

Centro de Investigación en Cómputo  
Instituto Politécnico Nacional  
Metaheurísticas  
Actividad No. 13  
Solución de problemas mediante Algoritmos Genéticos GA  
Curso impartido por: Dra Yenny Villuendas Rey

Adrian González Pardo

18 de noviembre de 2020

## 1. Ventajas y Desventajas de GA

Ventajas	Desventajas
Permite realizar multiples busqueda de soluciones a los problemas	Puede que el metodo de mutación o cruza seleccionado puede que no ayude a encontrar una buena solución
Esta bioinspirado en la genética y en la selección natural (Darwinismo)	Puede que el que en la aplicación en alguna etapa del GA ya no avance
Es una heurística poblacional	Puede que genere bastante uso de recursos en memoria y procesamiento
Es posible el trabajar soluciones de formas paralelizables o distribuidas	Puede ser difícil de implementar

## 2. Genotipo vs Fenotipo

**Genotipo:** es una representación en cadenas de bits en la cual generalmente es trabajada para generar un nuevo individuo en el algoritmo.

**Fenotipo:** es la representación que tiene la cadena de bits en el ambito del problema, es decir, la cadena de bits puede representar números reales  $\mathbb{R}$ , números enteros  $\mathbb{Z}$ , valores binarios  $\{0, 1\}$ , índices de la solución a algún problema.

## 3. Modelo Generacional vs Estacionario

### 3.1. Semejanzas

- Ambos generan en cada iteración nuevas respuesta a analizar
- Ambos rempazan a la generación anterior (Con precaución de como son seleccionados y de como trabajan en la siguiente iteración).
- Ambos realizan conceptualmente las mismas operaciones (Solo que de diferente manera a la hora de selección y cruza).

### 3.2. Diferencias

- El modelo generacional crea una nueva población completa, mientras que el modelo estacionario escoje dos partes de la población de acuerdo al muestreo que realice y sobre ellos aplica los operadores genéticos.

- El modelo generacional reemplaza completamente a la anterior generación, mientras que el modelo estacionario reemplaza a los  $M$  cromosomas con los  $N$  descendientes de la población inicial  $M \leq N$ .
- El modelo generacional tiene un reemplazo aleatorio, mientras que el modelo estacionario reemplaza a los  $N$  peores.
- El modelo generacional teóricamente realiza una excesiva exploración lo cual no garantiza que tenga una convergencia en un óptimo local (Explora espacialmente las regiones de solución), el modelo estacionario realiza una excesiva explotación lo cual converge en un óptimo local (Busca mejorar al mejor individuo).

## 4. Operadores de selección

### 4.1. Muestreo Aleatorio Universal (SUS)

El modelo de selección se trata de seleccionar varios elementos de acuerdo al tamaño de la población  $N$  dado que se obtiene una circunferencia donde existen varias flechas de selección dadas por  $\frac{2\pi}{N}$  donde se busca girar todas las flechas uniformemente de tal manera que el giro y la selección se determina por  $\left[ rand \times \left( \frac{2\pi}{N} \right) \right]$  donde  $rand \sim U(0,1)$ , el cual al ser una parte universal elimina el sesgo probabilístico a la hora de seleccionar los elementos.

### 4.2. Torneo

Este modelo se trata de ir realizando un  $k$  número de arreglos permutados de forma aleatoria de tamaño  $N$  que es el tamaño dimensional de la respuesta de tal modo en que se busca seleccionar el valor índice de aquellos datos que estén mejor evaluados en el torneo, gráficamente el torneo se ve de la siguiente manera:

$k = 2$ , entonces  $[0,5, 1, 4, 0,1]$ , la generación aleatoria, generaría lo siguiente por ejemplo:

2	1	4	3
4	2	3	1

*Se accedera a los valores de cada subíndice del arreglo en su valor y se evaluara el menor valor vía índices es decir de arriba hacia abajo para seleccionar al mejor.*

4	1	4	1
---	---	---	---

### 4.3. Proporcional

Para este tipo de selección se usa la función

$$p_i = \frac{f_i}{\sum_{j=1}^N f_j}$$

Donde:

$p_i$  es la probabilidad de ser seleccionado  $p_i$

$f_i$  es la función de fitness para ese valor de  $i$

Por tanto al obtener esto tendremos un pequeño sesgo probabilístico a la hora de seleccionar un área por tanto definiremos al error selectivo como:

$$n_i = p_i \times N = \left\lfloor N \times \frac{f_i}{\sum_{j=1}^N f_j} \right\rfloor$$

En esta ecuación nos podremos percatar de que redondeamos el error a un valor entero, lo que significa que cada individuo podría perderse cualquiera que sea su valor relativo de aptitud.

### 4.4. Por Ruleta

Para determinar este modelo se considera que la probabilidad para generar la circunferencia con ayuda de la ecuación  $p_i$  de la proporcional pero con la ayuda de la selección de 1 elemento a diferencia de como lo hace el Muestreo Universal que selecciona  $K$  elementos.

## 4.5. Por Emparejamiento Variado Inverso (NAM)

Un padre se escoge aleatoriamente, para el otro selecciona  $N_{nam}$  padres y escoge el más lejano al primer ( $N_{nam} = 3, 5, \dots$ ). Está orientado a generar diversidad.

## 5. Estructuras de datos necesarias para la implementación

Para obtener una mejor selección de nuestras soluciones es necesario hacer notar que muchas de estas soluciones son acompañadas del valor índice podemos pensar en un comportamiento de una cola circular de modo en que podamos realizar permutaciones y giros de acuerdo a como nos plasca, por otro lado también podemos pensar en hacer uso de arreglos de dimensión  $k$  donde  $k \leq N$  de modo en que podamos crear una cantidad de arreglos y permutarlos de forma aleatoria para implementar gran cantidad de nuevos elementos, al menos para lograr implementar una selección por torneo, mientras que por otro lado podemos obtener la cola donde cada índice sea un valor a la solución total podemos pensar en que este nos ayuda a implementar las funciones de probabilidad de modo que podemos ahorrarnos bastante tiempo a la hora de realizar las evaluaciones.

## 6. Implementación

```
1  #!/usr/bin/env ruby
2
3  class Seleccion
4    attr_accessor :array
5    def initialize(array=[])
6      """
7      @array -> es el arreglo el cual contiene las evaluaciones de la funcion a
8              optimizar
9      """
10     @array=array
11   end
12
13   def sum_fitness()
14     sum=0
15     @array.length.times{|i|
16       sum+=array[i]
17     }
18     sum
19   end
20
21   def gen_permutacion()
22     permutacion=[]
23     (@array.length).times{|i|
24       permutacion.push(i)
25     }
26
27     (@array.length).times{
28       cambia=rand(@array.length-1)
29       cambia1=rand(@array.length-1)
30       val0=permutacion[cambia]
31       val1=permutacion[cambia1]
32       permutacion[cambia1]=val0
33       permutacion[cambia]=val1
34     }
35     puts "Permutacion obtenida\t#{permutacion.to_s}"
36     permutacion
37   end
38
39   def torneo()
40     k=rand(@array.length)
41     puts "Valor de cuantas iteraciones #{k}"
42     valores=[]
43     for i in 0..k-1
44       valores.push(gen_permutacion)
45     end
46
47     mejor=valores.pop
```

```

48     valores.each{|val|
49         (@array.length-1).times{|i|
50             if @array[mejor[i]]>@array[val[i]]
51                 mejor[i]=val[i]
52             end
53         }
54     }
55     mejor
56 end
57
58 def ruleta()
59     sum_t=sum_fitness()
60     pi=[]
61     @array.length.times{|i|
62         pi.push(@array[i].to_f/sum_t)
63     }
64     puts "Parte ruleta de cada pi #{pi.to_s}"
65     piece=2*Math::PI/@array.length
66     arrow_init=rand(0.0..1.0)*piece
67     puts "Inicio en la ruleta #{arrow_init}"
68     sum_acumulada=0
69     mejor_respuesta=[]
70     pi.length.times{|j|
71         sum_acumulada+=pi[j]
72         if sum_acumulada>=arrow_init
73             return j
74         end
75     }
76 end
77
78 def nam()
79     steps=rand(2..(@array.length/rand(2..3)))
80     mejor_respuesta=[]
81     init=rand(@array.length-1)
82     for i in (init..@array.length).step(steps)
83         mejor_respuesta.push(i)
84     end
85     mejor_respuesta
86 end
87
88
89 end
90
91 array_fitness=[2,5,6,1,4,10]
92 puts "Fitness function\t#{array_fitness.to_s}"
93 s=Seleccion.new(array_fitness)
94 mejor_torneo=s.torneo
95 puts "\nElementos sel torneo\t#{mejor_torneo}\n\n"
96
97 sus=s.ruleta()
98 puts "\nMejor por ruleta\t#{sus}\n\n"
99
100 na=s.nam()
101 puts "Respuesta por Emparejamiento Inverso\t#{na.to_s}"

```