Universidad de Carabobo

Facultad Experimental de Ciencias y Tecnología

Departamento de Computación

Electiva: Introducción a la Inteligencia Artificial

Tarea 1: Algoritmos Genéticos simples

Integrantes:

Jhonattan Garcia, C.I: 24.423.188

Luis Sierra, C.I: 25.582.661

Prof. Aníbal Guerra

Bárbula, 07 de septiembre de 2021

**Pasos para ejecutar el programa**

1. Tener instalado el intérprete de Python en su S.O.
2. Crear un entorno virtual con el siguiente comando:

* Windows: py -m venv venv
* Linux: python3 venv venv

1. Ejecutar el programa

* Windows: py main.py
* Linux: python3 main.py

**Datos utilizados para el programa**

Los datos de prueba se encuentran un el archivo **in.txt** donde cada línea corresponde a un caso de prueba distinto. Cada línea contiene 5 elementos separados por un espacio en blanco que corresponden a: N, Pc, Pm, #iter, Gap.

Entrada:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | PC | PM | #iter | Gap |
| 10 | 0.1 | 0.001 | 100 | 10 |
| 100 | 0.8 | 0.5 | 1000 | 100 |
| 10 | 0.1 | 0.5 | 100 | 100 |
| 10 | 0.8 | 0.001 | 100 | 100 |
| 10 | 0.8 | 0.001 | 100 | 10 |
| 100 | 0.1 | 0.5 | 1000 | 100 |
| 100 | 0.1 | 0.5 | 1000 | 10 |
| 100 | 0.8 | 0.001 | 1000 | 100 |
| 100 | 0.8 | 0.001 | 1000 | 10 |

Salida: para casa de prueba se generará un archivo outN.txt con N = 1, 2…, n correspondiente al reporte de las generaciones.

**Observaciones**

Generales

1. El programa puede tardar en generar el archivo de salida ya que al realizar un proceso de cruce o mutación se pueden crear individuos que no pertenezcan al espacio de soluciones y de ser así, se repetirá dicho proceso hasta que aparezcan los individuos permitidos dentro del rango [0, 1].
2. No todas las veces que se ejecuten los mismos casos de prueba se obtienen los resultados esperados.

Casos de prueba adicionales

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | PC | PM | #iter | Gap |
| 10 | 0.1 | 0.006 | 100 | 60 |

Al realizar el experimento se comprobó que utilizando valores mayores a 60% para el Gap y un número mayor a 100 generaciones, el tiempo de respuesta del programa es mucho más lento debido a la cantidad de individuos nuevos que van apareciendo a través de las nuevas generaciones. Adicionalmente, se verifico con una cantidad menor a 100 generaciones, pero se observó que el resultado no es el esperado.

Las imágenes que se presentan a continuación permiten visualizar la convergencia de la función con los datos introducidos.



1.a. Población inicial de 10 individuos aleatorios en [0, 1].

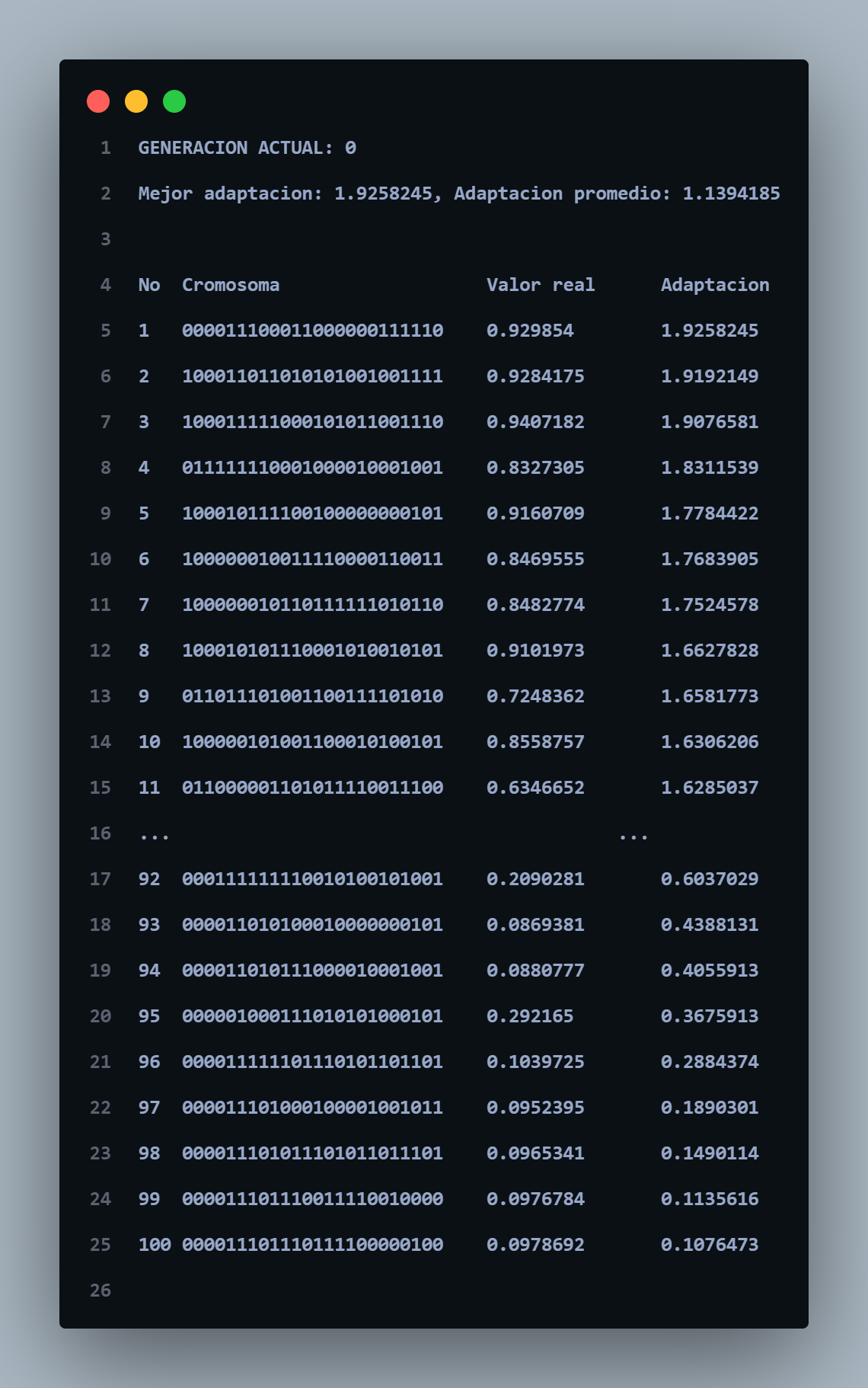


1.b. Población descendiente después de 100 generaciones.

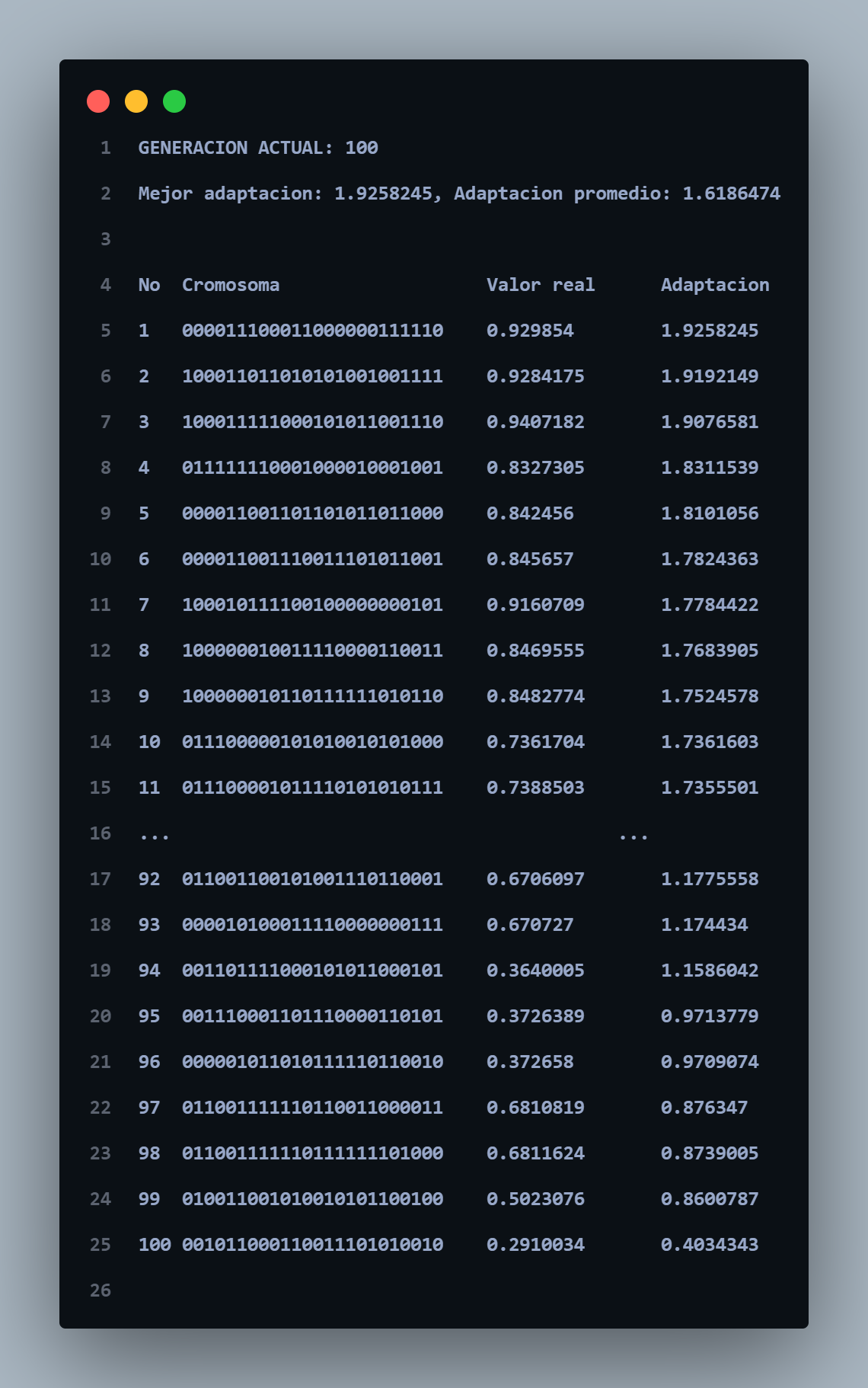
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | PC | PM | #iter | Gap |
| 100 | 0.7 | 0.002 | 100 | 20 |

Con este experimento observamos que a medida que se incrementa la cantidad de individuos, si la cantidad de generaciones es mayor a 100 y el Gap es mayor al 20%, el tiempo de respuesta es lento.

En las imágenes que se muestran a continuación se puede apreciar la convergencia de la función con estos parámetros.



2.a. Población inicial de 100 individuos aleatorios en [0, 1].



2.b. Población descendiente después de 100 generaciones.

**Listado del programa**

|  |  |  |
| --- | --- | --- |
| Función | Entrada | Salida |
| *float\_to\_bin* | **x:** número real,  **n\_bits:** cantidad de bits permitidos | Retorna el equivalente en binario de x |
| *créate\_population* | **n:** cantidad de individuos,  **n\_bits:** cantidad de bits permitidos | Genera una población de N individuos únicos. |
| *evaluate* | ***population:*** lista de individuos | Agrega el valor de la adaptación a cada individuo de la población. |
| *add\_adaptations* | ***population:*** lista de individuos | Retorna la suma de todas las adaptaciones de los individuos que forman la población de entrada |
| *show\_progress* | ***population:*** lista de individuos,  cases: número de caso de prueba actual,  ***generations:*** generación actual | Muestra la estadística de la generación actual |
| *selection\_probability* | **population:** lista de individuos | Agrega el valor de la probabilidad de selección a cada individuo de la población |
| *search* | **search\_space:** espacio de búsqueda que corresponde a la lista de la rueda de la ruleta,  **elem:** elemento a comparar con los valores del espacio de búsqueda | Retorna el índice del individuo donde se encuentre elem |
| *roulette\_wheel* | ***population:*** lista de individuos | Retorna dos individuos aleatorios |
| *encode* | **num:** número real,  **n\_bits:** umero de bits permitidos | Dado un número real, obtiene su correspondiente número entero |
| *decoding* | **num:** número entero,  **n\_bits:** umero de bits permitidos | Dado un número entero, obtiene su correspondiente número real |
| *scientific\_notation* | **num:** número real | Convierte el número real de formato x.xxe-0x a 0.0000xxx |
| *modification\_point* | **prob:** probabilidad de cruce o mutación,  **n\_bits:** cantidad de bits permitidos | Retorna el punto de modificación de bits para el cruce y la mutación |
| *crossing* | **parent1 y parent2:** números reales que representan a los padres,  **pc:** probabilidad de cruce,  **n\_bits:** cantidad de bits permitidos | Retorna 2 hijos de dos individuos |
| *mutation* | **p1 y p2**: números reales que representan a los individuos,  **pc:** probabilidad de cruce,  **n\_bits**: cantidad de bits permitidos | Retorna la mutación de p1 y p2 |
| *main* |  | Cuerpo principal del AGS |