Inteligencia Artificial

Guía 5.1. Algoritmos Genéticos (AG)

4° Licenciatura En Sistemas de Información

2020

Universidad: UADER FCyT Concepción del Uruguay

Profesor: Lopez De Luise Daniela, Bel Walter

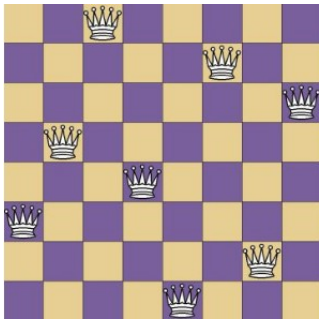
Alumnos: Exequiel Gonzalez, Cepeda Leandro

En los AG, se define como población al conjunto de soluciones posibles, individuo a cada una de las soluciones del espacio pareto, función de fitness a los mecanismos de evaluación sistemática de los individuos para los fines evolutivos. En todos los casos de AG el objeto final es hallar una optimización y es imprescindible aprender a codificar en los genomas al problema, de lo contrario la solución nunca será apropiada para el problema en cuestión.

1. Determinación del genoma:
Determine el vector que represente al problema de 8 reinas.
NOTA: considere que sólo usaremos como lenguaje de representación vectores numéricos.

Definiremos como el vector que representa al problema de 8 reinas, un vector "g" de tamaño i, i=[0..7]; en el que la posición del vector representa la columna, y el valor del vector en dicha posición, la fila donde se ubica la reina en el tablero.

Por ejemplo, esta es una solución del problema 8-reinas representada como vector. G = [5,3,0,4,7,1,6,2]



2. Determinación de la población:
- Cantidad de individuos de la población: TAMANIO_POBLACION = 10
- Porcentaje de mutación: PORCENTAJE_MUTACION = 0.2
- Porcentaje de cruza: PORCENTAJE_CRUZA = 0.6
- Cantidad de generaciones que permitirá al algoritmo evolucionar: GENERACIONES = 100

3. Determine la función de fitness (FF):
- Recuerde que debe medir la distancia numérica entre cada individuo y la solución buscada.
- Recuerde punificar las soluciones que violan las reglas del juego de 8 reinas, que son:
    * Una reina no debe ser comida por otra previamente colocada.
    * Una reina sólo se puede colocar en una casilla válida para comer a otra pieza.
- Recuerde la condición de terminación es que se coloquen las 8 reinas: exista o no ubicaciones óptimas para todas las reinas deben siempre colocarse todas.

La función de fitness que definimos consiste en ir recorriendo cada posición del vector, es decir cada reina, y acumulando la cantidad de amenazas que cada una de estas tiene (tanto en las filas, las columnas y las diagonales). Un vector solución del problema 8-reinas es aquel que tiene una función de fitness igual a 0, es decir que ninguna reina se amenaza entre sí. De esta manera podemos definir que mientras más cerca esté la función de fitness de 0, mejor será el vector.

4. Determine la función de selección:

- Para simplificar el algoritmo escriba sólo la selección de individuos en base a su fitness. Para ello deberá especificar un seudocódigo que permita seleccionar por mejor FF.

```
Funcion seleccion(poblacion: vector[1..n], pressure: int): vector[1..n]
|     """
|     Devuelve una lista de individuos seleccionados por mejor fitness, limitada por pressure.
|     """
Var
|     evaluados: vector[1..n]
|     poblacion_evaluada: vector[1..n]
|     seleccionados: vector[1..n]
Inicio
|
|     # crea un array de tuplas con el fitnes del individuo y el individuo
|     Para i: 1 .. tamanio(poblacion), i++ hacer:
|     |     evaluados[i] = (calcular_fitnes(poblacion[i]), poblacion[i])
|
|     # ordena el vector de individuos evaluados por fitnes de menor a mayor.
|     poblacion_evaluada = Ordenar(evaluados)
|
|     # crea un vector de individuos seleccionados, la cantidad de individuos seleccionados sera igual al pressure
|     for i: 1 .. pressure, i++ hacer:
|     |     seleccionados = evaluados[i][1]
|
|     return seleccionados
Fin
```

5. Determine la función de cruza:

- Para simplificar el algoritmo se pondrá solo un punto de corte. Además la cruza es con reemplazo, es decir, un algoritmo donde los padres ya no son seleccionables si han sido usados para cruzarse durante esa generación.

```
Funcion cruza(poblacion, seleccionados, porcentaje_cruza, pressure=3): vector [1..n]
|     """
|     Devuelve una nueva poblacion en la que los individuos menos aptos son cruzados
|     a partir de individuos padres mas aptos. Esta nueva poblacion se crea a partir
|     de la recibida como parametro, ademas es posible definirse la cantidad de elementos
|     aptos a seleccionar para posteriormente realizar la cruza, a partir del parametro "pressure"
|     """
Var
|     nueva_poblacion, padre1, padre2: vector[1..n]
|     numero_random: int
Inicio
|
|     #Creamos una nueva poblacion con los mismos valores de la original
|     para i: 1..tamanio(poblacion) hacer:
|     |     nueva_poblacion[i] = poblacion[i]
|
|     # Cruza los individuos que fueron seleccionados como padres
|     para i: 0..presure hacer:
|     |
|     |     #Se elige un punto para hacer el intercambio
|     |     punto = random.randint(1, len(seleccionados[0])-1)
|     |
|     |     #Se eligen dos padres
|     |     numero_random = random.randint(0, pressure)
|     |     padre1 = seleccionados[numero_random]
|     |
|     |     numero_random = random.randint(0, pressure)
|     |     padre2 = seleccionados[numero_random]
|     |
|     |     #Se mezcla el material genetico de los padres en cada nuevo individuo
|     |     nueva_poblacion[i][:punto] = padre1[:punto]
|     |     nueva_poblacion[i][punto:] = padre2[punto:]
|
|
|     # Cruza los individuos que no fueron seleccionados como padres
|     para i: pressure.. tamanio(nueva_poblacion) hacer:
|     |
|     |     si random.random() ⩽ porcentaje_cruza hacer:
|     |     |
|     |     |     #Se elige un punto para hacer el intercambio
|     |     |     punto = random.randint(1, len(seleccionados[0])-1)
|     |     |
|     |     |     #Se eligen dos padres
|     |     |     numero_random = random.randint(0, pressure)
|     |     |        padre1 = seleccionados[numero_random]
|     |     |
|     |     |     numero_random = random.randint(0, pressure)
|     |     |     padre2 = seleccionados[numero_random]
|     |     |
|     |     |     #Se mezcla el material genetico de los padres en cada nuevo individuo
|     |     |     nueva_poblacion[i][:punto] = padre1[:punto]
|     |     |     nueva_poblacion[i][punto:] = padre2[punto:]
|
```

```
|     return nueva_poblacion
FIN
```

6.Determine la función de mutación:

Para simplificar el algoritmo, consiste sólo en el cambio a otra casilla disponible, siempre que se supere un umbral de XX %. Es decir, una vez determinado que un par de individuos se cruzarán, se obtiene un número al azar entre 0 y 99, y sólo se efectúa la cruza si dicho número supera el umbral de XX

```
Funcion mutacion(poblacion, porcentaje_mutacion, pressure=3): vector [1..n]
|     """
|     Devuelve una nueva poblacion con los individuos de la poblacion recibida
|     como parametro ya mutados. Se debe tener en cuenta que la mutacion es posterior
|     a la cruza. El criterio de mutacion para el caso es el intercambio de dos genes
|     seleccionados al azar.
|   """
Var
|     nueva_poblacion: vector[1..n]
|     punto1, valor: int
|
Inicio
|
|     para i: 1..tamanio(poblacion) hacer:
|     |     nueva_poblacion[i] = poblacion[i]
|
|     para i: 1..tamanio(nueva_poblacion) hacer:
|     |
|     |     #Cada individuo de la poblacion (menos los padres) tienen una probabilidad de mutar
|     |     si random.random() ⩽ porcentaje_mutacion:
|     |     |
|     |     |     #Se elige un punto al azar
|     |     |     punto1 = random.randint(0,tamanio(nueva_poblacion[0])-1)
|     |     |
|     |     |     #Se elige un valor al azar
|     |     |     valor = random.randint(0,tamanio(nueva_poblacion[0])-1)
|     |     |
|     |     |     mientras nueva_poblacion[i][punto1] == valor:
|     |     |     |     valor = random.randint(0,tamanio(nueva_poblacion[0])-1)
|     |     |
|     |     |     #Se aplica la mutacion
|     |     |     nueva_poblacion[i][punto1] = valor
|
|     return nueva_poblacion
Fin
```

7. Determine el proceso de inicialización:

a) Inicie la población al azar y mida los resultados de la FF al finalizar 5 corridas.

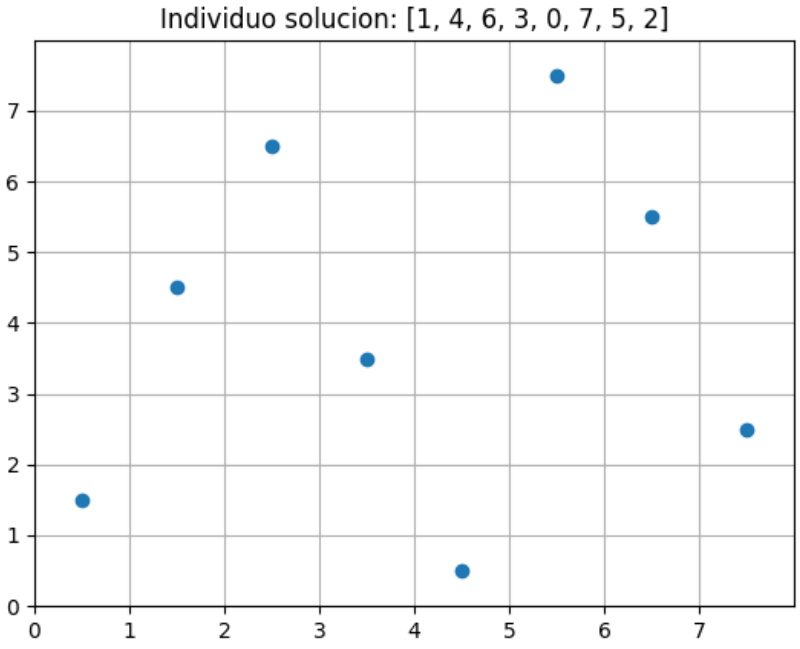
```
Guia 5.1. Algoritmos Genéticos (AG) - Problema de las n-Reinas

TAMANIO_INDIVIDUO: 8
TAMANIO_POBLACION: 10
GENERACIONES: 100
PORCENTAJE_MUTACION: 0.2
PRESSURE: 3

EJECUCION N°: 1

POBLACION INICIAL: [[7, 2, 1, 0, 6, 3, 5, 4], [4, 0, 2, 3, 1, 5, 7, 6], [1, 5, 6, 3, 2, 7, 0, 4], [7, 5, 1, 2, 4, 0, 6, 3], [7, 1, 0, 2, 5, 4, 3, 6], [6, 4, 3, 0, 1, 7, 5, 2], [0, 6, 4, 1, 7, 5, 3, 2], [3, 6, 7, 0, 2, 1, 4, 5], [0, 1, 5, 6, 7, 3, 2, 4], [4, 2, 7, 0, 5, 6, 1, 3]]

POBLACION FINAL (individuo, fitness): [([1, 4, 6, 3, 0, 7, 5, 2], 0), ([1, 4, 6, 3, 0, 7, 5, 2], 0), ([1, 4, 6, 3, 0, 7, 5, 2], 0), ([1, 4, 6, 3, 0, 7, 5, 2], 0), ([1, 4, 6, 3, 0, 7, 5, 2], 0), ([1, 4, 6, 3, 0, 7, 5, 2], 0), ([1, 4, 6, 3, 0, 7, 5, 2], 0), ([1, 4, 6, 3, 0, 7, 5, 1], 4), ([1, 4, 6, 3, 0, 7, 5, 2], 0)]

MEJOR INDIVIDUO ENCONTRADO: [1, 4, 6, 3, 0, 7, 5, 2], SU FITNESS ES: 0
```

```
EJECUCION N°: 2

POBLACION INICIAL: [[6, 0, 4, 2, 3, 7, 5, 1], [7, 5, 0, 3, 4, 6, 1, 2], [0, 7, 6, 2, 3, 5, 4, 1], [0, 2, 5, 4, 1, 3, 7, 6], [4, 5, 6, 2, 3, 1, 7, 0], [4, 3, 2, 7, 0, 5, 1, 6], [3, 6, 7, 1, 4, 2, 0, 5], [4, 0, 6, 3, 1, 5, 2, 7], [6, 2, 3, 1, 7, 0, 4, 5], [4, 3, 0, 6, 5, 2, 7, 1]]

POBLACION FINAL (individuo, fitness): [([2, 2, 0, 1, 4, 2, 0, 3], 9), ([2, 2, 7, 1, 4, 2, 0, 3], 3), ([2, 2, 7, 1, 4, 2, 0, 3], 3), ([2, 2, 7, 4, 4, 2, 0, 3], 9), ([2, 2, 7, 1, 4, 2, 0, 0], 7), ([2, 2, 7, 1, 4, 2, 0, 3], 3), ([7, 2, 7, 1, 4, 2, 0, 3], 6), ([2, 2, 7, 1, 4, 2, 0, 3], 3), ([2, 2, 7, 1, 4, 2, 0, 3], 3), ([2, 2, 7, 1, 4, 2, 0, 3], 3)]

MEJOR INDIVIDUO ENCONTRADO: [2, 2, 7, 1, 4, 2, 0, 3], SU FITNESS ES: 3
```

```
EJECUCION N°: 3

POBLACION INICIAL: [[5, 3, 2, 7, 0, 6, 4, 1], [1, 2, 0, 4, 6, 3, 7, 5], [0, 5, 4, 7, 2, 6, 1, 3], [5, 6, 4, 1, 2, 0, 7, 3], [7, 4, 0, 6, 5, 1, 3, 2], [6, 3, 1, 5, 4, 2, 0, 7], [3, 6, 4, 5, 7, 2, 1, 0], [7, 3, 6, 0, 2, 5, 1, 4], [0, 2, 5, 7, 3, 1, 6, 4], [6, 4, 1, 3, 5, 0, 7, 2]]

POBLACION FINAL (individuo, fitness): [([0, 2, 7, 5, 3, 1, 6, 4], 2), ([0, 2, 7, 4, 3, 1, 6, 4], 8), ([0, 2, 7, 5, 3, 1, 6, 4], 2), ([0, 2, 7, 5, 3, 1, 6, 4], 2), ([0, 2, 7, 5, 3, 1, 7, 4], 4), ([0, 2, 7, 5, 3, 1, 6, 4], 2), ([0, 2, 7, 5, 3, 1, 6, 4], 2), ([0, 2, 7, 5, 3, 1, 6, 4], 2), ([0, 2, 7, 2, 3, 1, 6, 4], 6), ([0, 2, 7, 5, 3, 1, 6, 4], 2)]

MEJOR INDIVIDUO ENCONTRADO: [0, 2, 7, 5, 3, 1, 6, 4], SU FITNESS ES: 2
```

```
EJECUCION N°: 4

POBLACION INICIAL: [[5, 7, 6, 2, 4, 1, 0, 3], [4, 3, 6, 2, 0, 5, 1, 7], [1, 4, 2, 7, 6, 0, 3, 5], [5, 2, 4, 6, 7, 0, 1, 3], [2, 5, 3, 0, 6, 7, 1, 4], [0, 6, 7, 1, 3, 2, 4, 5], [1, 3, 0, 5, 6, 7, 4, 2], [0, 4, 3, 6, 2, 7, 5, 1], [5, 2, 6, 0, 3, 1, 7, 4], [2, 6, 5, 1, 7, 4, 0, 3]]

POBLACION FINAL (individuo, fitness): [([7, 2, 2, 6, 2, 0, 5, 1], 3), ([5, 2, 2, 6, 2, 0, 5, 1], 7), ([7, 2, 2, 6, 2, 0, 5, 1], 3), ([7, 2, 2, 6, 2, 0, 5, 1], 3), ([7, 2, 2, 6, 2, 0, 5, 1], 3), ([7, 2, 2, 6, 2, 0, 5, 1], 3), ([7, 2, 2, 6, 2, 0, 5, 1], 3), ([7, 2, 2, 6, 2, 0, 5, 1], 3), ([7, 2, 2, 6, 2, 0, 5, 1], 3)]

MEJOR INDIVIDUO ENCONTRADO: [7, 2, 2, 6, 2, 0, 5, 1], SU FITNESS ES: 3
```

```
EJECUCION N°: 5

POBLACION INICIAL: [[5, 1, 4, 3, 0, 7, 6, 2], [5, 3, 7, 6, 2, 4, 0, 1], [3, 2, 0, 5, 4, 6, 7, 1], [1, 7, 0, 5, 2, 6, 3, 4], [6, 3, 7, 1, 0, 4, 5, 2], [4, 1, 2, 5, 6, 0, 7, 3], [2, 3, 7, 6, 0, 5, 1, 4], [0, 2, 6, 5, 1, 4, 7, 3], [5, 7, 0, 3, 4, 6, 1, 2], [5, 4, 2, 1, 6, 0, 7, 3]]

POBLACION FINAL (individuo, fitness): [([7, 2, 0, 5, 1, 4, 0, 3], 2), ([7, 2, 0, 5, 1, 4, 0, 3], 2), ([7, 2, 0, 5, 1, 4, 0, 3], 2), ([7, 2, 7, 5, 1, 4, 0, 3], 4), ([7, 2, 0, 5, 1, 4, 0, 3], 2), ([7, 2, 0, 5, 1, 4, 0, 3], 2), ([7, 1, 0, 5, 1, 4, 0, 3], 6), ([7, 2, 0, 5, 1, 4, 0, 3], 2), ([7, 2, 0, 5, 1, 4, 0, 3], 2), ([7, 2, 0, 2, 1, 4, 0, 2], 11)]

MEJOR INDIVIDUO ENCONTRADO: [7, 2, 0, 5, 1, 4, 0, 3], SU FITNESS ES: 2


Se encontraron 1 soluciones: [[1, 4, 6, 3, 0, 7, 5, 2]]
```



b) Inicie la población en una solución trivial y mida los resultados de la FF al finalizar 5 corridas.

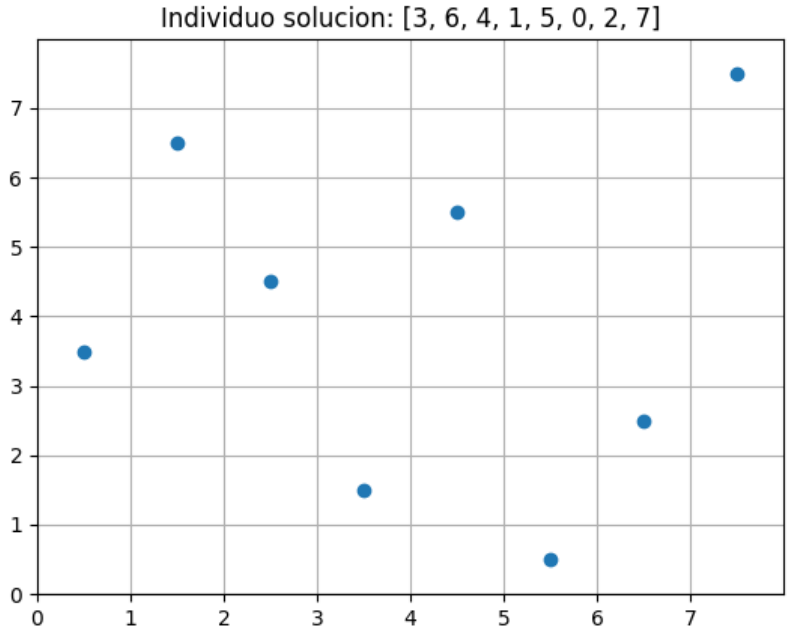
```
Guía 5.1. Algoritmos Genéticos (AG) - Problema de las n-Reinas

TAMANIO_INDIVIDUO: 8
TAMANIO_POBLACION: 10
GENERACIONES: 100
PORCENTAJE_MUTACION: 0.2
PRESSURE: 3

EJECUCION N°: 1

POBLACION INICIAL: [[7, 3, 0, 2, 5, 1, 6, 4], [7, 2, 0, 5, 1, 4, 6, 3], [7, 1, 4, 2, 0, 6, 3, 5], [7, 1, 3, 0, 6, 4, 2, 5], [6, 4, 2, 0, 5, 7, 1, 3], [6, 3, 1, 7, 5, 0, 2, 4], [6, 3, 1, 4, 7, 0, 2, 5], [6, 2, 7, 1, 4, 0, 5, 3], [6, 2, 0, 5, 7, 4, 1, 3], [6, 1, 5, 2, 0, 3, 7, 4]]

POBLACION FINAL (individuo, fitness): [([6, 4, 2, 0, 5, 7, 1, 1], 2), ([6, 4, 2, 0, 5, 5, 1, 1], 6), ([6, 4, 2, 0, 5, 7, 1, 1], 2), ([6, 4, 2, 0, 5, 7, 1, 1], 2), ([6, 4, 2, 0, 5, 7, 1, 1], 2), ([6, 4, 2, 0, 5, 7, 1, 1], 2), ([6, 4, 2, 0, 5, 7, 1, 1], 2), ([0, 4, 2, 0, 5, 7, 1, 1], 6), ([6, 4, 2, 0, 5, 7, 0, 1], 6), ([6, 4, 2, 0, 5, 7, 1, 1], 2)]

MEJOR INDIVIDUO ENCONTRADO: [6, 4, 2, 0, 5, 7, 1, 1], SU FITNESS ES: 2
```

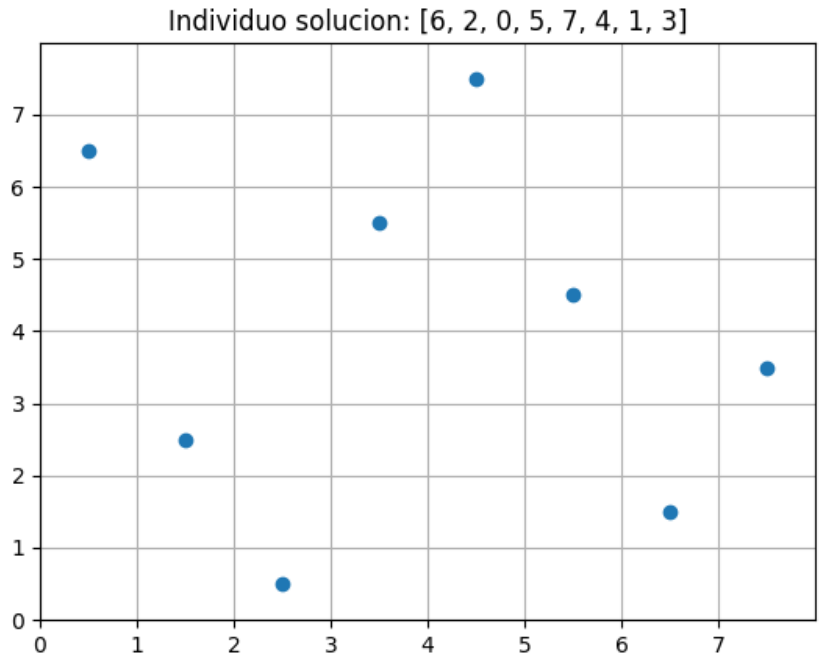
```
EJECUCION N°: 2

POBLACION INICIAL: [[7, 3, 0, 2, 5, 1, 6, 4], [7, 2, 0, 5, 1, 4, 6, 3], [7, 1, 4, 2, 0, 6, 3, 5], [7, 1, 3, 0, 6, 4, 2, 5], [6, 4, 2, 0, 5, 7, 1, 3], [6, 3, 1, 7, 5, 0, 2, 4], [6, 3, 1, 4, 7, 0, 2, 5], [6, 2, 7, 1, 4, 0, 5, 3], [6, 2, 0, 5, 7, 4, 1, 3], [6, 1, 5, 2, 0, 3, 7, 4]]

POBLACION FINAL (individuo, fitness): [([3, 6, 4, 1, 5, 0, 2, 7], 0), ([3, 6, 4, 1, 5, 0, 2, 7], 0), ([3, 6, 4, 1, 5, 0, 2, 7], 0), ([3, 6, 4, 1, 5, 0, 2, 7], 0), ([3, 6, 4, 1, 5, 0, 2, 7], 0), ([3, 5, 4, 1, 5, 0, 2, 7], 4), ([3, 6, 4, 1, 5, 0, 2, 7], 0), ([3, 6, 0, 7, 5, 0, 2, 7], 4), ([3, 7, 4, 1, 5, 0, 2, 7], 4), ([3, 6, 4, 1, 5, 0, 2, 7], 0)]

MEJOR INDIVIDUO ENCONTRADO: [3, 6, 4, 1, 5, 0, 2, 7], SU FITNESS ES: 0
```

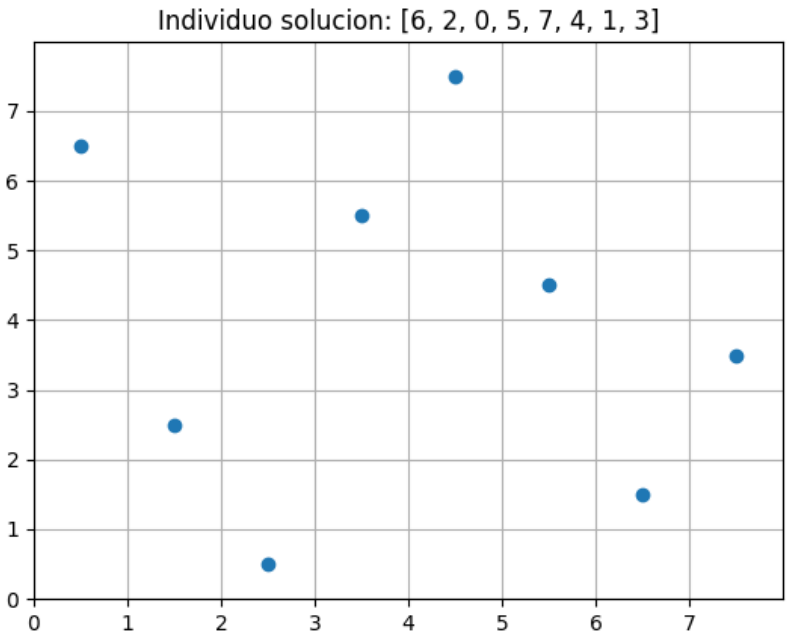
```
EJECUCION N°: 3

POBLACION INICIAL: [[7, 3, 0, 2, 5, 1, 6, 4], [7, 2, 0, 5, 1, 4, 6, 3], [7, 1, 4, 2, 0, 6, 3, 5], [7, 1, 3, 0, 6, 4, 2, 5], [6, 4, 2, 0, 5, 7, 1, 3], [6, 3, 1, 7, 5, 0, 2, 4], [6, 3, 1, 4, 7, 0, 2, 5], [6, 2, 7, 1, 4, 0, 5, 3], [6, 2, 0, 5, 7, 4, 1, 3], [6, 1, 5, 2, 0, 3, 7, 4]]

POBLACION FINAL (individuo, fitness): [([6, 2, 0, 5, 7, 4, 1, 3], 0), ([6, 2, 0, 5, 7, 4, 1, 3], 0), ([6, 2, 0, 5, 7, 4, 1, 3], 0), ([6, 2, 0, 5, 7, 4, 1, 0], 4), ([6, 2, 0, 5, 7, 4, 1, 3], 0), ([6, 2, 0, 5, 7, 4, 1, 3], 0), ([6, 2, 0, 5, 7, 4, 0, 3], 4), ([6, 2, 0, 5, 7, 4, 1, 3], 0), ([6, 2, 0, 5, 7, 4, 1, 2], 6), ([6, 2, 0, 5, 7, 4, 1, 3], 0)]

MEJOR INDIVIDUO ENCONTRADO: [6, 2, 0, 5, 7, 4, 1, 3], SU FITNESS ES: 0
```

```
EJECUCION N°: 4

POBLACION INICIAL: [[7, 3, 0, 2, 5, 1, 6, 4], [7, 2, 0, 5, 1, 4, 6, 3], [7, 1, 4, 2, 0, 6, 3, 5], [7, 1, 3, 0, 6, 4, 2, 5], [6, 4, 2, 0, 5, 7, 1, 3], [6, 3, 1, 7, 5, 0, 2, 4], [6, 3, 1, 4, 7, 0, 2, 5], [6, 2, 7, 1, 4, 0, 5, 3], [6, 2, 0, 5, 7, 4, 1, 3], [6, 1, 5, 2, 0, 3, 7, 4]]

POBLACION FINAL (individuo, fitness): [([4, 2, 0, 5, 3, 1, 6, 4], 2), ([4, 2, 0, 5, 3, 1, 6, 4], 2), ([4, 2, 0, 5, 3, 1, 6, 4], 2), ([4, 2, 0, 5, 3, 1, 6, 4], 2), ([4, 2, 0, 5, 3, 1, 6, 4], 2), ([4, 2, 0, 5, 3, 1, 6, 4], 2), ([4, 2, 0, 2, 3, 1, 6, 4], 6), ([4, 2, 0, 5, 3, 1, 6, 4], 2), ([4, 2, 1, 5, 3, 1, 6, 4], 8), ([4, 2, 0, 5, 0, 1, 6, 4], 8)]

MEJOR INDIVIDUO ENCONTRADO: [4, 2, 0, 5, 3, 1, 6, 4], SU FITNESS ES: 2
```

```
EJECUCION N°: 5

POBLACION INICIAL: [[7, 3, 0, 2, 5, 1, 6, 4], [7, 2, 0, 5, 1, 4, 6, 3], [7, 1, 4, 2, 0, 6, 3, 5], [7, 1, 3, 0, 6, 4, 2, 5], [6, 4, 2, 0, 5, 7, 1, 3], [6, 3, 1, 7, 5, 0, 2, 4], [6, 3, 1, 4, 7, 0, 2, 5], [6, 2, 7, 1, 4, 0, 5, 3], [6, 2, 0, 5, 7, 4, 1, 3], [6, 1, 5, 2, 0, 3, 7, 4]]

POBLACION FINAL (individuo, fitness): [([6, 2, 0, 5, 7, 3, 1, 3], 6), ([6, 2, 0, 5, 7, 4, 1, 3], 0), ([6, 2, 0, 5, 7, 4, 1, 3], 0), ([6, 2, 0, 5, 7, 4, 0, 3], 4), ([6, 2, 0, 5, 7, 4, 1, 3], 0), ([6, 2, 0, 5, 7, 4, 1, 3], 0), ([6, 2, 0, 0, 7, 4, 1, 3], 4), ([6, 2, 0, 5, 7, 4, 0, 3], 4), ([6, 2, 0, 5, 6, 4, 1, 3], 6), ([6, 2, 0, 5, 7, 4, 1, 3], 0)]

MEJOR INDIVIDUO ENCONTRADO: [6, 2, 0, 5, 7, 4, 1, 3], SU FITNESS ES: 0


Se encontraron 3 soluciones: [[3, 6, 4, 1, 5, 0, 2, 7], [6, 2, 0, 5, 7, 4, 1, 3], [6, 2, 0, 5, 7, 4, 1, 3]]
```







c) Genere un informe con todos los resultados. compare y justifique.

Parametros	POBLACION AL AZAR	POBLACION SOLUCION TRIVIAL
TAMANIO_INDIVIDUO	8	8
TAMANIO_POBLACION	10	10
GENERACIONES	100	100
PRESSURE	3	3
PORCENTAJE_CRUZA	0.2	0.2
PORCENTAJE_MUTACION	0.6	0.6
CORRIDAS	5	5
SOLUCIONES	[[1, 4, 6, 3, 0, 7, 5, 2]]	[[3, 6, 4, 1, 5, 0, 2, 7], [6, 2, 0, 5, 7, 4, 1, 3], [6, 2, 0, 5, 7, 4, 1, 3]]

Individuo solución: 🟩

Individuo no solución: 🟥

	POBLACION AL AZAR			POBLACION SOLUCION TRIVIAL		
CORRIDA	POBLACION_INICIAL	MEJOR INDIVIDUO	FITNESS	POBLACION_INICIAL	MEJOR INDIVIDUO	FITNESS
1	[[7, 2, 1, 0, 6, 3, 5, 4], [4, 0, 2, 3, 1, 5, 7, 6], [1, 5, 6, 3, 2, 7, 0, 4], [7, 5, 1, 2, 4, 0, 6, 3], [7, 1, 0, 2, 5, 4, 3, 6], [6, 4, 3, 0, 1, 7, 5, 2], [0, 6, 4, 1, 7, 5, 3, 2], [3, 6, 7, 0, 2, 1, 4, 5], [0, 1, 5, 6, 7, 3, 2, 4], [4, 2, 7, 0, 5, 6, 1, 3]]	[1, 4, 6, 3, 0, 7, 5, 2]	0	[[7, 3, 0, 2, 5, 1, 6, 4], [7, 2, 0, 5, 1, 4, 6, 3], [7, 1, 4, 2, 0, 6, 3, 5], [7, 1, 3, 0, 6, 4, 2, 5], [6, 4, 2, 0, 5, 7, 1, 3], [6, 3, 1, 7, 5, 0, 2, 4], [6, 3, 1, 4, 7, 0, 2, 5], [6, 2, 7, 1, 4, 0, 5, 3], [6, 2, 0, 5, 7, 4, 1, 3], [6, 1, 5, 2, 0, 3, 7, 4]]	[6, 4, 2, 0, 5, 7, 1, 1]	2
2	[[6, 0, 4, 2, 3, 7, 5, 1], [7, 5, 0, 3, 4, 6, 1, 2], [0, 7, 6, 2, 3, 5, 4, 1], [0, 2, 5, 4, 1, 3, 7, 6], [4, 5, 6, 2, 3, 1, 7, 0], [4, 3, 2, 7, 0, 5, 1, 6], [3, 6, 7, 1, 4, 2, 0, 5], [4, 0, 6, 3, 1, 5, 2, 7], [6, 2, 3, 1, 7, 0, 4, 5], [4, 3, 0, 6, 5, 2, 7, 1]]	[2, 2, 7, 1, 4, 2, 0, 3]	3	[[7, 3, 0, 2, 5, 1, 6, 4], [7, 2, 0, 5, 1, 4, 6, 3], [7, 1, 4, 2, 0, 6, 3, 5], [7, 1, 3, 0, 6, 4, 2, 5], [6, 4, 2, 0, 5, 7, 1, 3], [6, 3, 1, 7, 5, 0, 2, 4], [6, 3, 1, 4, 7, 0, 2, 5], [6, 2, 7, 1, 4, 0, 5, 3], [6, 2, 0, 5, 7, 4, 1, 3], [6, 1, 5, 2, 0, 3, 7, 4]]	[3, 6, 4, 1, 5, 0, 2, 7]	0
3	[[5, 3, 2, 7, 0, 6, 4, 1], [1, 2, 0, 4, 6, 3, 7, 5], [0, 5, 4, 7, 2, 6, 1, 3], [5, 6, 4, 1, 2, 0, 7, 3], [7, 4, 0, 6, 5, 1, 3, 2], [6, 3, 1, 5, 4, 2, 0, 7], [3, 6, 4, 5, 7, 2, 1, 0], [7, 3, 6, 0, 2, 5, 1, 4], [0, 2, 5, 7, 3, 1, 6, 4], [6, 4, 1, 3, 5, 0, 7, 2]]	[0, 2, 7, 5, 3, 1, 6, 4]	2	[[7, 3, 0, 2, 5, 1, 6, 4], [7, 2, 0, 5, 1, 4, 6, 3], [7, 1, 4, 2, 0, 6, 3, 5], [7, 1, 3, 0, 6, 4, 2, 5], [6, 4, 2, 0, 5, 7, 1, 3], [6, 3, 1, 7, 5, 0, 2, 4], [6, 3, 1, 4, 7, 0, 2, 5], [6, 2, 7, 1, 4, 0, 5, 3], [6, 2, 0, 5, 7, 4, 1, 3], [6, 1, 5, 2, 0, 3, 7, 4]]	[6, 2, 0, 5, 7, 4, 1, 3]	0
4	[[5, 7, 6, 2, 4, 1, 0, 3], [4, 3, 6, 2, 0, 5, 1, 7], [1, 4, 2, 7, 6, 0, 3, 5], [5, 2, 4, 6, 7, 0, 1, 3], [2, 5, 3, 0, 6, 7, 1, 4], [0, 6, 7, 1, 3, 2, 4, 5], [1, 3, 0, 5, 6, 7, 4, 2], [0, 4, 3, 6, 2, 7, 5, 1], [5, 2, 6, 0, 3, 1, 7, 4], [2, 6, 5, 1, 7, 4, 0, 3]]	[7, 2, 2, 6, 2, 0, 5, 1]	3	[[7, 3, 0, 2, 5, 1, 6, 4], [7, 2, 0, 5, 1, 4, 6, 3], [7, 1, 4, 2, 0, 6, 3, 5], [7, 1, 3, 0, 6, 4, 2, 5], [6, 4, 2, 0, 5, 7, 1, 3], [6, 3, 1, 7, 5, 0, 2, 4], [6, 3, 1, 4, 7, 0, 2, 5], [6, 2, 7, 1, 4, 0, 5, 3], [6, 2, 0, 5, 7, 4, 1, 3], [6, 1, 5, 2, 0, 3, 7, 4]]	[4, 2, 0, 5, 3, 1, 6, 4]	2
5	[[5, 1, 4, 3, 0, 7, 6, 2], [5, 3, 7, 6, 2, 4, 0, 1], [3, 2, 0, 5, 4, 6, 7, 1], [1, 7, 0, 5, 2, 6, 3, 4], [6, 3, 7, 1, 0, 4, 5, 2], [4, 1, 2, 5, 6, 0, 7, 3], [2, 3, 7, 6, 0, 5, 1, 4], [0, 2, 6, 5, 1, 4, 7, 3], [5, 7, 0, 3, 4, 6, 1, 2], [5, 4, 2, 1, 6, 0, 7, 3]]	[7, 2, 0, 5, 1, 4, 0, 3]	2	[[7, 3, 0, 2, 5, 1, 6, 4], [7, 2, 0, 5, 1, 4, 6, 3], [7, 1, 4, 2, 0, 6, 3, 5], [7, 1, 3, 0, 6, 4, 2, 5], [6, 4, 2, 0, 5, 7, 1, 3], [6, 3, 1, 7, 5, 0, 2, 4], [6, 3, 1, 4, 7, 0, 2, 5], [6, 2, 7, 1, 4, 0, 5, 3], [6, 2, 0, 5, 7, 4, 1, 3], [6, 1, 5, 2, 0, 3, 7, 4]]	[6, 2, 0, 5, 7, 4, 1, 3]	0

Se puede observar que al realizar 5 corridas utilizando los mismos parámetros para el algoritmo, variando unicamente los individuos de la población inicial, se obtiene un mejor resultado en cuanto a cantidad de soluciones encontradas en el caso de la inicialización de individuos en una población solución trivial.