Tarea 3. Evolución diferencial, LSHADE y APSO aplicado a función de 10 variables

Ciro Fabian Bermudez Marquez
INAOE
Mexico, Puebla
cirofabian.bermudez@gmail.com

***Resumen*—En este trabajo se pone a prueba el algoritmo de evolución diferencial (ED), LSHADE y APSO para minimizar una función de 10 variables.**

***Index Terms*—ED, python, heuristica.**

I. Descripción del problema

Utilizando los algoritmos de evolución diferencial (ED), LSHADE y APSO resolver el problema de minimizar la función de 10 variables $f(\mathbf{x})$ que se muestra en la ecuación (1).

$$f(\mathbf{x}) = 0.1\sum_{i=1}^{10}(x_i - 2)(x_i - 5) + \sin(1.5\pi x_i) \qquad (1)$$

donde $\mathbf{x} = [x_1, x_2, \cdots, x_{10}]^T$.

II. Evolución Diferencial (ED)

La evolución diferencial usa números reales en su presentación de las variables, en general realiza un mejor trabajo que el algoritmo genético y se pude usar una condición automática de paro. Las heurísticas se deben usar si el problema tiene tres o más variables o si se trata de problemas multimodales.

El algoritmo requiere las siguientes variables para su funcionamiento:

- Tamaño de población μ
- Numero de generaciones g
- Valor del constante de recombinación R
- Valor del constante de diferencias F
- Valor del umbral s.

Para un problema de n variables el tamaño de población debe ser igual a $10n$. Es importante resaltar que si R, F las cuales se encuentran en el rango de $[0, 1]$ son iguales a 1 se está usando una búsqueda aleatoria. El parámetro s está definido en el espacio de la función. Si se tiene un umbral u significa $0.1u$ en las variables. Si se tiene una precisión 1×10^{-5}, equivale a 1×10^{-6} en las variables.

Las especificaciones a tener en cuenta para esta heurística son $F = 0.2$, $R = 0.4$, $\mu = 100$, $g = 30, 50, 100, 150$, espacio de búsqueda x_i en el rango $[0, 7]$ para todas las $i = \{1, 2, \ldots, 10\}$.

Se repitió 100 veces esta heurística variando el número de generaciones y se ingreso la función objetivo al archivo **evalua.py** de la siguiente manera:

```python
import math

def Evalua( _n, vpar ) :
  suma = 0
  for i in range(_n):
    suma += (vpar[i]-2)*(vpar[i]-5) + math.sin
    ( 1.5*math.pi*vpar[i])
    v = 0.1*suma
  return v
```

Código 1. Función objetivo.

El mínimo global de la función se encuentra en $x_i = 3.652887442162$ para toda $i = \{1, 2, \cdots, 10\}$, y $f(\mathbf{x}) = -3.224518019$, para determinar si el algoritmo encontró el valor de la función objetivo se utiliza el siguiente criterio:

$$|f_{\text{algoritmo}} - f_{\text{objetivo}}| < 1e-4 \qquad (2)$$

En la Tabla I se muestran los resultados de aplicar la heurística de ED.

Tabla I
Resultados de algoritmo ED en 100 repeticiones.

# Generaciones	Eficiencia %
50	0
60	0
70	14
80	95
90	100
100	100
150	100

III. LSHADE

El algoritmo LSHADE es un algoritmo autoajustable y que unicamente requiere de los siguientes parámetros:

- Número de variables del problema
- Número máximo de evaluaciones
- La semilla
- Tamaño de población

Se modifico la función objetivo de la siguiente manera:

```cpp
#include <math.h>
#define PI
     3.14159265358979323846264338327950 29
double evaluate( double *vx, int n )
{
  //double x = vx[0];
  double y = 0.0;
```

```
8    for (int i = 0; i <= 9; i++) {
9      y += (vx[i]-2.0)*(vx[i]-5.0) + sin(1.5*PI*
       vx[i]);
10       }
11
12   y *= 0.1;
13   return( y );
14 }
```

Código 2. Función objetivo para LSHADE.

El número máximo de evaluaciones recomendado para el algoritmo es igual a 2000 veces el número de variables del problema y el tamaño de población igual a 18 veces el número de variables del problema, (estas recomendaciones se encuentran descritas en el archivo **main.cc**) sin embargo para poner a prueba el algoritmo se buscó el mínimo valor para el cual presenta resultados aceptables, utilizando el mismo criterio para determinar si el algoritmo encontró el valor de la función objetivo en la Tabla II se muestran los resultados.

Tabla II
RESULTADOS DE ALGORITMO LSHADE EN 100 REPETICIONES.

# Max Evaluaciones	Eficiencia %
4500	0
4600	0
4800	100
5000	100
5200	100

IV. APSO

Para este algoritmo es necesario ingresar los siguientes parámetros

- Tamaño de población
- El número de generaciones
- El número de variables
- Parámetros de control $\alpha = 1$ y $\beta = 0.5$

Para poder comparar este algoritmo con la evolución diferencial se eligió un tamaño de población de 100 individuos y variar el número de generaciones entre 50 y 200. De mismo modo esto se repetio 100 veces y se utilizó el mismo criterio para determinar si el algoritmo encontró el valor de la función objetivo, los resultados se muestran en la Tabla III.

Tabla III
RESULTADOS DE ALGORITMO APSO EN 100 REPETICIONES Y POBLACIÓN DE 100 INDIVIDUOS.

# Generaciones	Eficiencia %
50	0
100	0
150	0
200	0

V. CONCLUSIONES

Hay que resaltar que tanto el algoritmo de ED como el APSO se probaron con una codificación en python y debido a esto el tiempo para ejecutar las 100 repeticiones fue mucho más largo que el algoritmo LSHADE el cual estuvo codificado en C++. De esta observación se puede recomendar que para probar algoritmos es muy buena idea hacerlo en python debido a la facilidad de codificación y lo rápido de desarrollar plataformas de prueba sin embargo una vez comprobado el funcionamiento correcto del algoritmo es sumamente recomendado pasar el algoritmo a C/C++ debido a la rapidez en el tiempo de ejecución. De los tres algoritmos tanto el LSHADE como la ED fueron buenos en encontrar la solución a nuestro problema sin embargo, LSHADE tiene la ventaja de tanto ejecutarse más rápido como requerir menos parámetros debido a su característica de ser autoajustable por lo que fue el mejor de los tres. En cuanto APSO este algoritmo presento diversas peculiaridades, para una población de 100 individuos y variando el número de generaciones no encontró ninguna solución, esto podría deberse a los parámetros de control los cuales no se modificaron se sus valores por defecto.

REFERENCIAS

[1] Dr. Luis Gerardo de la Fraga. “Apuntes de clase” .