Tarea 2. Algoritmo genético

Ciro Fabián Bermúdez Márquez
INAOE
Mexico, Puebla
cirofabian.bermudez@gmail.com

***Resumen*—En este trabajo se pone a prueba el algoritmo genético (AG) para un problema de minimización de una función de 10 variables.**

I. Descripción del problema

Utilizando el algoritmo genético (AG) resolver el problema de minimizar la función de 10 variables $f(\mathbf{x})$ que se muestra en la ecuación 1 teniendo en cuenta los siguientes criterios:

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

$p_c = 0.7$, $p_m = 0.1$, $\mu = 200$, $g = 50, 100, 150$, codificación de 10 bits, espacio de búsqueda x_i en el rango $[0, 7]$ para todas las $i = \{1, 2, \ldots, 10\}$, donde $\mathbf{x} = [x_1, x_2, \cdots, x_{10}]^T$.

II. El algoritmo genético

Los métodos de búsqueda heurística consisten en añadir información, basándose en el espacio estudiado hasta ese momento para hacer una búsqueda inteligente. Para problemas no lineales y multimodales se justifica usar heurísticas de mismo modo para problemas que tiene tres o más variables.

Para entender el algoritmo genético es necesario conocer la terminología que se utiliza.

Individuo: es un vector que codifica las variables del problema.

Población: una matriz, un conjunto de vectores.

Las variables para el AG se codifican en cadenas binarias, de 0 o 1. Para la representación binaria se utiliza la codificación Grey, ya que esta solo cambia un bit entre cada valor. El AG codifica problemas combinarlos, no continuos.

Para cambia una cadena binaria a una variable real, se puede mapear linealmente:

$$\frac{x - \min_x}{\max_x - \min_x} = \frac{b}{2^p - 1} \qquad (2)$$

donde la precisión es igual a

$$\text{precisión} = \frac{\max_x - \min_x}{2^p} \qquad (3)$$

y p es el número de bits.

A grandes rasgos la filosofía del algoritmo genético se puede resumir en los siguientes pasos:

- Utiliza un conjunto de soluciones (se le llama población).
- Combina soluciones para generar otras.
- La solución que sobrevive es la mejor (la más apta).
- Usa los operadores de selección, cruza, mutación y elitismo.
- La población guarda la *inteligencia* del algoritmo.

El algoritmo requiere las siguientes variables para su funcionamiento:

- Tamaño de población μ
- Numero de generaciones g
- Probabilidad de cruza p_c
- Probabilidad de mutación p_m
- La función a optimizar.

En orden el AG realiza las siguientes operaciones:

1. Se inicializa aleatoriamente la población
2. Se evaluá la población
3. Para un número de generaciones:
   a) Se seleccionan dos individuos
   b) Se cruzan
   c) Su mutan y se evalúan los hijos
   d) Se aplican elitismo
   e) La población de hijos sustituye a la de padres
4. Se reporta el mejor individuo

II-A. Detalles de pasos de AG

La selección, la cruza, la mutación y el elitismo son operadores genéticos.

La selección de los individuos se realiza por medio de un **torneo binario**.

i1 = Se escogen dos individuos y gana el mejor

i2 = Se escogen dos individuos y gana el mejor

Al inicio de cada iteración se revuelven aleatoriamente la población (se barajean los índices). Y se van tomando de dos en dos.

La **cruza de dos puntos** consisten en intercambiar partes de las cadenas binarias entre padres e hijos y se usa una probabilidad de cruza en [0,1].

En la **mutación** se escoge aleatoriamente una posición de la cadena y se invierte el bit. Se aplica con un *probabilidad de mutación* que toma valores en [0,1].

Algo importante a resaltar es que si la probabilidad de cruza y mutación son iguales a 1, el algoritmo está haciendo una búsqueda aleatoria. Es recomendado que la probabilidad de mutación este en un valor menor que 0.2.

El **elitismo** consiste en guardar la mejor solución para garantizar convergencia, el mejor individuo se mantiene en la población.

El precio de usar una heurística es que la función del problema se tiene que ejecutar el número de generaciones multiplicada por el número de individuos.

III. Resultados

Utilizando los parámetros descritos en la sección descripción del problema y modificando el archivo **evalua.py** de la siguiente manera:

```
import math

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

Código 1. Función de 10 variables.

se realizaron 100 pruebas para cada uno de los números de generaciones, el mínimo global de la función se encuentra en $x_i = 3.652887442162$ para toda i, y $f(\mathbf{x}) = -3.224518019$, para saber si el algoritmo encontró el valor de la función objetivo se utiliza el siguiente criterio:

$$|f_{\text{algoritmo}} - f_{\text{objetivo}}| < 1e-4 \tag{4}$$

Los resultados se muestran en la Tabla I.

Tabla I
Resultados de AG para encontrar el mínimo global de la función repetido 100 veces.

# Generaciones	Eficiencia %
50	0
60	0
70	4
80	22
100	85
150	97
200	99

IV. Conclusiones

Cuando se tienen arriba de 100 generaciones el porcentaje de acierto es mayor al 85 porciento y para generaciones arriba de 150 la diferencia es mínima. A diferencia del método de Newton para encontrar máximos y mínimos que tenia una eficiencia del 9 % para el el caso de una variable, aplicado a este problema tendría una eficiencia de $(0.1)(0.09^{10})$ lo cual es $3.4e-10$ %, lo que demuestra la utilidad de utilizar este tipo de heurísticas.

Referencias

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