Centro de Investigación en Cómputo
Instituto Politécnico Nacional
Metaheurísticas
Actividad 18: Solución de problemas mediante Algoritmos Genéticos
Curso impartido por: Dra Yenny Villuendas Rey

Adrian González Pardo

22 de diciembre de 2020

1. Funciones a optimizar

Función a evaluar	Forma o descripción de la función
Alpine Function	$f_1(x) = \sum_{i=1}^{D} \lvert x_i \sin(x_i) + 0{,}1x_i \rvert$
Dixon & Price Function	$f_2(x) = (x_1 - 1)^2 \sum_{i=2}^{D} i \left(2\sin(x_i) - x_{i-1}\right)^2$
Quintic Function	$f_3(x) = \sum_{i=1}^{D} \left\lvert x_i^5 - 3x_i^4 + 4x_i^3 - 2x_i^2 - 10x_i - 4 \right\rvert$
Schwefel 2.23 Function	$f_4(x) = \sum_{i=1}^{D} x_i^{10}$
Streched V Sine Wave Function	$f_5(x) = \sum_{i=1}^{D-1} (x_{i+1}^2 + x_i^2)^{0,25} \left[\sin^2\left\{50(x_{i+1}^2 + x_i^2)^{0,1}\right\} + 1\right]$
Sum Squares Function	$f_6(x) = \sum_{i=1}^{D} i x_i^2$

2. Código de implementación

El código fue implementado en lenguaje Ruby para el cálculo de las funciones en D dimensiones y a traves de 1 indice se determina la selección de que función se trabajara.

2.1. Función que selecciona que $f(x)$ trabajara

```ruby
# Recibe indice de cual funcion trabajara
def get_eval(array=[],index=0)
  if index==0
    return evaluar_alpine(array)
  elsif index==1
    return evaluar_dixon(array)
  elsif index==2
    return evaluar_quintic(array)
  elsif index==3
    return evaluar_schwefel(array)
  elsif index==4
    return evaluar_streched(array)
  elsif index==5
    return evaluar_sum_squares(array)
  else
    return 2**1000
  end
end
```

2.2. Función Alpine

```ruby
def evaluar_alpine(array=[])
  func=0
  array.each{|i|
    func+= (i*Math.sin(i)+0.1*i).abs
  }
  func
end
```

2.3. Función Dixon & Price

```ruby
def evaluar_dixon(array=[])
  func=(array[0]-1)**2
  (@dimension-1).times{|i|
    func+=(i+1)*((2*Math.sin(array[i])-array[i-1])**2)
  }
  func
end
```

2.4. Función Quintic

```ruby
def evaluar_quintic(array=[])
  func=0
  array.each{|i|
    func+=((i**5)-3*(i**4)+4*(i**3)-2*(i**2)-10*i-4).abs
  }
  func
end
```

2.5. Función Schwefel

```ruby
def evaluar_schwefel(array=[])
  func=0
  array.each{|i|
    func+=i**10
```

```
  }
  func
end
```

2.6. Función Streched

```
def evaluar_streched(array=[])
  func=0
  (@dimension-2).times{|i|
    func+=(((array[i+1]**2)+(array[i]**2))**0.25)*((Math.sin(50*((array[i+1]**2)+(array[i]**2))
    **0.1))**2+0.1)
  }
  func
end
```

2.7. Función Sum Squares

```
def evaluar_sum_squares(array=[])
  func=0
  array.each_with_index{|i,j|
    func+= (j+1)*(i**2)
  }
  func
end
```

3. Características de Hardware

- Procesador Intel Core i5-3210M CPU 2.50GHz
- Memoria RAM DDR3 12 GB
- Sistema Operativo Fedora 32 x86_64
- Ruby 2.7.2p137 para la ejecución

4. Características de ejecución:

- Dimensiones de la función 10 y 30
- Máximo número de iteraciones 500
- Poblacion 10
- Variación en el tipo de algoritmo genético fue utilizado para la resolución de cada problema
  - *Alpine, Quintic, Streched* GA Estacionario
  - *Dixon, Schwefel, Sum Cuadrados* GA Generacional
- Variación en operador de seleccion por función
  - *Alpine, Schwefel* Selección por torneo
  - *Dixon, Streched* Selección por ruleta
  - *Quintic, Sum Cuadrados* Selección proporcional

D=10	Evaluación					Tiempo ms				
Función	Mejor	Peor	Promedio	Mediana	Desviación estandar	Mejor	Peor	Promedio	Mediana	Desviación estandar
Alpine	0.6	3.43	1.717	1.745	0.715570401847365	16.68	29.3	19.0175	18.145	2.94883176020607
Dixon	2	98	43	42	30.5794048339728	100	145.54	130.641	132.6	11.6651347613304
Quintic	0.46	0.78	0.554	0.51	0.096560861636587	8.33	23.04	16.9115	19.265	5.15444107057206
Schewel	0	97	45.9	45.5	30.7048856047372	80.67	198.18	151.919	157.035	28.522703921613
Streched	0.78	1.54	1.143	1.17	0.208305064748796	17.45	23.42	20.292	19.94	1.52302199590157
Sum Squares	0	90	51.25	54.5	28.4374313185984	48.02	188.07	106.0265	103.845	39.888168957098
Promedio	0.64	48.46	23.927	24.2375	15.1236930142569	45.192	101.258	74.134583	75.13833	14.9503837444535

D=30	Evaluación					Tiempo ms				
Función	Mejor	Peor	Promedio	Mediana	Desviación estandar	Mejor	Peor	Promedio	Mediana	Desviación estandar
Alpine	0.51	4.59	2.7245	2.565	1.06767726865378	16.68	52.14	32.748	39.955	12.5141526281247
Dixon	1	90	52.3	54	24.0127049704943	48.86	181.91	111.2785	130.12	39.2905133938207
Quintic	0.38	0.85	0.5825	0.55	0.126999015744217	8.85	29.77	18.369	19.81	5.90410441980831
Schewel	3	94	49.8	52	31.7499606298968	123.12	195.57	157.178	160.805	19.1750401824872
Streched	0.83	1.61	1.1915	1.175	0.215459392925906	18.13	22.95	19.8525	19.45	1.2278349848412
Sum Squares	0	92	48.5	45	26.7721870604551	62.41	180.53	129.507	132.13	25.5451872375209
Promedio	0.9533	47.18	25.84975	25.88166667	13.990831389695	46.342	110.478	78.1555	83.71167	17.2761388077672

Figura 1: Tabla de resultados obtenidos de los algoritmos genéticos

5. Conclusiones

La implementación y planteamiento de nuevos operadores que permite crear o encontrar soluciones a funciones cuyos puntos críticos de minimización están definidos en intervalos muy fijos, fueron encontrados gracias a operadores aleatorios y fijos que permiten apoyar a la mutación de cada componente de la función a converger de forma rápida o esperada en un valor cercano a 0 de modo en que es sencillo de pensar que se puede computar de forma sencilla y que se obtenga una respuesta teóricamente rápida y aceptable.