



**UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO**

---

---

**UNIDAD ACADÉMICA PROFESIONAL TIANGUISTENCO**

**INGENIERÍA EN SOFTWARE**

**“Implementación de algoritmo genético para resolver  
el tablero de ajedrez roto de Sam Loyd”**

**PROTOCOLO DE TESIS**

**QUE PRESENTA**

**KEVIN BENITEZ VALENTIN**

**DIRECTOR: ING. EN SW. JONATHAN ROJAS SIMON**

**TIANGUISTENCO, MÉX.**

**NOVIEMBRE 2022**

Agradecimientos:

Agradezco a la Dra. Yulia Ledeneva y al Ing. en Sw Jonathan Rojas Simón por el formato proporcionado.

## Contenido

1. Introducción .....	3
Entorno de desarrollo .....	3
¿Cuáles son las condiciones (variables) que intervienen en el sistema? .....	3
2. Planteamiento del problema .....	4
Justificación.....	4
Pregunta a resolver .....	4
3. Codificación del Individuo .....	5
Características del individuo.....	5
Representación del individuo.....	5
Implementación del individuo dentro del tablero .....	6
4. Función de aptitud .....	9
5. Operadores usados .....	10
6. Criterios de parada .....	11
7. Solución del problema .....	11
8. Pruebas de ejecución .....	14
Ejecución 1:.....	14
Ejecución 2.....	15
Ejecución 3.....	16
Ejecución 4.....	18
Ejecución 5:.....	20
9. Pruebas de ejecución con el profesor .....	23
Prueba 1:.....	23
Prueba 2.....	24
Prueba 3.....	25
Prueba 4.....	28
Prueba 5:.....	28
Prueba 6.....	30
10. Conclusiones .....	32
11. Referencias.....	32

## **1. Introducción**

### **Entorno de desarrollo**

El proyecto estará trabajando con las piezas que se tienen dentro del tablero de ajedrez roto.

Este rompecabezas es particularmente interesante, ya que permite resolverlo no trabajando exclusivamente a base de prueba y error.

Para resolverlo "deductivamente" es necesario, en primer lugar, fijarse en la forma de las piezas, y acto seguido ir pensando y utilizar algunos métodos clásicos de abordar problemas como puede ser la clasificación, la ordenación o el apareamiento.

El rompecabezas se compone de piezas tipo tetrís en las que es necesario juntar todas las piezas del rompecabezas de una manera específica para obtener un tablero de ajedrez de 8x8 completamente lleno con todas las fichas correctamente alineadas.

### **¿Cuáles son las condiciones (variables) que intervienen en el sistema?**

Existen diversas variantes que nos apoyaran con la resolución de este problema como lo son:

- El rompecabezas se compone de 13 piezas
  - Para cada pieza se cuentan con aproximadamente 8x8 posiciones del tablero.
  - 4 rotaciones en un plano 2D.
  - 2 orientaciones de volteo (pieza original o pieza reflejada).
- Podemos descartar subconjuntos de casos que no tienen sentido para armar el rompecabezas, teniendo en cuenta los siguientes criterios:
  - Superposiciones de piezas.
  - Posiciones que tomarían las piezas que se encuentran fuera de los límites del tablero.
  - El espacio vacío que queda en el tablero.

- La alineación de las fichas.

## 2. Planteamiento del problema

### Justificación

El problema es una variación del problema de la mochila que es caracterizado por ser un problema computacional estándar donde los artículos de diferentes formas y tamaños deben empaquetarse de manera óptima en un contenedor de un volumen fijo dado. Existen muchas variantes entre las que se encuentra el tiempo, volumen, costo, etc. Enfocándonos en el ajedrez decimos que estos problemas generalmente requieren un tiempo exponencial a medida que aumenta el tamaño del problema, por ejemplo, si este rompecabezas fuera de 3 piezas podemos resolverlo rápidamente, pero si consta de más, entonces la cantidad de variaciones aumenta considerablemente.

### Pregunta a resolver

De qué manera podemos resolver el tablero satisfactoriamente tomando los siguientes criterios:

- Que no sobre ninguna pieza
- No dejar espacios en blanco en el tablero
- Resolverlo en un tiempo aceptable

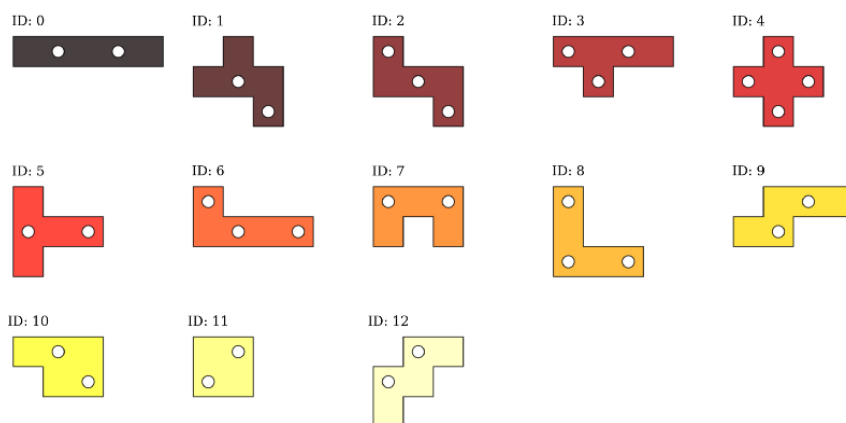


### 3. Codificación del Individuo

#### Características del individuo

Cada configuración de todas las piezas representa un individuo potencial con cierto nivel de aptitud, cuanto mejor sean sus condiciones físicas es decir (cuanto menor sea el número de piezas restantes) más cerca estaremos de encontrar al mejor y por ende la solución.

A continuación, tenemos las piezas armadas en Python, originalmente representan las piezas del rompecabezas, siendo así que los puntos blancos harán la representación de las damas blancas.



El orden en que colocaremos las piezas es de vital importancia, por lo que la secuencia en que serán colocadas las piezas dentro del tablero es un buen candidato para hacer la representación de nuestro individuo, el cromosoma.

#### Representación del individuo

El cromosoma se representará mediante los ID de cada pieza por lo que un individuo generado de manera aleatoria quedaría de la siguiente forma:

#### El individuo se representa por un cromosoma de 13 casillas

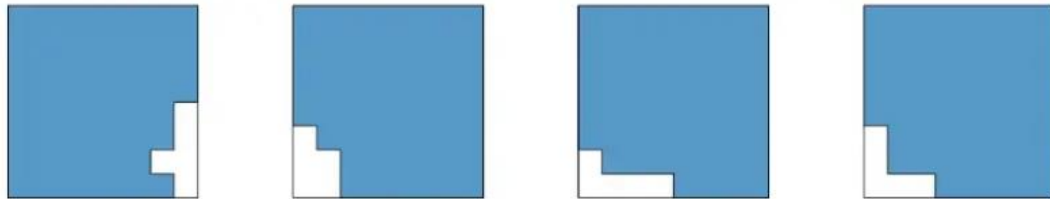
4	3	6	8	5	10	0	9	11	2	1	12	7
---	---	---	---	---	----	---	---	----	---	---	----	---

Lo que significa que el tablero será resuelto siguiendo el orden de las piezas con los ID **4,3,6,8,5,10,0,9,11,2,1,12,7**. Y para determinar de qué manera se deben de acomodar las piezas se irán colocando una a una, por cada vez que se haga la

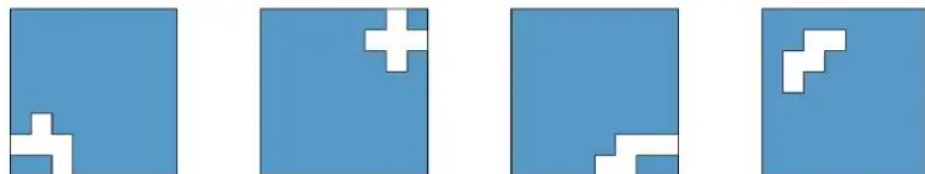
colocación de una pieza nueva, posteriormente la idea es capturar las casillas ocupadas del tablero y a esto le denominaremos perfil.

### Implementación del individuo dentro del tablero

De esta forma podemos colocar la primera pieza en el tablero.



Aquí un ejemplo de cómo no sería óptimo colocar la primera pieza:



Para obtener una ubicación óptima debemos buscar la configuración que resulte en un perfil con la longitud del perímetro más corta del límite, tanto externa como interna, a continuación, algunos ejemplos con las longitudes calculadas, mostrándolos de izquierda a derecha de mejor a peor.

### Ejemplo tomando la ficha con el ID: 5



**P = 36**

**P = 38**

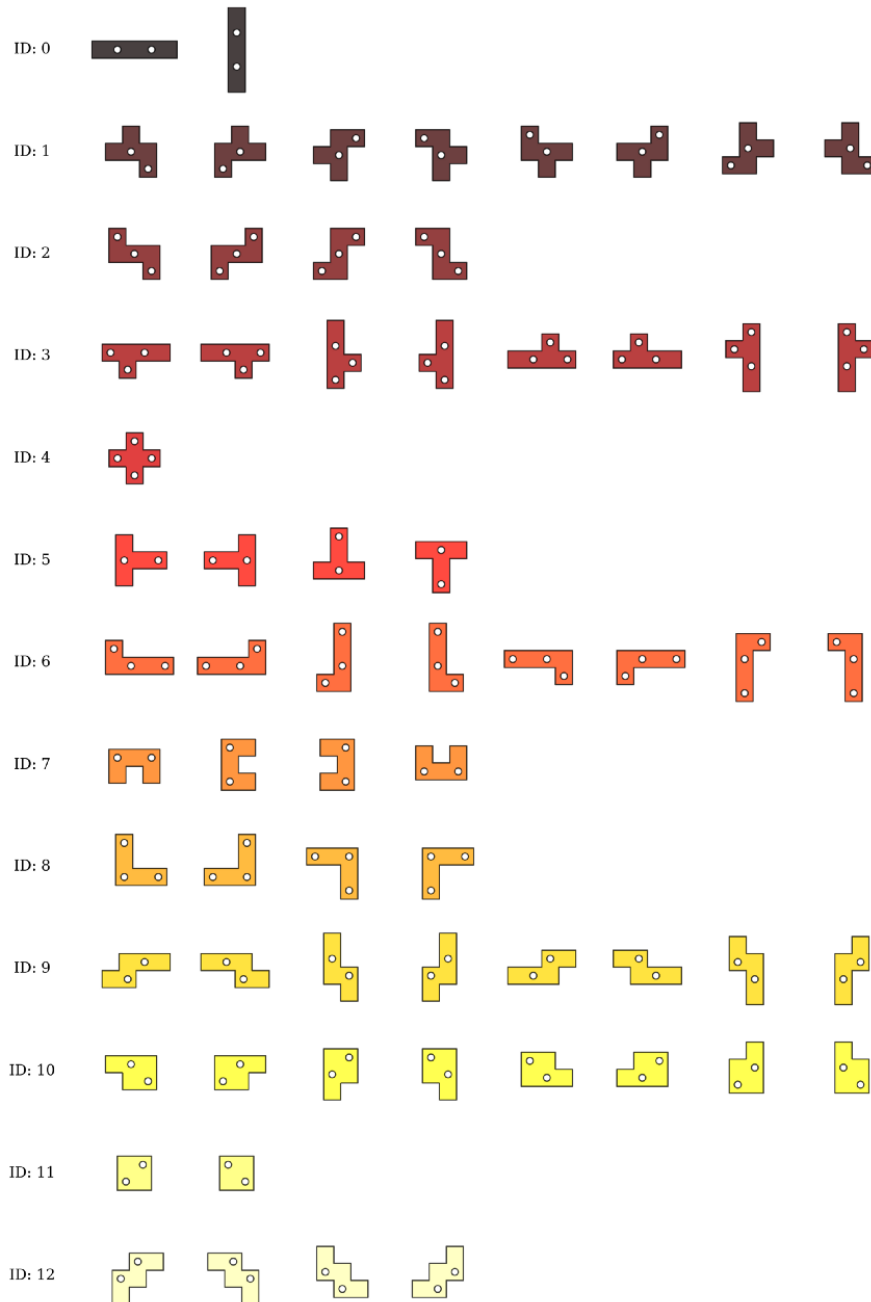
**P = 42**

**P = 44**

Para encontrar la mejor ubicación repasamos las configuraciones para la pieza dada, esto incluye también verificar las posiciones restantes que quedan teniendo en cuenta a su vez como se acomodaran las fichas

restantes.

El número de configuraciones para el tablero también depende de la forma de las piezas ya que algunas tienen más simetría que otras, la orientación que pueden tomar cada una de las piezas se muestra a continuación.



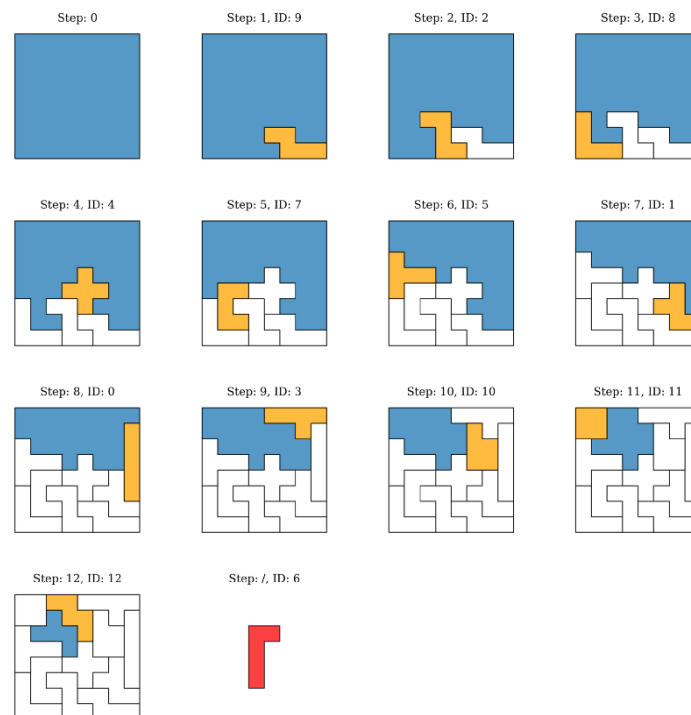
Una vez que todas las orientaciones fueron encontradas, estas se prueban una

a una de acuerdo con las que se consideran como validas, para determinar que posiciones en el tablero se deben cubrir cada uno de los siguientes aspectos.

- La pieza debe estar contenida dentro del perfil del tablero
- La pieza no deberá dividir el perfil del tablero en 2 partes desconocidas (las demás piezas no embonen).
- Las fichas de la pieza (damas) deberán estar alineadas con las fichas del perfil (la primera pieza establece las fichas del perfil)
- Una vez que se verifican todas las configuraciones de la pieza única, también puede suceder que existan múltiples ubicaciones optimas, por lo que se selecciona una de ellas al azar.

### Individuo de prueba haciendo los movimientos dentro del tablero

9	2	8	4	7	5	1	0	3	10	11	12	6
---	---	---	---	---	---	---	---	---	----	----	----	---



**Parte azul:** vacío

**Parte naranja:** Pieza en colocación

**Parte blanca:** Piezas colocadas

**Parte roja:** Piezas sobrantes.

Se realizaron 12 pasos faltando un paso para acomodar todas las piezas, sobrando una pieza (ID: 6)



#### 4. Función de aptitud

La función de aptitud de un individuo es calculada cuando el algoritmo de colocación optima es finalizado. De la configuración inicial del tablero extraemos el área que faltó rellenar, esto se hace contando la cantidad de pixeles o cuadros de nuestro tablero  $8 \times 8 = 64$ , posteriormente lo sumamos con el número de piezas que faltaron ensamblar, y posteriormente se suma la línea exterior del área que faltó rellenar.

El mejor resultado tiene que ser igual a 0, donde indicamos que todas las piezas han sido acomodadas de manera correcta.

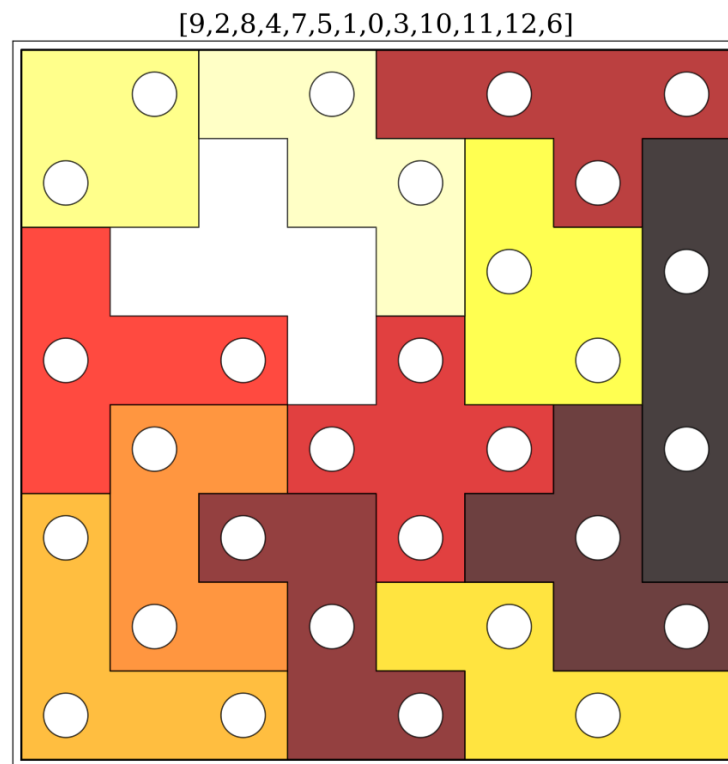
$$FA = Ar + Pf + Le$$

*Ar*, se refiere al area de la zona del tablero que faltó por rellenar

*Pf*, se refiere al total de piezas faltantes que no tiene el tablero

*Le*, se refiere a la longitud en unidades de la linea exterior del area que faltó por rellenar

En el siguiente ejemplo falta una pieza por colocar con un área de 5 unidades y una línea exterior de 12 cuadros, siendo un total de 18 como fitness score.



## 5. Operadores usados

**Selección por ruleta:** Se seleccionan  $n$  individuos de toda la población, la probabilidad con la que estos individuos son seleccionados es proporcional a su nivel de condición física (aptitud). Este método brinda la oportunidad de reproducirse también con un nivel de condición física más bajo.

**Cruce por orden (OX):** Este método fue elegido ya que es el menos disruptivo desde el punto de vista desde el orden del cromosoma. De dicha forma los padres transmitieron la información sobre el orden relativo mientras creaban descendientes totalmente distintos.

P1: [ 7, 10, 6, 1, 3, 5, 0, 8, 12, 9, 11, 4, 2]

P2: [ 1, 11, 8, 4, 7, 10, 6, 3, 0, 2, 5, 12, 9]

H1: [ x, x, x | 1, 3, 5 | x, x, x, x, x, x, x]

H2: [ x, x, x | 4, 7, 10 | x, x, x, x, x, x, x]

F1: [11, 8, 4, 1, 3, 5, 7, 10, 6, 0, 2, 12, 9]

F2: [ 6, 1, 3, 4, 7, 10, 5, 0, 8, 12, 9, 11, 2]

**Mutación por intercambio:** Se usa este método ya que solo se le aplica mutación a toda la población una sola vez, afecta a los cromosomas intercambiando aleatoriamente 2 genes del orden de 2 piezas. La probabilidad de que esto suceda es dada por un parámetro que se proporciona cuando se inicia el algoritmo, que inicialmente se ha establecido que afecte solo al 1% de todos los genes de la población. Como ejemplo tenemos lo siguiente:

Donde la mutación intercambia piezas con el ID 1 y 6

[ 9, 2, 8, 4, 7, 5, 1, 0, 3, 10, 11, 12, 6]

[ 9, 2, 8, 4, 7, 5, 6, 0, 3, 10, 11, 12, 1]

## **6. Criterios de parada**

Para que este programa pueda parar satisfactoriamente necesita cumplir lo siguiente:

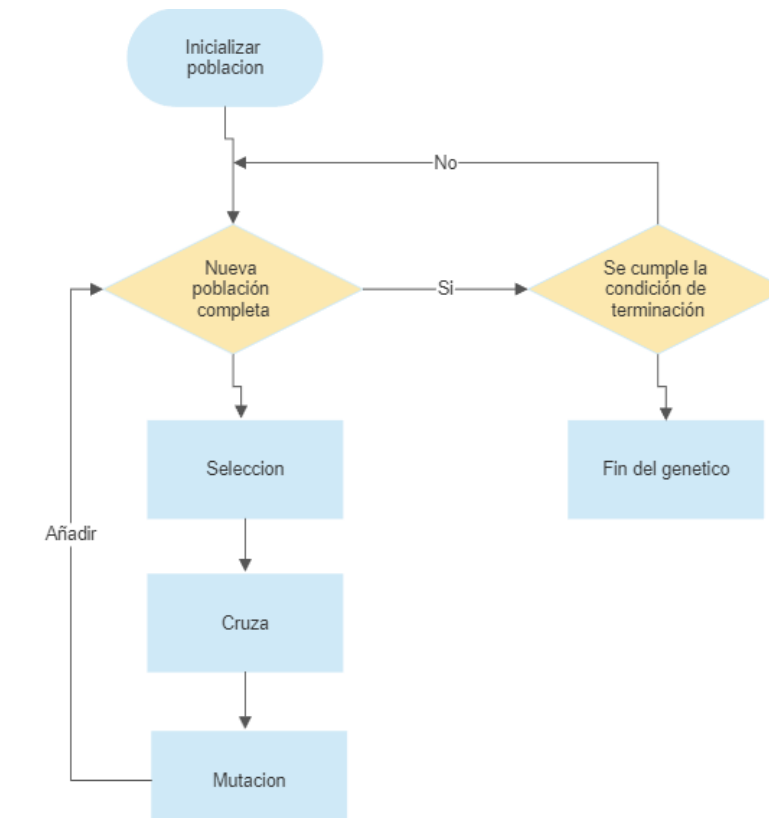
- Llegar al valor de aptitud esperado, en este caso se espera un valor de aptitud de 0 ya que este problema es de minimización.
- Que el numero de generaciones establecido sea alcanzado, en dado caso de no obtener una solución óptima, se muestra el individuo con la mejor puntuación.

## **7. Solución del problema**

En el primer paso, inicializaremos una población de  $N$  individuos generados de manera aleatoria, donde la cantidad de estos se especifica en el inicio del algoritmo como parámetro externo. Después de verificar su nivel de aptitud, verificamos si se cumple la condición de terminación, lo que muy probablemente no sucederá a la primera. Si esto no se cumple, continuamos ingresando a un segundo ciclo de creación de una nueva generación:

1. Seleccionar 2 individuos
2. Crear nueva descendencia

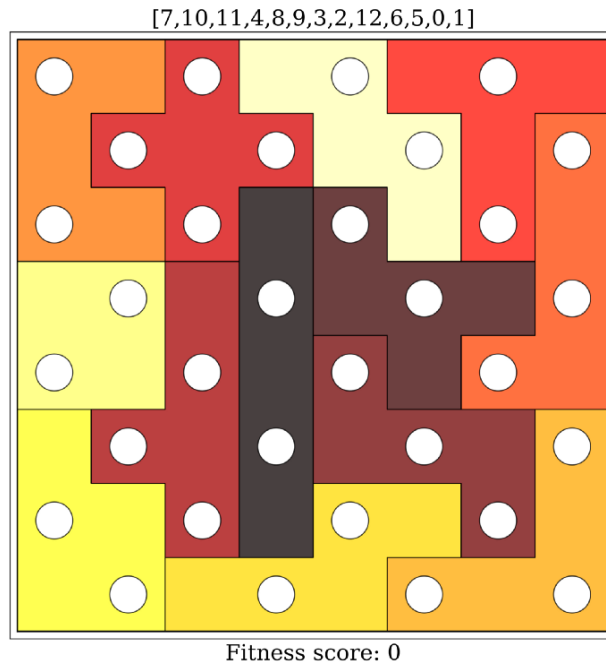
Este ciclo es repetitivo hasta que el tamaño de la población  $N$  se logra, que después de la primera generación es de  $N/2$ , es decir si se especificó un numero de 50, solo 25 individuos serán considerados, después que se genera cada nueva generación se produce la mutación para completarse el ciclo. Surgen  $N$  generaciones, donde  $N$  esta limitado al número que el usuario haya especificado al inicio del algoritmo, si en alguna generación se encuentra algún individuo con aptitud 0 entonces el algoritmo termina.



Se anexa un gráfico de un proceso de evolución aleatoria, donde se generó una población del tamaño de 50, muestra el nivel de condición física de los individuos recién engendrados (eje y) durante una serie de generaciones (eje X) hasta que se haya encontrado la solución (en este caso en la generación 15).

La línea y el área azules muestran la media y la desviación estándar de nivel de aptitud por cada generación.

**Muestra del tablero solucionado**



La solución óptima (**nivel de aptitud 0**) se compone del siguiente individuo **[7,10,11,4,8,9,3,2,12,6,5,0,1]**, donde se muestran todas las piezas acomodadas. Está claro que esta puede ser una solución de varias, se tiene que ejecutar el proceso varias veces obteniendo resultados diferentes, esto puede conllevar diferentes tiempos y además diferentes números de generaciones a converger.

## 8. Pruebas de ejecución

### Ejecución 1:

Para la ejecución 1 se tomaron en cuenta los siguientes parámetros:

- **Tamaño del tablero: 8**
- **Figuras: 13 (todas)**
- **Tamaño de población: 50**
- **Probabilidad de mutación: 0.01**

Ejecución del algoritmo:

Se inicializo con una población generada de manera aleatoria, pasando el tiempo el algoritmo dejo de evolucionar en la **generación 3**, arrojando como mejor individuo al cromosoma **[8,2,0,7,12,9,3,10,5,11,1,6,4]** con una **aptitud de 28.0** sin llegar a una solución con aptitud 0 (solución final), sin embargo, dejamos el algoritmo correr hasta llegar a una generación 22, arrojando el mismo individuo por lo que se consideró como el más optimo.

```
Generacion: 0: Inicializando
```

```
[array([65, 54, 82, 86, 52, 77, 48, 56, 44, 52, 82, 58, 48, 64, 60, 69, 76,
       91, 56, 73, 64, 72, 78, 44, 84, 72, 76, 72, 44, 60, 73, 56, 67, 80,
       76, 86, 89, 69, 77, 53, 65, 66, 83, 72, 65, 74, 59, 93, 58, 81]))]
Puntuacion del mejor candidato: 44.0, cromosoma: [8,2,0,7,12,9,3,10,5,11,1,6,4]
Generacion: 1
```

```
Puntuacion del mejor candidato: 38.0, cromosoma: [8,2,0,7,12,9,3,5,11,1,4,10,6]
Generacion: 2
```

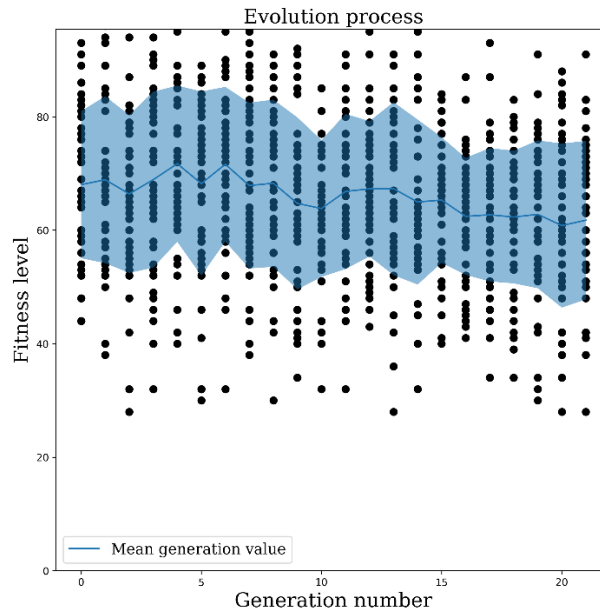
```
Puntuacion del mejor candidato: 28.0, cromosoma: [10,7,2,12,4,9,3,5,11,8,1,0,6]
Generacion: 3
```

```
Puntuacion del mejor candidato: 28.0, cromosoma: [10,7,2,12,4,9,3,5,11,8,1,0,6]
Generacion: 4
```

```
Puntuacion del mejor candidato: 28.0, cromosoma: [10,7,2,12,4,9,3,5,11,8,1,0,6]
Generacion: 5
```

```
Puntuacion del mejor candidato: 28.0, cromosoma: [10,7,2,12,4,9,3,5,11,8,1,0,6]
Generacion: 6
```

La evolución del algoritmo se puede apreciar en la siguiente grafica



## Ejecución 2

Para la ejecución 2 se tomaron en cuenta los siguientes parámetros:

- **Tamaño del tablero: 8**
- **Figuras: 13 (todas)**
- **Tamaño de población: 50**
- **Probabilidad de mutación: 0.09**

Ejecución del algoritmo:

Generacion: 0: Inicializando

```
[array([ 42, 65, 58, 67, 65, 78, 64, 72, 54, 84, 68, 50, 59,
        87, 70, 80, 46, 58, 94, 75, 68, 97, 76, 54, 70, 73,
        71, 83, 72, 101, 70, 56, 97, 55, 72, 60, 60, 95, 83,
        47, 66, 81, 68, 58, 83, 64, 60, 81, 40, 55])]
Puntuacion del mejor candidato: 40.0, cromosoma: [12,3,1,9,7,4,11,10,6,5,2,0,8]
Generacion: 1
```

```
Puntuacion del mejor candidato: 34.0, cromosoma: [0,11,4,1,5,2,9,8,7,10,6,3,12]
Generacion: 2
```

```
Puntuacion del mejor candidato: 34.0, cromosoma: [0,11,4,1,5,2,9,8,7,10,6,3,12]
Generacion: 3
```

```
Puntuacion del mejor candidato: 32.0, cromosoma: [0,11,4,7,10,6,9,5,8,1,2,12,3]
Generacion: 4
```

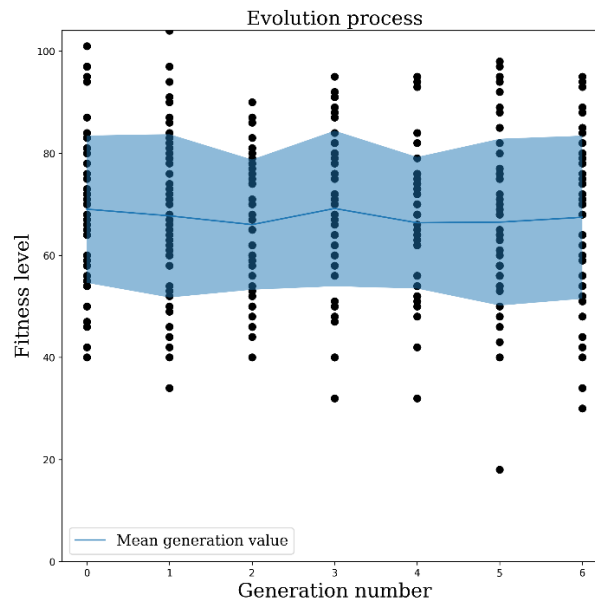
```
Puntuacion del mejor candidato: 32.0, cromosoma: [0,11,4,7,10,6,9,5,8,1,2,12,3]
Generacion: 5
```

```
Puntuacion del mejor candidato: 18.0, cromosoma: [1,9,2,3,10,11,12,8,7,4,0,6,5]
Generacion: 6
```

```
Puntuacion del mejor candidato: 18.0, cromosoma: [1,9,2,3,10,11,12,8,7,4,0,6,5]
Generacion: 7
```

Se inicializo con una población generada de manera aleatoria, pasando el tiempo el algoritmo dejo de evolucionar en la **generación 6**, arrojando como mejor individuo al cromosoma **[1,9,2,3,10,11,12,8,7,4,0,6,5]** con una **aptitud de 18.0** sin llegar a una solución con aptitud 0 (solución final), sin embargo, dejamos el algoritmo correr hasta llegar a una generación 7, arrojando el mismo individuo por lo que se consideró como el más optimo.

La evolución del algoritmo se puede apreciar en la siguiente gráfica:



### Ejecución 3

Para la ejecución 2 se tomaron en cuenta los siguientes parámetros:

- **Tamaño del tablero: 8**
- **Figuras: 13 (todas)**
- **Tamaño de población: 50**
- **Probabilidad de mutación: 0.09**

Ejecución del algoritmo:



```

• Generacion: 0: Inicializando

[array([ 38, 60, 58, 64, 76, 70, 74, 52, 67, 83, 70, 70, 62,
        76, 56, 56, 66, 77, 95, 75, 66, 73, 78, 69, 85, 81,
        54, 65, 75, 75, 58, 81, 60, 62, 58, 56, 76, 78, 87,
        57, 61, 68, 70, 49, 83, 102, 81, 52, 48, 71])]
Puntuacion del mejor candidato: 38.0, cromosoma: [8,4,3,6,0,7,11,10,12,9,5,2,1]
Generacion: 1

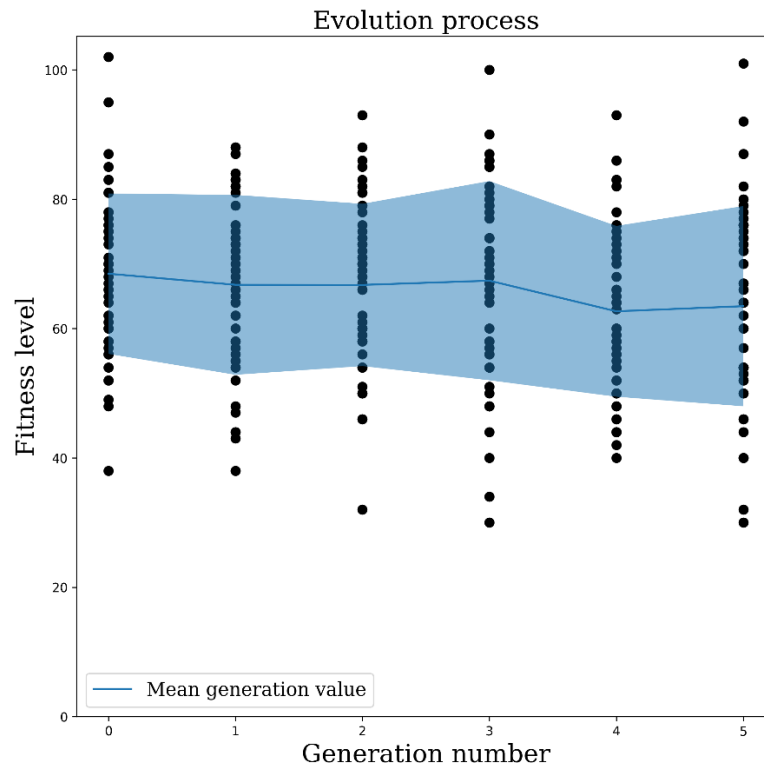
Puntuacion del mejor candidato: 38.0, cromosoma: [8,4,3,6,0,7,11,10,12,9,5,2,1]
Generacion: 2

Puntuacion del mejor candidato: 32.0, cromosoma: [6,5,10,3,0,2,11,4,12,9,7,8,1]
Generacion: 3
0% 0/25 [00:00<?, ?it/s]
Puntuacion del mejor candidato: 30.0, cromosoma: [2,4,9,8,0,3,7,6,10,5,11,1,12]
Generacion: 4
Puntuacion del mejor candidato: 30.0, cromosoma: [2,4,9,8,0,3,7,6,10,5,11,1,12]
Generacion: 5
Puntuacion del mejor candidato: 30.0, cromosoma: [2,4,9,8,0,3,7,6,10,5,11,1,12]
Generacion: 6

```

Se inicializo con una población generada de manera aleatoria, pasando el tiempo el algoritmo dejo de evolucionar en la **generación 4**, arrojando como mejor individuo al cromosoma **[2,4,9,8,0,3,7,6,10,5,11,1,12]** con una **aptitud de 30.0** sin llegar a una solución con aptitud 0 (solución final), sin embargo, dejamos el algoritmo correr hasta llegar a una generación 6, arrojando el mismo individuo por lo que se consideró como el más optimo.

La evolución del algoritmo se puede apreciar en la siguiente gráfica:



#### Ejecución 4

Para la ejecución 2 se tomaron en cuenta los siguientes parámetros:

- **Tamaño del tablero: 8**
- **Figuras: 13 (todas)**
- **Tamaño de población: 50**
- **Probabilidad de mutación: 0.08**

Ejecución del algoritmo:

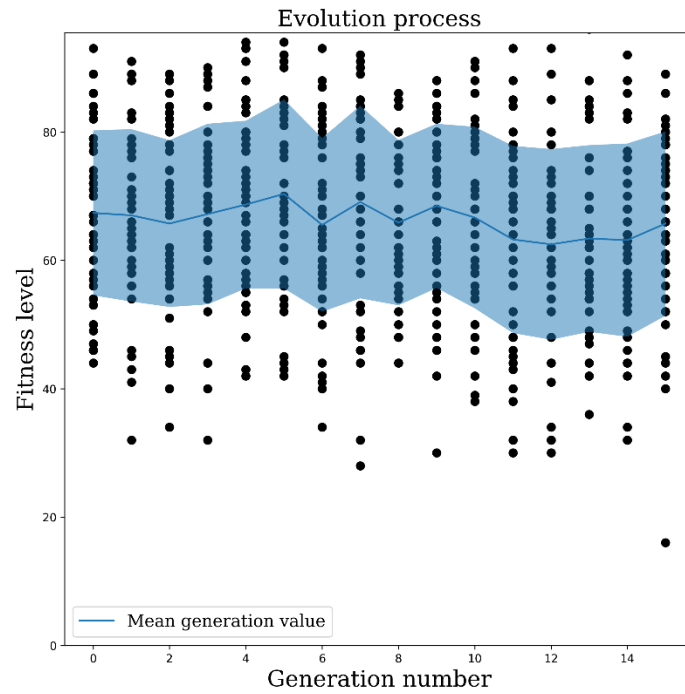
```

Generacion: 0: Inicializando
[array([67, 57, 82, 50, 66, 79, 62, 74, 46, 49, 70, 77, 72, 56, 84, 53, 78,
       93, 71, 47, 67, 77, 86, 46, 86, 47, 58, 50, 60, 63, 83, 79, 82, 67,
       71, 72, 71, 54, 72, 72, 73, 79, 72, 70, 89, 64, 64, 44, 46, 72)])]
Puntuacion del mejor candidato: 44.0, cromosoma: [12,8,7,4,6,5,10,3,1,11,0,9,2]
Generacion: 1
Puntuacion del mejor candidato: 32.0, cromosoma: [1,9,8,3,0,5,2,11,6,7,10,12,4]
Generacion: 2
Puntuacion del mejor candidato: 32.0, cromosoma: [1,9,8,3,0,5,2,11,6,7,10,12,4]
Generacion: 3
Puntuacion del mejor candidato: 32.0, cromosoma: [1,9,8,3,0,5,2,11,6,7,10,12,4]
Generacion: 4
Puntuacion del mejor candidato: 32.0, cromosoma: [1,9,8,3,0,5,2,11,6,7,10,12,4]
Generacion: 5
Puntuacion del mejor candidato: 32.0, cromosoma: [1,9,8,3,0,5,2,11,6,7,10,12,4]
Generacion: 6
Puntuacion del mejor candidato: 32.0, cromosoma: [1,9,8,3,0,5,2,11,6,7,10,12,4]
Generacion: 7
Puntuacion del mejor candidato: 28.0, cromosoma: [5,0,7,10,4,3,12,1,8,11,6,9,2]
Generacion: 8
Puntuacion del mejor candidato: 28.0, cromosoma: [5,0,7,10,4,3,12,1,8,11,6,9,2]
Generacion: 9
Puntuacion del mejor candidato: 28.0, cromosoma: [5,0,7,10,4,3,12,1,8,11,6,9,2]
Generacion: 10
Puntuacion del mejor candidato: 28.0, cromosoma: [5,0,7,10,4,3,12,1,8,11,6,9,2]
Generacion: 11
Puntuacion del mejor candidato: 28.0, cromosoma: [5,0,7,10,4,3,12,1,8,11,6,9,2]
Generacion: 12
Puntuacion del mejor candidato: 28.0, cromosoma: [5,0,7,10,4,3,12,1,8,11,6,9,2]
Generacion: 13
Puntuacion del mejor candidato: 28.0, cromosoma: [5,0,7,10,4,3,12,1,8,11,6,9,2]
Generacion: 14
Puntuacion del mejor candidato: 28.0, cromosoma: [5,0,7,10,4,3,12,1,8,11,6,9,2]
Generacion: 15
Puntuacion del mejor candidato: 16.0, cromosoma: [0,8,6,2,4,3,9,7,11,5,1,10,12]
Generacion: 16

```

Se inicializo con una población generada de manera aleatoria, pasando el tiempo el algoritmo dejo de evolucionar en la **generación 15**, arrojando como mejor individuo al cromosoma **[0,8,6,2,4,3,9,7,11,5,1,10,12]** con una **aptitud de 16.0** sin llegar a una solución con aptitud 0 (solución final), sin embargo, dejamos el algoritmo correr hasta llegar a una generación 16, arrojando el mismo individuo por lo que se consideró como el más óptimo.

La evolución del algoritmo se puede apreciar en la siguiente gráfica:



### Ejecución 5:

Para la ejecución 2 se tomaron en cuenta los siguientes parámetros:

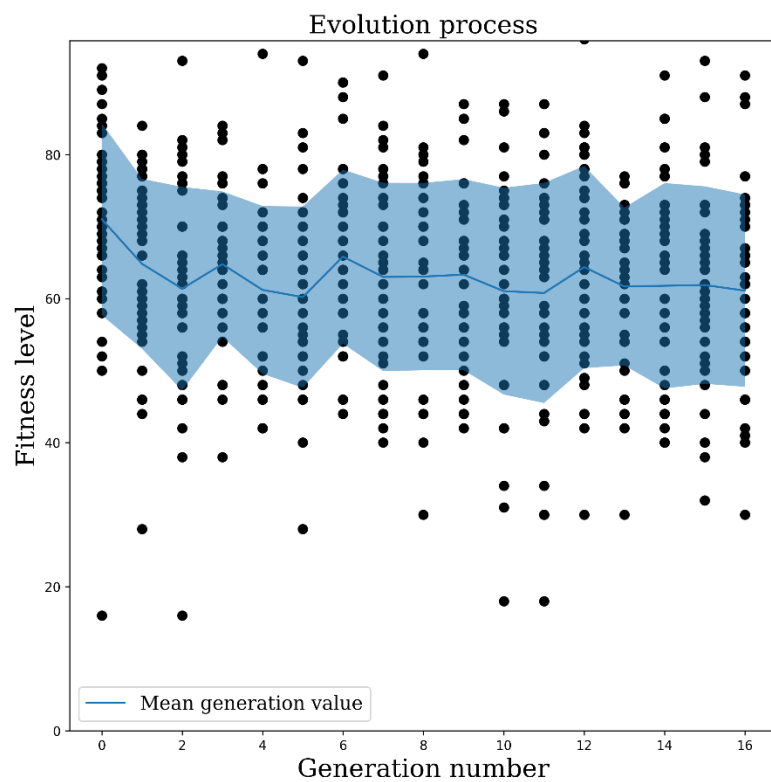
- **Tamaño del tablero: 8**
- **Figuras: 13 (todas)**
- **Tamaño de población: 40**
- **Probabilidad de mutación: 0.05**

Ejecución del algoritmo:

```
Generacion: 0: Inicializando
[array([76, 79, 72, 76, 87, 70, 66, 68, 50, 54, 70, 79, 67, 75, 69, 75, 77,
      80, 75, 64, 91, 85, 78, 67, 74, 71, 60, 76, 89, 58, 61, 63, 92, 66,
      83, 84, 16, 70, 52, 76])]
Puntuacion del mejor candidato: 16.0, cromosoma: [2,3,5,8,1,10,0,6,4,7,9,11,12]
Generacion: 1
Puntuacion del mejor candidato: 16.0, cromosoma: [2,3,5,8,1,10,0,6,4,7,9,11,12]
Generacion: 2
Puntuacion del mejor candidato: 16.0, cromosoma: [2,3,5,8,1,10,0,6,4,7,9,11,12]
Generacion: 3
Puntuacion del mejor candidato: 16.0, cromosoma: [2,3,5,8,1,10,0,6,4,7,9,11,12]
Generacion: 4
Puntuacion del mejor candidato: 16.0, cromosoma: [2,3,5,8,1,10,0,6,4,7,9,11,12]
Generacion: 5
Puntuacion del mejor candidato: 16.0, cromosoma: [2,3,5,8,1,10,0,6,4,7,9,11,12]
Generacion: 6
Puntuacion del mejor candidato: 16.0, cromosoma: [2,3,5,8,1,10,0,6,4,7,9,11,12]
Generacion: 7
Puntuacion del mejor candidato: 16.0, cromosoma: [2,3,5,8,1,10,0,6,4,7,9,11,12]
Generacion: 8
Puntuacion del mejor candidato: 16.0, cromosoma: [2,3,5,8,1,10,0,6,4,7,9,11,12]
Generacion: 9
Puntuacion del mejor candidato: 16.0, cromosoma: [2,3,5,8,1,10,0,6,4,7,9,11,12]
Generacion: 10
Puntuacion del mejor candidato: 16.0, cromosoma: [2,3,5,8,1,10,0,6,4,7,9,11,12]
Generacion: 11
Puntuacion del mejor candidato: 16.0, cromosoma: [2,3,5,8,1,10,0,6,4,7,9,11,12]
Generacion: 12
Puntuacion del mejor candidato: 16.0, cromosoma: [2,3,5,8,1,10,0,6,4,7,9,11,12]
Generacion: 13
Puntuacion del mejor candidato: 16.0, cromosoma: [2,3,5,8,1,10,0,6,4,7,9,11,12]
Generacion: 14
Puntuacion del mejor candidato: 16.0, cromosoma: [2,3,5,8,1,10,0,6,4,7,9,11,12]
Generacion: 15
Puntuacion del mejor candidato: 16.0, cromosoma: [2,3,5,8,1,10,0,6,4,7,9,11,12]
Generacion: 16
Puntuacion del mejor candidato: 16.0, cromosoma: [2,3,5,8,1,10,0,6,4,7,9,11,12]
```

Se inicializo con una población generada de manera aleatoria, pasando el tiempo el algoritmo dejo de evolucionar en la **generación 1**, arrojando como mejor individuo al cromosoma **[2,3,5,8,1,10,0,6,4,7,9,11,12]** con una **aptitud de 16.0** sin llegar a una solución con aptitud 0 (solución final), sin embargo, dejamos el algoritmo correr hasta llegar a una generación 16, arrojando el mismo individuo por lo que se consideró como el más óptimo.

La evolución del algoritmo se puede apreciar en la siguiente gráfica:



## 9. Pruebas de ejecución con el profesor

### Prueba 1:

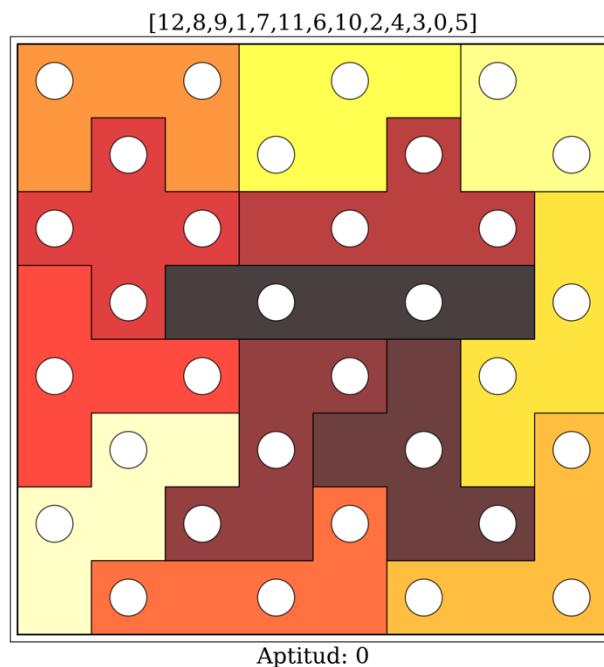
Parámetros para esta prueba

- Numero de generaciones: 50
- Tamaño de población: 100
- Tamaño del torneo: 2
- Probabilidad de mutación 0.0

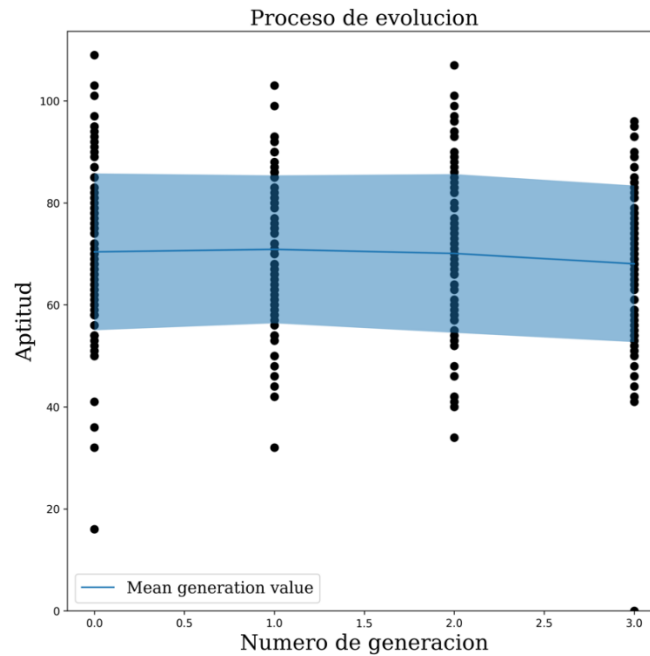
```
Generacion: 0: Inicializando
Numero de generaciones: 50 Poblacion: 100 Tamaño del torneo: 2 Probabilidad de mutacion: 0.0
Aptitud del mejor candidato: 16.0, cromosoma: [2,3,6,12,10,8,1,11,7,0,4,5,9]
Generacion: 1
Aptitud del mejor candidato: 16.0, cromosoma: [2,3,6,12,10,8,1,11,7,0,4,5,9]
Generacion: 2
Aptitud del mejor candidato: 16.0, cromosoma: [2,3,6,12,10,8,1,11,7,0,4,5,9]
Generacion: 3
Aptitud del mejor candidato: 0.0, cromosoma: [12,8,9,1,7,11,6,10,2,4,3,0,5]
----- Resultados -----
Se ha concluido de armar el puzzle con exito
Solucion (cromosoma): [12,8,9,1,7,11,6,10,2,4,3,0,5]
Tablero guardado
```

---

Resultados de la prueba: Inicialmente se obtuvo una buena población de individuos, seleccionando el mejor con su aptitud de **16.0** posteriormente surgieron 3 generaciones más, logrando en la 4ta obtener un individuo cuya aptitud fue 0, donde su cadena fue de **[12,8,9,1,7,11,6,10,2,4,3,0,5]**. Obteniendo el siguiente resultado gráficamente.



Donde nuestro grafico de evolución quedó de la siguiente manera:



## Prueba 2

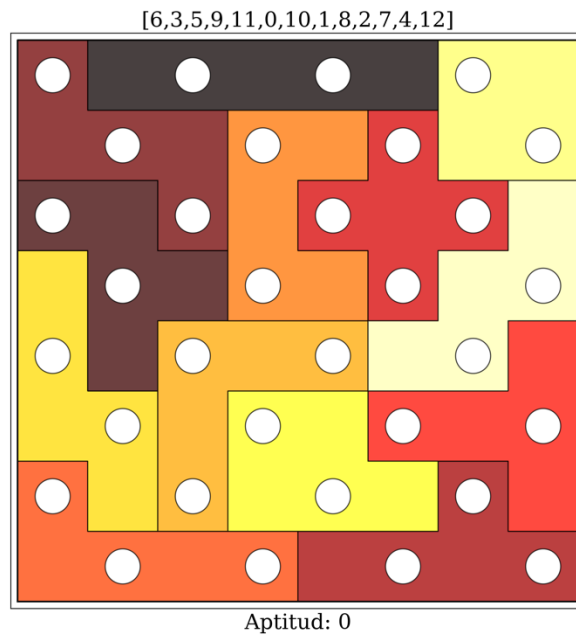
Parámetros para esta prueba:

- Numero de generaciones: 100
- Población: 50
- Tamaño del torneo: 3
- Probabilidad de mutación: 0.1

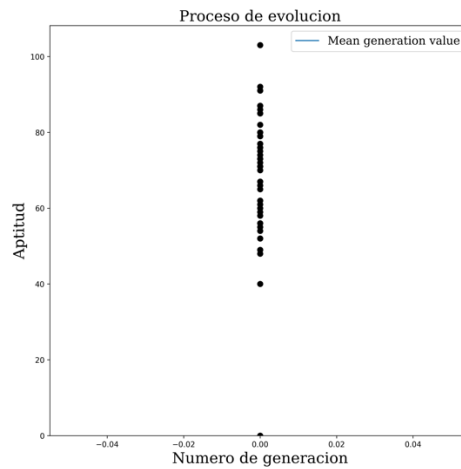
```
Generacion: 0: Inicializando
Numero de generaciones: 100 Poblacion: 50 Tamaño del torneo: 3 Probabilidad de mutacion: 0.1
Aptitud del mejor candidato: 0.0, cromosoma: [6,3,5,9,11,0,10,1,8,2,7,4,12]
----- Resultados -----
Se ha concluido de armar el puzzle con exito
Solucion (cromosoma): [6,3,5,9,11,0,10,1,8,2,7,4,12]
Tablero guardado
```

Resultados de la prueba: Inicialmente se obtuvo una buena población de individuos, de tal manera que en la primera generación se obtuvo un individuo cuya aptitud fue de **0.0** es decir, en la primera generación se obtuvo un resultado optimo, dando la cadena de **[6,3,5,9,11,0,10,1,8,2,7,4,12]**. Obteniendo el siguiente resultado gráficamente.





Donde nuestro grafico de evolución quedó de la siguiente manera:



### Prueba 3

Parámetros para esta prueba:

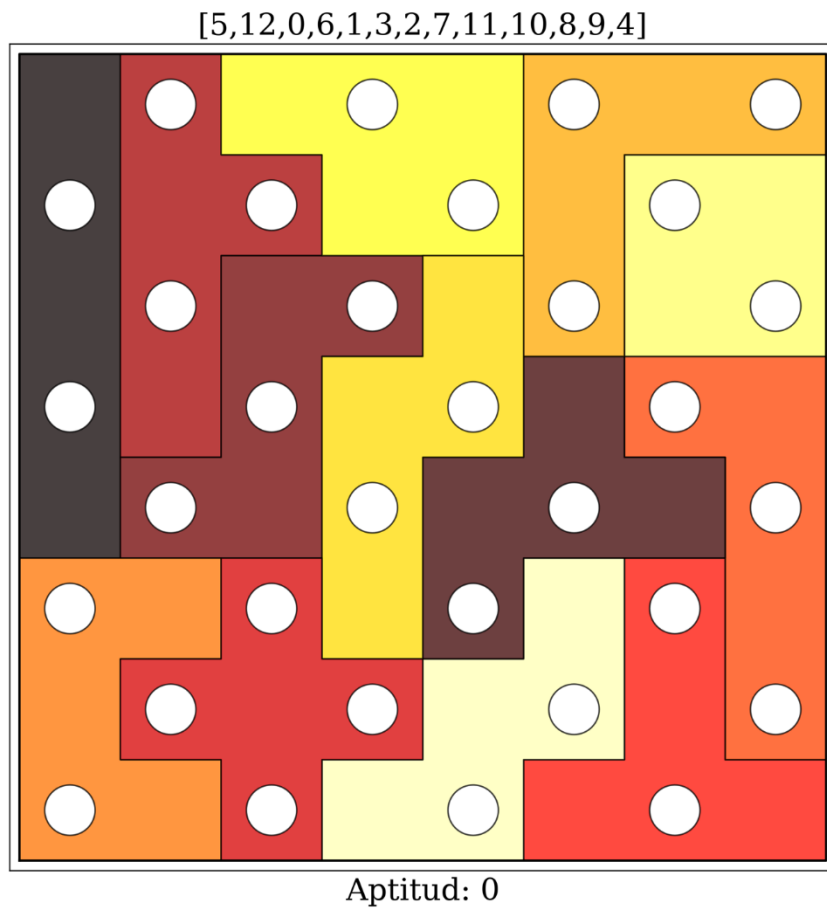
- Numero de generaciones: 100
- Población: 100
- Tamaño del torneo: 3
- Probabilidad de mutación: 0.01

```

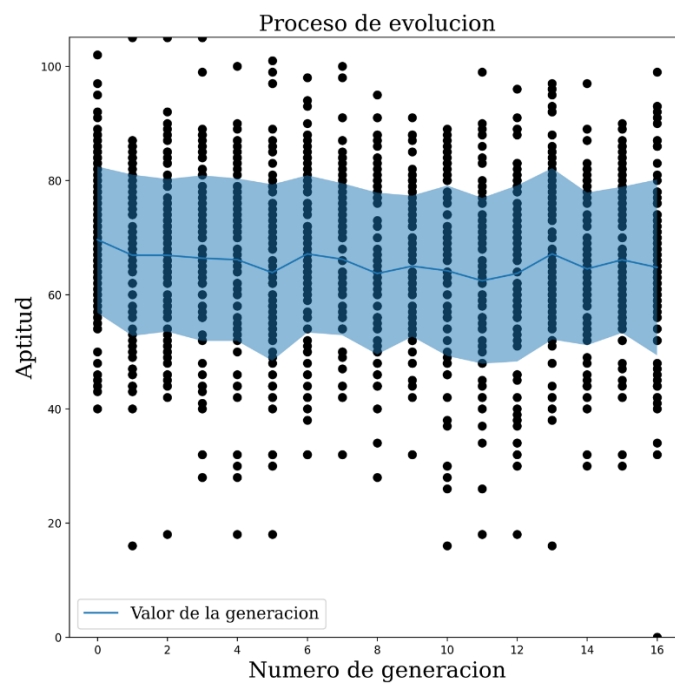
Generacion: 0: Inicializando
Numero de generaciones: 100 Poblacion: 100 Tamaño del torneo: 3 Probabilidad de mutacion: 0.01
Aptitud del mejor candidato: 40.0, cromosoma: [8,12,4,5,7,10,9,2,11,6,1,0,3]
Generacion: 1
Aptitud del mejor candidato: 16.0, cromosoma: [8,5,4,3,9,11,2,0,1,6,10,12,7]
Generacion: 2
Aptitud del mejor candidato: 16.0, cromosoma: [8,5,4,3,9,11,2,0,1,6,10,12,7]
Generacion: 3
Aptitud del mejor candidato: 16.0, cromosoma: [8,5,4,3,9,11,2,0,1,6,10,12,7]
Generacion: 4
Aptitud del mejor candidato: 16.0, cromosoma: [8,5,4,3,9,11,2,0,1,6,10,12,7]
Generacion: 5
Aptitud del mejor candidato: 16.0, cromosoma: [8,5,4,3,9,11,2,0,1,6,10,12,7]
Generacion: 6
Aptitud del mejor candidato: 16.0, cromosoma: [8,5,4,3,9,11,2,0,1,6,10,12,7]
Generacion: 7
Aptitud del mejor candidato: 16.0, cromosoma: [8,5,4,3,9,11,2,0,1,6,10,12,7]
Generacion: 8
Aptitud del mejor candidato: 16.0, cromosoma: [8,5,4,3,9,11,2,0,1,6,10,12,7]
Generacion: 9
Aptitud del mejor candidato: 16.0, cromosoma: [8,5,4,3,9,11,2,0,1,6,10,12,7]
Generacion: 10
Aptitud del mejor candidato: 16.0, cromosoma: [8,5,4,3,9,11,2,0,1,6,10,12,7]
Generacion: 11
Aptitud del mejor candidato: 16.0, cromosoma: [8,5,4,3,9,11,2,0,1,6,10,12,7]
Generacion: 12
Aptitud del mejor candidato: 16.0, cromosoma: [8,5,4,3,9,11,2,0,1,6,10,12,7]
Generacion: 13
Aptitud del mejor candidato: 16.0, cromosoma: [8,5,4,3,9,11,2,0,1,6,10,12,7]
Generacion: 14
Aptitud del mejor candidato: 16.0, cromosoma: [8,5,4,3,9,11,2,0,1,6,10,12,7]
Generacion: 15
Aptitud del mejor candidato: 16.0, cromosoma: [8,5,4,3,9,11,2,0,1,6,10,12,7]
Generacion: 16
Aptitud del mejor candidato: 0.0, cromosoma: [5,12,0,6,1,3,2,7,11,10,8,9,4]
----- Resultados -----
Se ha concluido de armar el puzzle con exito
Solucion (cromosoma): [5,12,0,6,1,3,2,7,11,10,8,9,4]
Tablero guardado

```

Resultados de la prueba: Inicialmente se obtuvo un individuo con una aptitud de **40.0**, posteriormente transcurrieron 17 generaciones para encontrar a un resultado optimo, dando la cadena de **[5,12,0,6,1,3,2,7,11,10,8,9,4]**. Obteniendo el siguiente resultado gráficamente.



Donde nuestro grafico de evolución quedó de la siguiente manera:



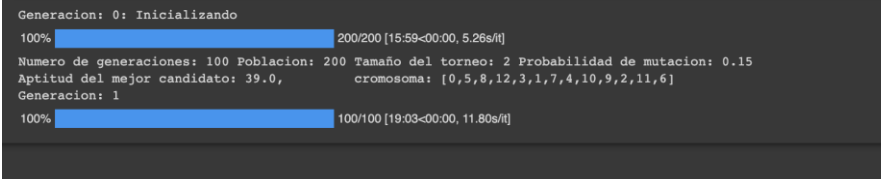
## Prueba 4

Parámetros para esta prueba:

- Numero de generaciones: 100
- Población: 200
- Tamaño del torneo: 2
- Probabilidad de mutación: 0.15

En esta prueba, tardo en generar la primera población de **200** individuos aproximadamente 30 minutos, esto debido a todas las combinaciones que se tienen que realizar para meterlas cada una de ellas a un individuo y a partir de ahí armar una población, posteriormente en la generación 2 el algoritmo se quedó colgado, sin obtener algún resultado como salida final.

```
[[0, 0, 1], [1, 0, 0], [0, 1, 0], [1, 1, 1]],  
[[0, 0, 0], [1, 1, 0], [2, 2, 0], [0, 1, 1], [1, 2, 1]]  
])  
polygons = create_polygons(center_blocks)  
  
if __name__ == '__main__':  
  
    evo = Evolution(200, polygons, board_size=8, mutation_probability=0.15, generations=100, tournament_size=2)  
    evo.run('./proceso.png')
```



## Prueba 5:

Parámetros de prueba

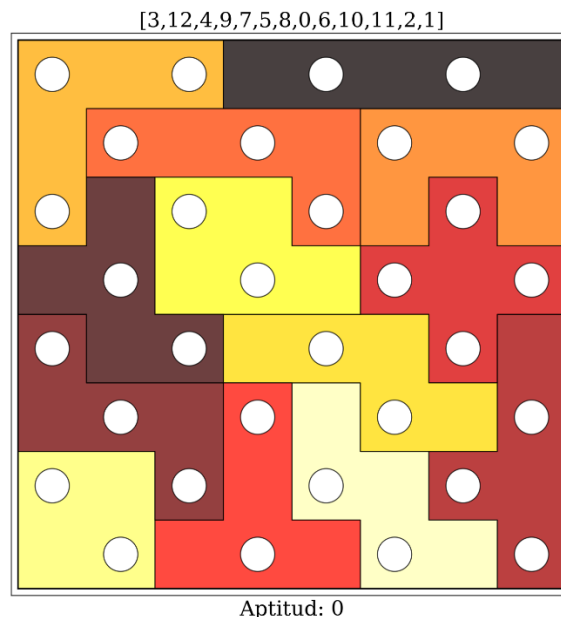
- Numero de generaciones: 30
- Población: 40
- Tamaño del torneo: 2
- Probabilidad de mutación: 0.04

```

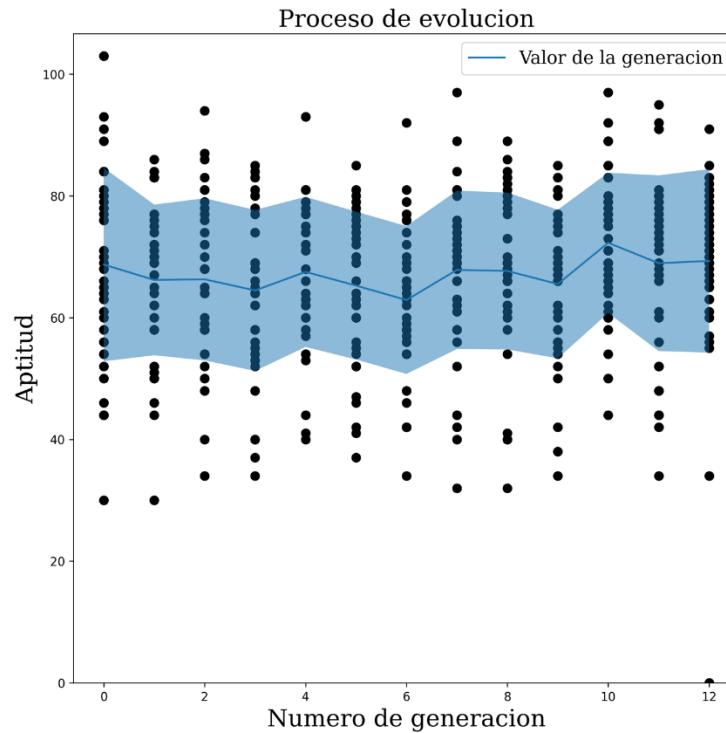
Generacion: 0: Inicializando
Numero de generaciones: 30 Poblacion: 40 Tamaño del torneo: 2 Probabilidad de mutacion: 0.04
Aptitud del mejor candidato: 30.0, cromosoma: [6,4,9,1,3,10,11,8,0,5,7,2,12]
Generacion: 1
Aptitud del mejor candidato: 30.0, cromosoma: [6,4,9,1,3,10,11,8,0,5,7,2,12]
Generacion: 2
Aptitud del mejor candidato: 30.0, cromosoma: [6,4,9,1,3,10,11,8,0,5,7,2,12]
Generacion: 3
Aptitud del mejor candidato: 30.0, cromosoma: [6,4,9,1,3,10,11,8,0,5,7,2,12]
Generacion: 4
Aptitud del mejor candidato: 30.0, cromosoma: [6,4,9,1,3,10,11,8,0,5,7,2,12]
Generacion: 5
Aptitud del mejor candidato: 30.0, cromosoma: [6,4,9,1,3,10,11,8,0,5,7,2,12]
Generacion: 6
Aptitud del mejor candidato: 30.0, cromosoma: [6,4,9,1,3,10,11,8,0,5,7,2,12]
Generacion: 7
Aptitud del mejor candidato: 30.0, cromosoma: [6,4,9,1,3,10,11,8,0,5,7,2,12]
Generacion: 8
Aptitud del mejor candidato: 30.0, cromosoma: [6,4,9,1,3,10,11,8,0,5,7,2,12]
Generacion: 9
Aptitud del mejor candidato: 30.0, cromosoma: [6,4,9,1,3,10,11,8,0,5,7,2,12]
Generacion: 10
Aptitud del mejor candidato: 30.0, cromosoma: [6,4,9,1,3,10,11,8,0,5,7,2,12]
Generacion: 11
Aptitud del mejor candidato: 30.0, cromosoma: [6,4,9,1,3,10,11,8,0,5,7,2,12]
Generacion: 12
Aptitud del mejor candidato: 0.0, cromosoma: [3,12,4,9,7,5,8,0,6,10,11,2,1]
----- Resultados -----
Se ha concluido de armar el puzzle con exito
Solucion (cromosoma): [3,12,4,9,7,5,8,0,6,10,11,2,1]
Tablero guardado

```

Resultados de la prueba: Inicialmente se obtuvo un individuo con una aptitud de **30.0**, posteriormente transcurrieron 11 generaciones para encontrar a un resultado optimo, dando la cadena de **[3,12,4,9,7,5,8,0,6,10,11,2,1]**. Obteniendo el siguiente resultado gráficamente.



Donde nuestro grafico de evolución quedó de la siguiente manera:

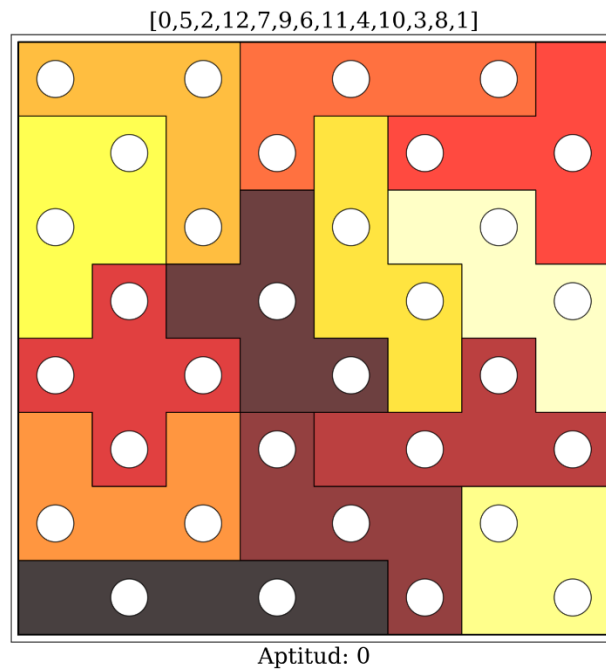


## Prueba 6

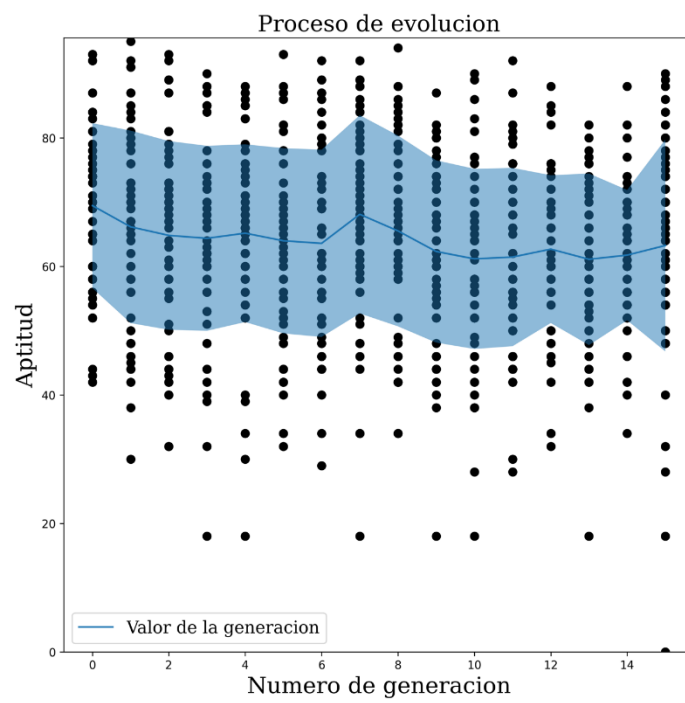
Parámetros de prueba

- Numero de generaciones: 20
- Población: 60
- Tamaño del torneo: 2
- Probabilidad de mutación: 0.0

Resultados de la prueba: Inicialmente se obtuvo un individuo con una aptitud de **25.0**, posteriormente transcurrieron 15 generaciones para encontrar a un resultado optimo, dando la cadena de **[0,5,2,12,7,9,6,11,4,10,3,8,1]**. Obteniendo el siguiente resultado gráficamente.



Donde nuestro grafico de evolución quedo de la siguiente manera



## 10. Conclusiones

Este proyecto fue bastante interesante ya que se abordó un tema que no es tan conocido, todos los algoritmos aquí tratados pueden ser sensibles a la hora de ser trabajados, ya que se deben saber trabajar de acuerdo con las especificaciones del proyecto, de lo contrario puede no funcionar correctamente el genético.

La construcción de este proyecto fue compleja a la vez que ardua, ya que fue complicado entender las diversas fases de un individuo y que hacer para que la función de aptitud mejorara, sin embargo, una vez que todos los puntos se comprenden con claridad es bastante entendible armar todo.

## 11. Referencias

- *FOGEL, D. B. Evolutionary Computation. IEEE Press, New York, 1995.*
- *Sam Loyd's broken chess board.* (s. f.). <https://mmaca.cat/en/moduls/tauler-escacs-trencat/>

```
Generacion: 0: Inicializando
[array([61, 64, 54, 83, 32, 62, 76, 95, 62, 50, 82, 84, 64, 76, 67, 74, 85,
        64, 64, 87, 83, 76, 80, 68, 86, 62, 48, 87, 64, 78, 62, 82, 64, 61,
        60, 68, 68, 66, 78, 73])]
Puntuacion del mejor candidato: 32.0, cromosoma: [7,10,12,6,11,8,0,9,5,1,4,3,2]
Generacion: 1
Puntuacion del mejor candidato: 32.0, cromosoma: [7,10,12,6,11,8,0,9,5,1,4,3,2]
Generacion: 2
Puntuacion del mejor candidato: 32.0, cromosoma: [7,10,12,6,11,8,0,9,5,1,4,3,2]
Generacion: 3
Puntuacion del mejor candidato: 32.0, cromosoma: [7,10,12,6,11,8,0,9,5,1,4,3,2]
Generacion: 4
Puntuacion del mejor candidato: 31.0, cromosoma: [12,3,2,7,4,8,0,6,9,5,1,11,10]
Generacion: 5
Puntuacion del mejor candidato: 31.0, cromosoma: [12,3,2,7,4,8,0,6,9,5,1,11,10]
Generacion: 6
Puntuacion del mejor candidato: 31.0, cromosoma: [12,3,2,7,4,8,0,6,9,5,1,11,10]
Generacion: 7
Puntuacion del mejor candidato: 26.0, cromosoma: [6,3,9,5,1,12,10,11,4,8,7,2,0]
Generacion: 8
Puntuacion del mejor candidato: 26.0, cromosoma: [6,3,9,5,1,12,10,11,4,8,7,2,0]
Generacion: 9
Puntuacion del mejor candidato: 0.0, cromosoma: [6,3,5,12,10,9,8,2,11,0,4,7,1]
```