o = 0.5, Q_{min} = 0.0 y Q_{max} = 2.0; aunque los elementos más importantes a variar entre estos son A_o y r_o por lo que se explora el mapa completo de soluciones para los rangos de 0 a 1 con paso de 0.1 para ambos casos, resultando imperceptible el resultado en funciones de baja dimensionalidad y ligeramente visibles en funciones de alta dimensionalidad. Aunque un ánalisis más detallado de lo mismo se presenta en [3] dado que es el autor original del artículo, de igual forma [5] utiliza los mismos parámetros al asegurar su eficiencia.

De igual forma los valores c, α y γ son obtenidos por ensayo y error y son α = 0.8, c = 0.1 y γ = 10. Para ello se considero el peso del factor de escala c y el α de reducción de volumen.

Para el cálculo del número de muerciélagos se decidio que hayan entre 1 a 10 muerciélagos por unidad cuadrada del dominio, esto es entre 1000 a 10000 muerciélagos en la compilación.

Por otro lado según [5] entre los 3 procesos del DE, es la mutación por lo que al ser F un factor aleatorio se lo restringe entre los valores $[0.2, 0.8]$ de forma randomica a diferencia de lo mencionado en [3].

4.2. Paralelización del algoritmo

Para el proceso de paralelización se utiliza **CUDA** sobre la tarjeta gráfica GPU. Para ello se divide por cantidad de datos (número de murciélagos) en cada thread de forma uniforme, luego se elige el mejor por torneo binario (para el análisis del *best*), todo ello muestra resultados en tiempo muy superiores para gran número de murciélagos.

5. Resultados

En esta sección se presentan los resultados de lo anteiormente expuesto

5.1. Estadísticas del HBA

En base a los parámetros sintonizados se realizan las pruebas estadísticas del algoritmo para 50 iteraciones, 400 murciélagos en 2 y 8 dimensiones (D) con 100 repeticiones del experimiento. Los valores comparativos de las mejores (B), promedio (A) y peores (W) resultados de cada función se presentan en la tabla 1.

	Ackley		**Schewfel**		**Función 3**	
D	2	8	2	8	2	8
B	1.32E-8	0.00455	2.55E-5	1.01E-4	1	0.98557
A	3.79E-2	0.62947	1.19E-4	1.01E-4	1	0.99109
W	3.58E-1	0.25564	4.72E-3	1.01E-4	1	0.99944

TABLE 1: Resultado comparativo para 50 iteraciones, 100 repeticiones y 10000 murciélagos

5.2. Prueba de suma de rangos de Wilcoxon

Ahora, sobre los resultados obtenidos se realiza una comparación con pruebas no parámetricas, esto ayuda a establecer la diferenciación entre 2 poblaciones independientes, para elegir la mejor. En la tabla 2 se presentan los resultados de la *prueba de suma de rangos de Wilcoxon* para los siguientes 9 métodos respecto al HBA con $\alpha = 0.05$, para la presentación de los resultados se marca como (+) si los resultados del HBA son mejores y (-) si son peores.

- Cuckoo Search via Levy Flights (CS).
- Bat-Inspired Algorithm (BA).
- Particle Swarm Optimization (PSO).
- Flower Pollination Algorithm (FPA).
- Dynamic Differential Evolution (DE).
- Gravitational Search Algorithm (GSA).
- Harmony Search (HS).
- GENOCOP: A Genetic Algorithm for Numerical Optimization Problems with Linear Constraints.
- Grey Wolf Optimizer (GWO).

6. Conclusiones y trabajos futuros

Se observa que lo que principalmente cambia el rendimiento del algoritmo es la inicialización, es decir para casos como los presentados en que caer en mínimos locales es muy sencillo, es necesario inicializar de manera uniforme el algoritmo para ello se requieren varios murciélagos en la distribución para aumentar la chance de acercarse al mínimo global.

La mejora del método se observa en para problemas de mayor dimensión como los presentados por [3] o [5].

Como se menciona en [5], el proceso DE que ms aporta a la diversidad de soluciones (y por lo tanto a escapar de mínimos locales) es la mutación diferenciada.

	Ackley		**Schewfel**		**Función 3**	
Método	2	8	2	8	2	8
CS	1.32E-8	0.00455	2.55E-5	1.01E-4	1	0.98557
BA	3.79E-2	0.62947	1.19E-4	1.01E-4	1	0.99109
PSO	3.58E-1	0.25564	4.72E-3	1.01E-4	1	0.99944
FPA	1.32E-8	0.00455	2.55E-5	1.01E-4	1	0.98557
DE	3.79E-2	0.62947	1.19E-4	1.01E-4	1	0.99109
GSA	3.58E-1	0.25564	4.72E-3	1.01E-4	1	0.99944
HS	1.32E-8	0.00455	2.55E-5	1.01E-4	1	0.98557
GENOCOP	3.79E-2	0.62947	1.19E-4	1.01E-4	1	0.99109
GWO	3.58E-1	0.25564	4.72E-3	1.01E-4	1	0.99944

TABLE 2: Resultado de la prueba de suma de rangos de Wilcoxon para $\alpha = 0.05$

Agradecimientos

Este trabajo fue apoyado por el fondo 234-2015-FONDECYT (Programa de maestría) de Cienciactiva del Consejo Nacional de Ciencia, Tecnología e Innovación tecnológica (CONCYTEC-PERú).

References

[1] X.S. Yang. Bat algorithm for multi-objective optimisation. International Journal of Bio-Inspired Computation, 3(5):267274, 2011.

[2] F. Neri and V. Tirronen. Recent advances in differential evolution: a survey and experimental analysis. Artificial Intelligence Review, 33(12):61106, 2010.

[3] I. Jr. Fister, D. Fister, X.-S Yang. A hybrid bat algorithm. Elektrotehniki vestnik, 2013, in press.

[4] R. Storn and K. Price. Differential evolutiona simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization, 11(4):341359, 1997.

[5] Ahmadianfar, Iman, Arash Adib, and Meysam Salarijazi. "Optimizing multireservoir operation: Hybrid of bat algorithm and differential evolution." Journal of Water Resources Planning and Management 142.2 (2015): 05015010.