Centro de Investigación en Cómputo
Instituto Politécnico Nacional
Metaheurísticas
Actividad No. 17
Solución de problemas mediante Algoritmos Genéticos Generacionales
Curso impartido por: Dra Yenny Villuendas Rey

Adrian González Pardo

15 de diciembre 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 ámbito 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. Operadores

3.1. Mutación

3.1.1. Aleatoria para fenotipo real

Para este tipo de mutación es necesario conocer la dimensión del arreglo, de modo en que para la mutación de este tipo podemos hacer dos métodos distintos hacer la modificación mediante una selección aleatoria en cualquier x_i con valores aleatorios que van de $[min, max]$ y la otra que itera sobre cada x_i bajo una función de probabilidad cuyo valor es pequeño y si se genera un valor aleatorio cuyo valor compite en la función de probabilidad y este es menor se generara el valor aleatorio en el intervalo del método anterior.

3.1.2. Aleatoria para fenotipo binario

Para esta mutación se hará uso de un valor de probabilidad pequeño de modo en que se debe obtener un valor pequeño de tal modo que este significara que hay una modificación sobre cada bit particular $b_i = 0, 1$ de tal forma en que se puede realizar esto con cada índice b_i o se puede iterar sobre todos y cada uno de ellos

3.1.3. Aleatorio para fenotipo de orden

Para este tipo de mutación al igual que los anteriores podemos apoyarnos del uso del valor de probabilidad para mutar sobre dos valores p_i que intercambia lugar con p_j de tal modo en que mediante esta permutación podremos generar una nueva solución, por otro lado se puede realizar esta permutación una sola ocasión o realizarla como si se iterara sobre cada elemento de la solución es decir N veces

3.2. Cruzamiento

3.2.1. Dos puntos de corte

Es una forma de intercambiar información entre dos padres. Esta forma de cruce se selecciona dos puntos aleatorios de corte de forma que se los cortes van de $[1, a], [a, b], [b, N-1]$, quedando un ejemplo de la siguiente forma:

0	0	0	1	1	0	1	1
1	0	1	0	0	1	1	0

Pasan a:

0	0	0	1	0	1	1	1
1	0	1	0	1	0	1	0

3.2.2. Uniforme

Este tipo de cruzamiento se realiza bajo un vector con valores de probabilidad, destacando que cada elemento del arreglo tiene una probabilidad de cruza del 0,5 por lo que generaran valores aleatorios en el vector de tal manera que si $v_i \geq 0{,}5$ este índice realiza la cruza de datos de ese índice.
Un ejemplo de ello seria el siguiente: **Se genera un vector con los siguientes datos**

0.55	0.03	0.67	0.77	0.30	0.25	0.99	0.78

Teniendo los siguientes padres:

0	0	0	1	1	0	1	1
1	0	1	0	0	1	1	0

Entonces:

1	0	1	0	1	0	1	0
0	0	0	1	0	1	1	1

3.2.3. Aritmético

Para este tipo de algoritmo es necesario pensar en que nuestros dos vectores padres son valores reales los cuales van a ser cruzados a través de un valor aleatorio $\alpha \in [0, 1]$ de modo que actuaran como valores escalares para nuestros vectores, de tal forma que existe un segundo valor $\beta = 1 - \alpha$ formando la siguiente ecuación para realizar la cruza:

Sean los vectores

$$X^1 = x_0^1, x_1^1, \cdots, x_N^1$$
$$X^2 = x_0^2, x_1^2, \cdots, x_N^2$$
$$y = \alpha x^1 + \beta x^2$$

3.3. Selección

3.3.1. 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 accederá 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

3.3.2. 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.

3.3.3. Por Emparejamiento Variado Inverso (NAM)

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

4. Impacto en convergencia a la solución con GA

Como bien sabemos la existencia de un algoritmo bioinspirado en la selección natural y el la biología en si misma nos hace pensar que en cuestión de generación de soluciones ya sean aleatorias en cada nueva iteración o con alguna conservación en sus estados es de suma importancia ya que de este modo podemos explorar ya sea de forma continua o paralela el espacio de soluciones de modo en que se puede converger en la solución al problema de una forma más rápida y sencilla, por lo que a mayor tamaño de población podría significar mayor procesamiento pero si se implementa alguna técnica de paralelización pueda que en tiempo sea casi aproximado a las soluciones de los otros algoritmos heurísticos como lo son Random-Mutation Hill-Climbing (RMHC) o Simulated Annelling (SA). Para este caso es necesario guardar en memoria cual fue la mejor generación y la mejor respuesta arrojada por el programa, por lo que es necesario conocer y reconocer la implementación de algunas variables auxiales las cuales almacenaran los datos previamente utilizados

5. Estructuras de datos necesarias para la implementación en problemas

5.1. Knapsack

Modelación Matemática
Sean dos funciones

$$f(x) = \sum_{i=1}^{N} x_i^{'} h(x_i)$$

$$g(x) = \sum_{i=1}^{N} x_i^{'} p(x_i) \ \leq \ peso\_máximo$$

Donde:
$X\ es\ un\ vector\ de\ la\ forma\ X = [x_1, x_2, \cdots, x_N]$

$X^{'}\ es\ un\ vector\ el\ cual\ es\ parecido\ a\ X\ pero\ sus\ valores\ son\ binarios\ x_i^{'} \in \{0,1\}$

$h(x_i)\ es\ una\ función\ la\ cual\ devuelve\ el\ beneficio\ total\ de\ los\ objetos\ x_i\ en\ la\ mochila$

$p(x_i)\ es\ una\ función\ la\ cual\ devuelve\ el\ peso\ total\ de\ los\ objetos\ x_i\ en\ la\ mochila$

De modo que para los operadores GA en este problema el genotipo puede ser la cadena de bits de modo que el fenotipo que representa de la cadena es la representación de un valor binario de si el objeto es incluido o no es incluido en la solución.
Finalmente la estructura de datos para esta solución es un arreglo binario el cual contendrá un valor 0 o 1 de tal forma en que se pueda tener cada arreglo a la selección y torneo de GA, por tanto así podría calcularle los valores de su función de maximización de beneficio y de tratar de minimizar o no sobrepasar el peso máximo de la mochila

5.2. Travel Salesman Problem (TSP)

Modelación Matemática
Sea la función

$$f(x) = \sum_{i=1}^{N} x_i \cdot g\left(y_{x_i,i}\right)$$

Donde:
$Y\ es\ una\ matriz\ de\ NxN\ la\ cual\ trabajara\ para\ obtener\ el\ costo\ con\ la\ función\ g(x)$

$X\ es\ un\ vector\ de\ dimensión\ N\ el\ cual\ contiene\ el\ \acute{i}ndice\ del\ vértice\ al\ cual\ pasa\ de\ i\ a\ j$

$g(y_{x_i,i})\ es\ una\ función\ la\ cual\ devuelve\ el\ costo\ de\ ir\ de\ i\ hacia\ el\ contenido\ de\ x_i$

De modo en que nuestro GA puede ser implementado con un fenotipo entero el cual representa la ciudad n hacia que vértice se conecta posteriormente.

Ejemplo de arreglo X: $x = [2,3,1,0]$ de modo que sabemos que nuestro grafo tiene 4 aristas y pensando en que todos están conectados partiremos de la siguiente manera $(0 \to 2)$, $(2 \to 1)$, $(1 \to 3)$, $(3 \to 0)$ de tal forma que el formato en como realiza el movimiento de un vértice a otro es $(i \to j)$ siendo el valor de j el que sera evaluado en la matriz Y para obtener su costo.
Finalmente para este es tan sencillo como la realización de un arreglo de orden que contenga los valores de cada nodo o ciudad de la solución para así realizar el cálculo de su costo y así sobre cada solución.

5.3. Función de Minimización en D dimensiones

Modelación Matemática
Sea la función

$$f(x) = \sum_{i=1}^{D} x_i^2$$

Donde:
$x_i \in \mathbb{R}$

Descrito en los intervalos $x_i \in [-10, 10]$

Donde sabemos que los puntos mínimos de cada x_i los encontramos cuando el valor asignado a el es $x_i = 0$ por lo tanto el ir variando los valores para que el programa se acerque hacia 0

Por lo tanto para este GA podemos implementarlo con un fenotipo de valor real que vaya del intervalo indicado.
Para este problema podemos pensar en un arreglo de valores reales los cuales nos permitirán realizar o almacenar las soluciones para así pasar por cada etapa del GA.

6. Equipo de ejecución para la solución de los problemas

6.1. 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

7. Implementación e interfaces

Para el lenguaje seleccionado (Ruby) es difícil la realización de una interfaz gráfica por ello se implementa una interfaz por comandos o ventana la cual muestra el comportamiento de los programas

7.1. Knapsack

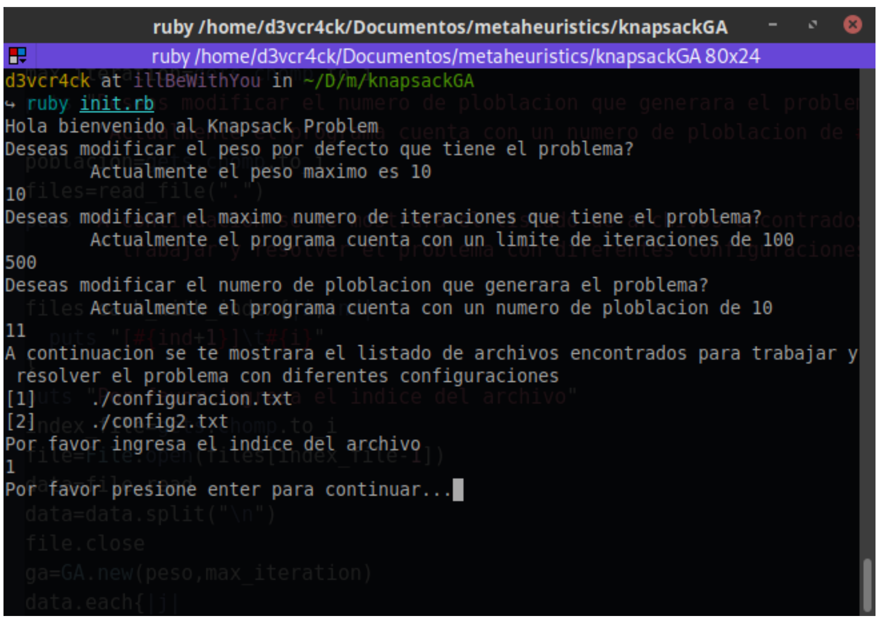

Figura 1: Ventana de la interfaz de datos solicitados para la solución

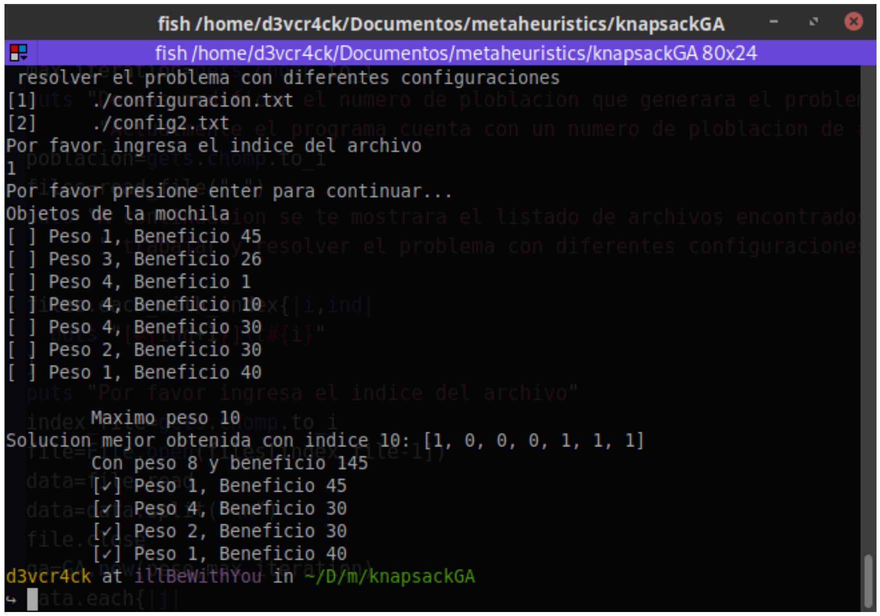

Figura 2: Ventana de la interfaz con la solución dada por el programa

7.2. Travel Salesman Problem (TSP)

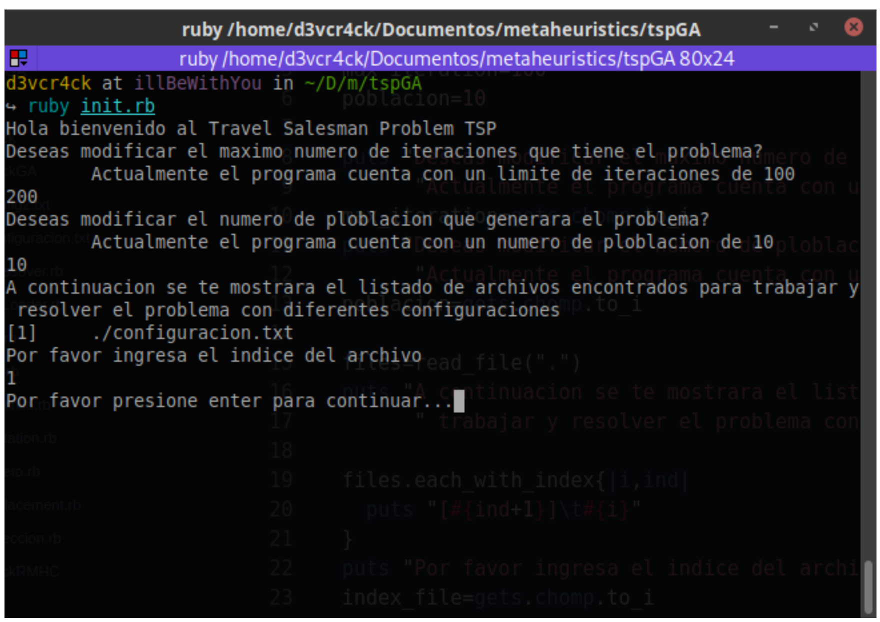

Figura 3: Ventana de la interfaz de datos solicitados para la solución

Figura 4: Ventana de la interfaz con la solución dada por el programa

7.3. Función de Minimización en D dimensiones

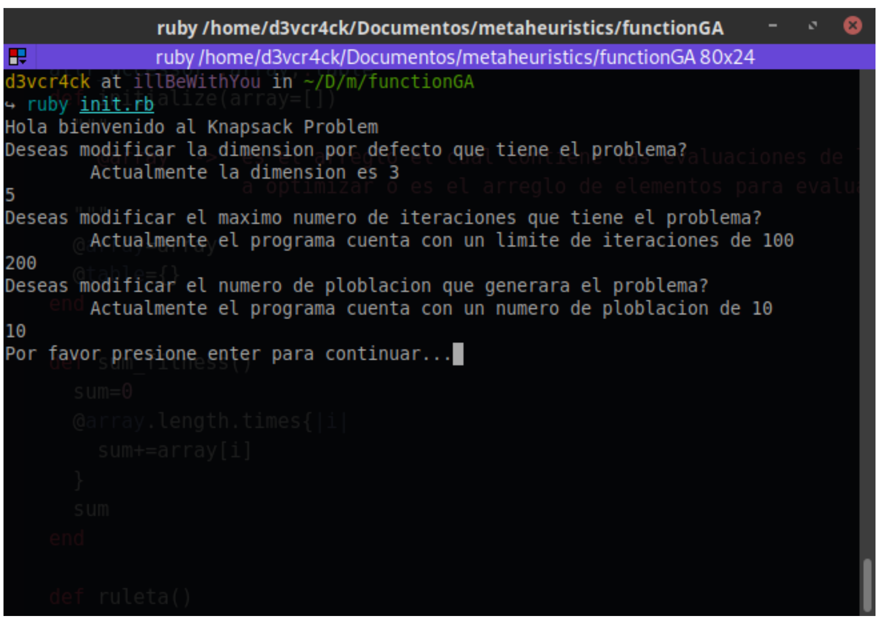

Figura 5: Ventana de la interfaz de datos solicitados para la solución

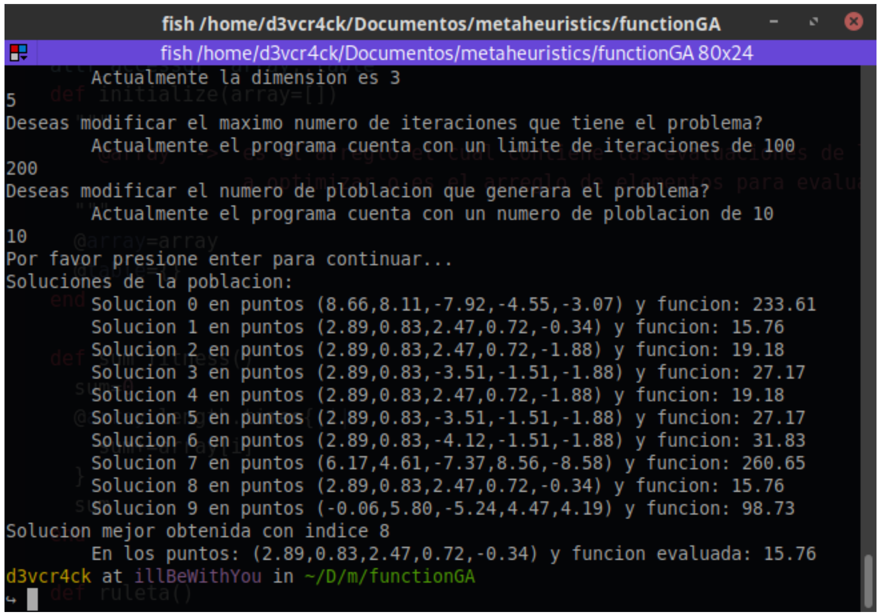

Figura 6: Ventana de la interfaz con la solución dada por el programa

8. Conclusión

La implementación de una heurística bioinspirada en la biología nos permite verificar y analizar que para la búsqueda de una solución con múltiples alternativas puede ser más sencillo llegar a una convergencia donde el problema pueda tener una cercanía más a una solución de tipo determinista bajo una metodología heurística.