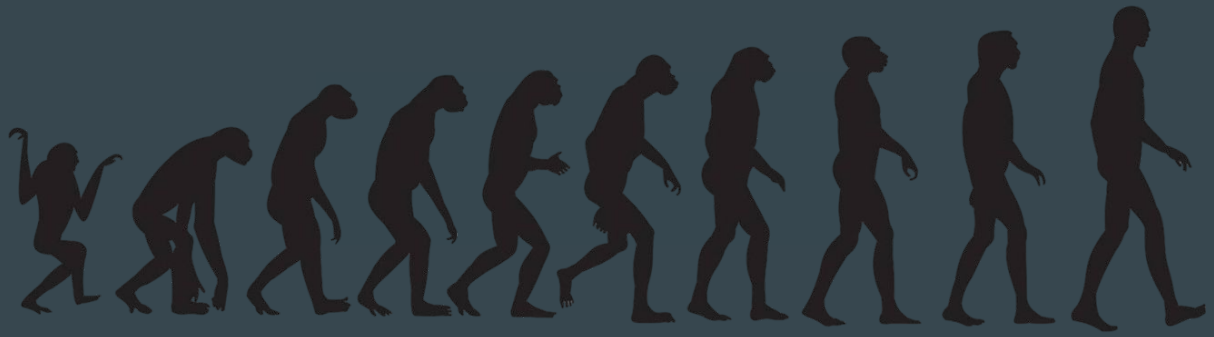
Sistemas de Inteligencia Artificial: TP2 - Algoritmos Genéticos



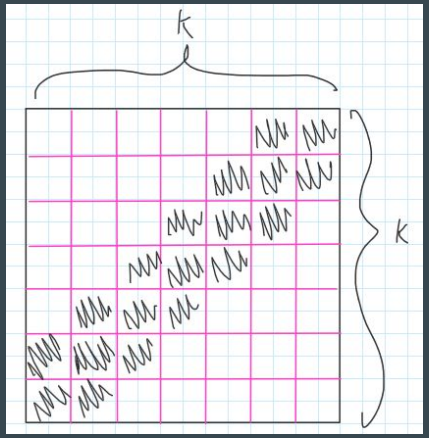
ITBA

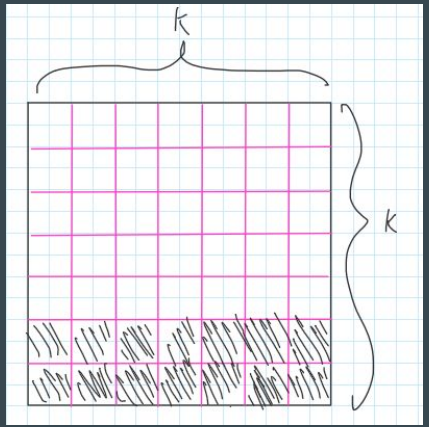
Franco Morroni (60417)
Bernardo Zapico (62318)

Ejercicio 1 - Reconstrucción de Imagen con caracteres ASCII

Ejercicio 1 - Idea

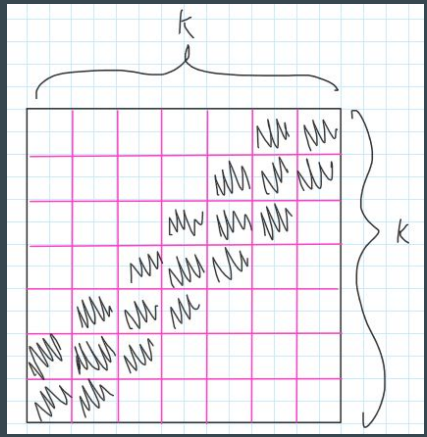
- Se busca representar varios píxeles de una imagen con un solo carácter
- Se elige una submatriz KxK que sirva para dibujar un carácter como representación de KxK pixeles en la imagen original. Para el ejemplo de una “/” y “_”se tomó K = 7
- La matriz tendrá 0s o 1s
- La representación total será una matriz de MxM siendo M múltiplo de K
- En caso de que sobren menos píxeles que K en alguna coordenada, se rellena con espacios vacíos la parte de la imagen que queda fuera de la submatriz KxK





Ejercicio 1 - Idea

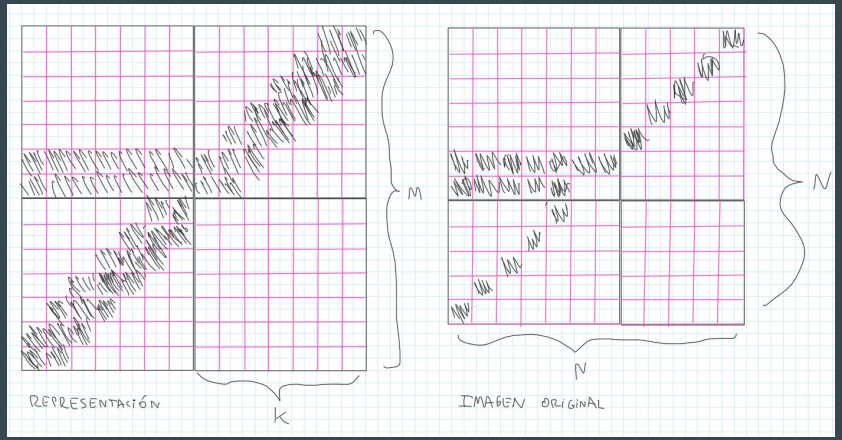
- Para cada representación de nuestro alfabeto de caracteres, tendremos su máscara correspondiente
- Se puede ver que si yo tuviera una porción de la imagen de tamaño KxK y la comparara con mi representación de un carácter; podría ver qué tan apropiada es yendo pixel por pixel en la porción y contando la cantidad de veces que coinciden en color blanco o negro
- Podría realizar esta comparación para cada submatriz KxK en mi representación en la matriz MxM para ver qué tan parecida es a la imagen NxN original




```
mask_slash = [ 0, 0, 0, 0, 0, 1, 1,
               0, 0, 0, 0, 1, 1, 1,
               0, 0, 0, 1, 1, 1, 0,
               0, 0, 1, 1, 1, 0, 0,
               0, 1, 1, 1, 0, 0, 0,
               1, 1, 1, 0, 0, 0, 0,
               1, 1, 0, 0, 0, 0, 0
             ]
```

Ejercicio 1 - Estructura

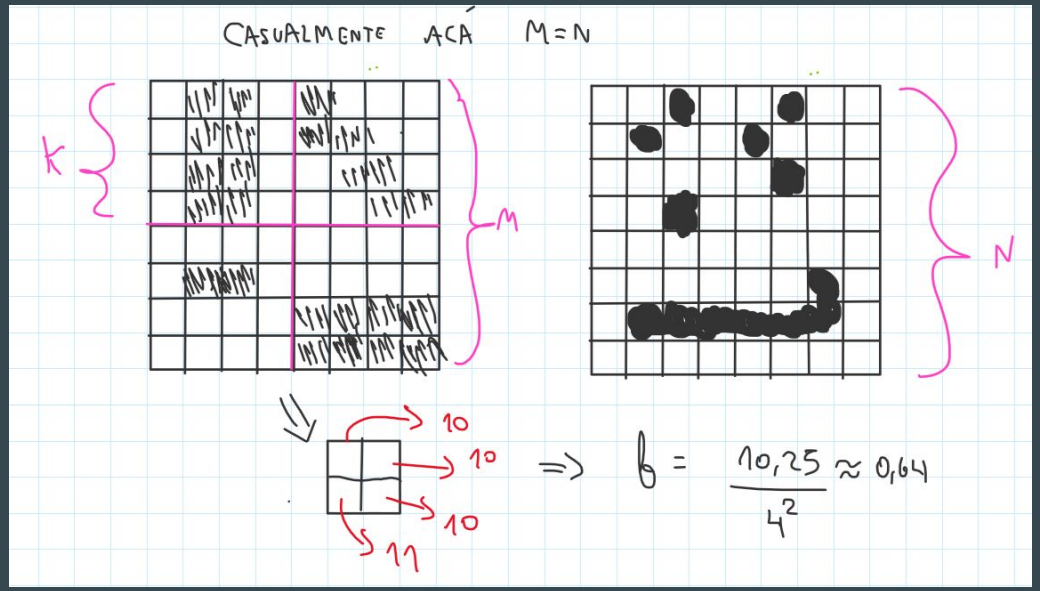
- Individuos: representaciones en una matriz MxM de la imagen de tamaño NxN. Tiene tantos genes como submatrices KxK entren en su matriz.
- Gen: máscara de 0s o 1s que representan la forma de los símbolos en su forma matricial con K^2 cantidad de dígitos. Ej: [100; 010; 001] -> 100010001 sería “\” en K=3.
- Alelos: máscaras de todos los caracteres posibles que quiera usar en mi representación.
- Población inicial: Se empieza con una cantidad fija de población, cada individuo tiene una matriz de genes



Ejercicio 1 - Fitness

- Primero se convierte la imagen original a blanco y negro, y luego se toma un valor límite a partir del cual un dado pixel se considera “prendido” o “apagado”.
- Para cada gen se comparan sus pixeles con los pixeles equivalentes en la imagen y se toma el total de pixeles que coinciden en prendido/apagado.
- $< >_g$: Promedio del total de pixeles coincidentes para todos los genes.
- Divido por K^2 para normalizar el resultado a una escala [0, 1] con 1 siendo un match perfecto

$$fitness = \frac{<casilleros\ iguales>_g}{K^2}$$



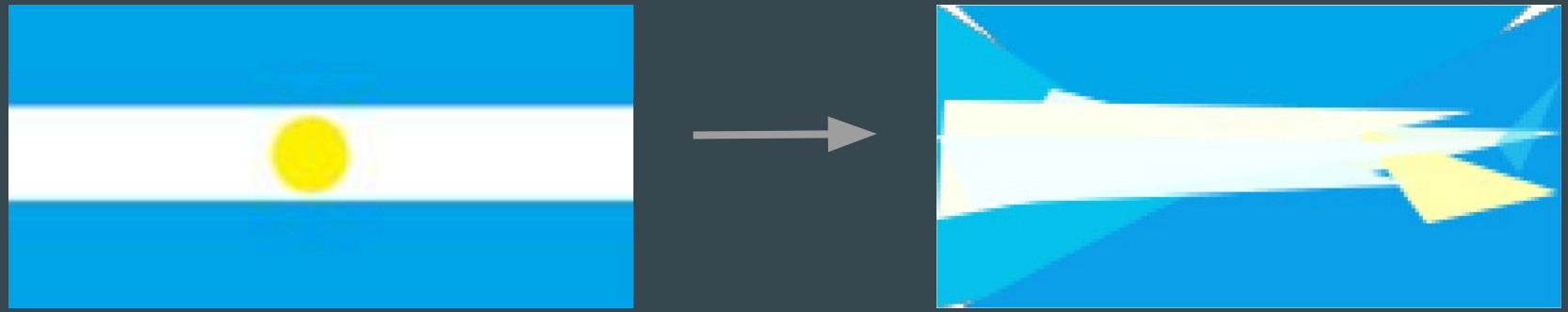
Ejercicio 1 - Crossover, mutación y condición de corte

- Para el crossover, la idea sería intercambiar los genes (máscaras de caracteres) de un padre por los de otro, ya se en un punto, dos, etc.
- La mutación, implicaría que los genes puedan mutar a otros alelos aleatoriamente.
- Como se busca una solución representativa se deberá poner un fitness mínimo de condición de corte, pero, en caso de no encontrarla, se deberá establecer una generación máxima para que no corra el algoritmo para siempre

Adicional: Resolución de imagen generada

El análisis mencionado compara un pixel del gen contra un pixel de la imagen. Esto daría la resolución más alta posible. Pero si queremos reducir la resolución se podría considerar que cada pixel del gen representa n x n pixeles de la imagen real, y en ese caso para el fitness se compararía cada pixel del gen contra el promedio de los pixeles n x n de la imagen.

Ejercicio 2 - Reconstrucción de imagen con triángulos



Estructura

- Individuos: cada individuo contiene una lista de N triángulos, donde N es un parámetro del algoritmo.
- Gen: una terna de tuplas (x,y) que representan los vértices de un triángulo y una tupla (r,g,b,a) que representa el color. Los vértices no se pueden pasar de las dimensiones de la imagen y cada componente del color está en intervalo [0, 1].
- Alelos: en el canvas a dibujar, todas las posibles representaciones de un triángulo. Puede variar: posición de sus vértices, color y transparencia.
- Población inicial: Se empieza con una cantidad fija de población, normalmente tomamos 100. Cada individuo tiene una lista de triángulos generados de manera azarosa.

Fitness:

El objetivo, es poder comparar qué tan parecida es la representación del individuo a la imagen objetivo. Se planteó:

Distancia Euclidiana entre dos imágenes:

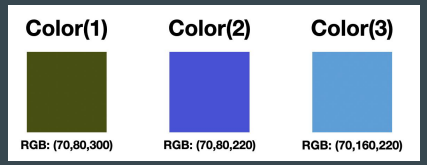
- Toma el arreglo de píxeles en formato RGBA y calcula su diferencia Euclidiana

$$dist\_euc(p1,p2)^2 = (\sqrt{(p1.r - p2.r)^2 + (p1.g - p2.g)^2 + (p1.b - p2.b)^2})^2$$

- Se eleva al cuadrado esta distancia para exigir que la estimación sea más exacta

✔️ Es bastante rápida

❌ No siempre trata la diferencia de colores como lo haríamos las personas



Diferencia Perceptual (ΔE) entre dos imágenes

- Toma el arreglo de píxeles en formato LAB (**L**: Luminosidad, **A**: dimensión entre verde y rojo, **B**: dimensión entre azul y amarillo)

✔️ Es mejor para diferencias visuales

❌ La transformación de píxeles RGB a LAB es costosa

Delta_E	Perception	Examples
<= 1.0	Not perceptible by human eyes	#8B8A88 #8B8A87 delta_E = 1
1 - 2	Perceptible through close observation	#E9DC87 #E8D97C delta_E = 2
2 - 10	Perceptible at glance	#85D2D8 #93D9F1 delta_E = 10
11 - 49	Color are more similar than opposite	#19EB74 #A9C7AE delta_E = 20
100	Color are exact opposite	#000000 #FFFFFF delta_E = 100

Condición de corte

Propuestas:

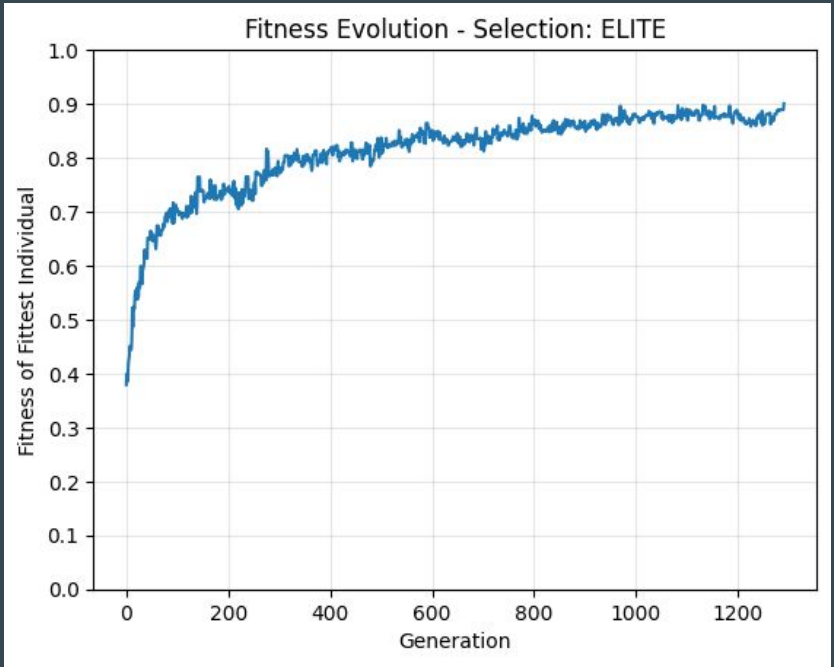
- Cantidad máxima de generaciones
- Mínimo fitness necesario

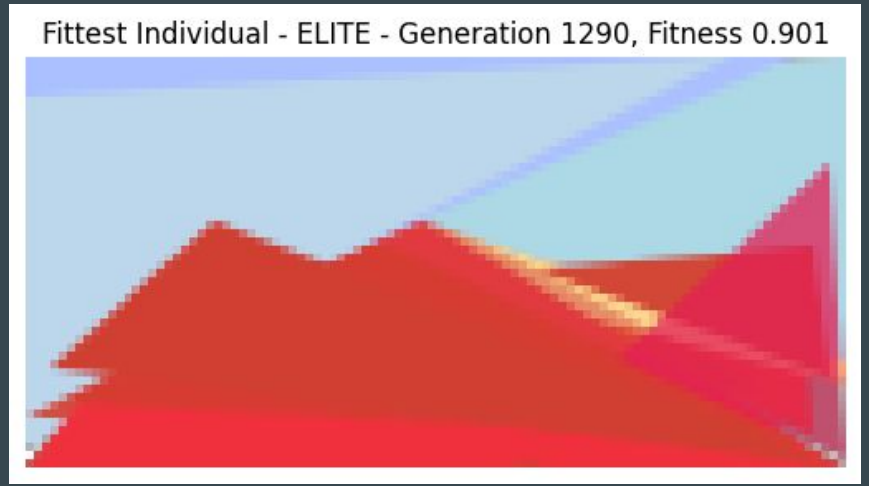
En lugar de quedarnos con una, decidimos cortar el algoritmo genético tan pronto suceda **cualquiera de las dos** . Esto es para lograr que:

- Si se encuentra la solución esperada, terminar
- Si no se encuentra, que no corra infinitamente el algoritmo

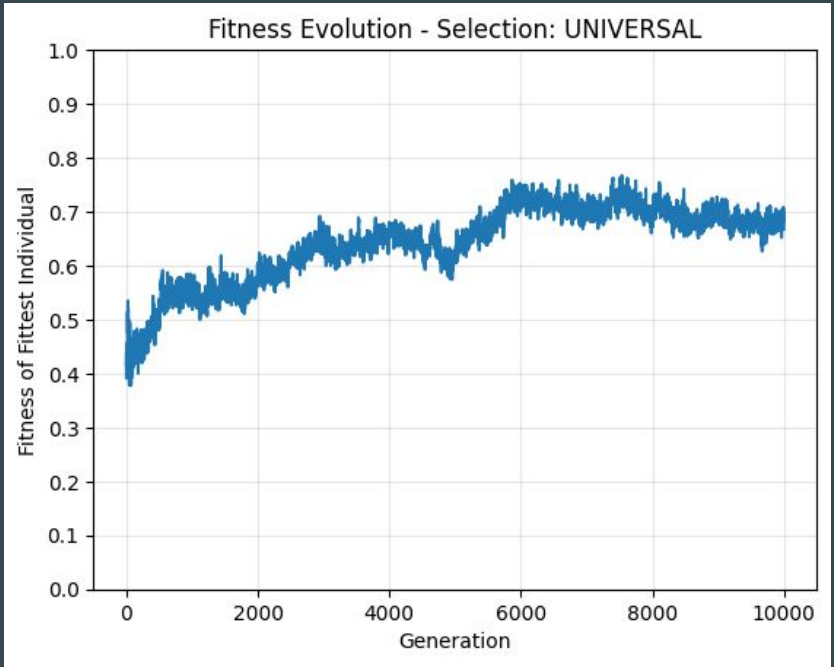
polonia.png

Selección Elite

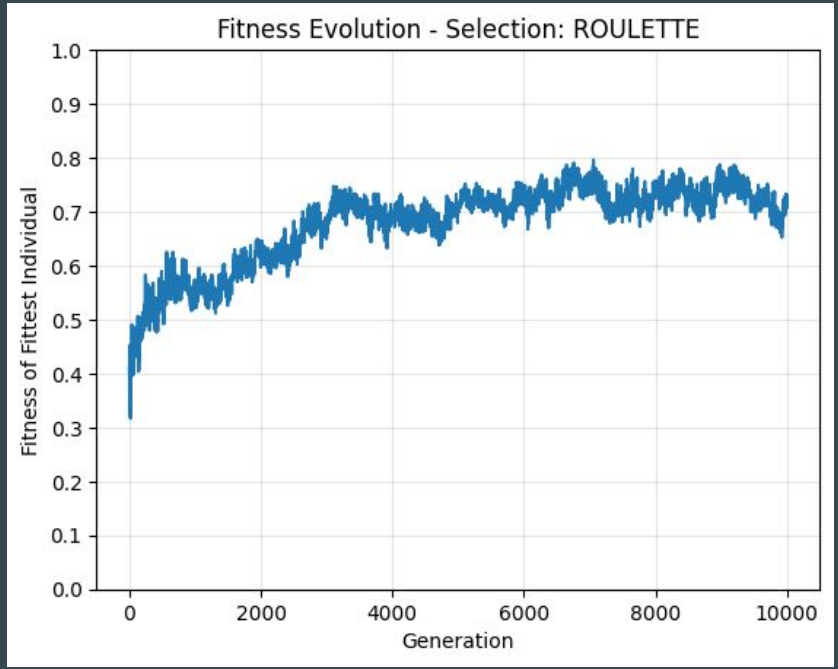




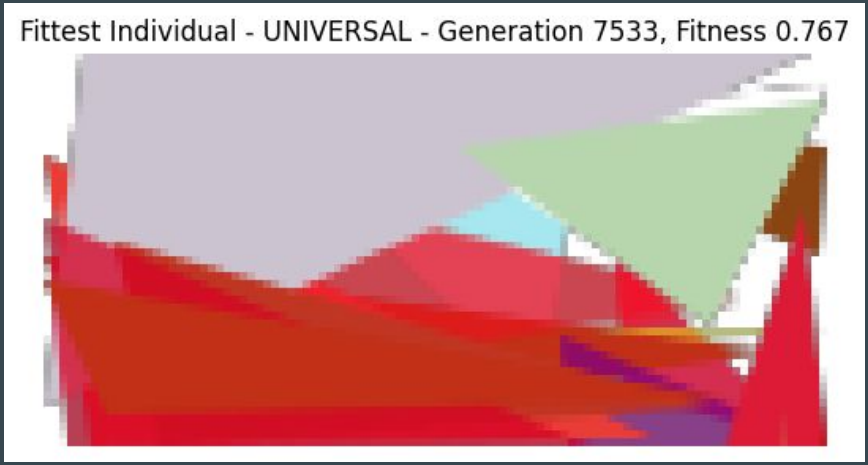
Selección Universal



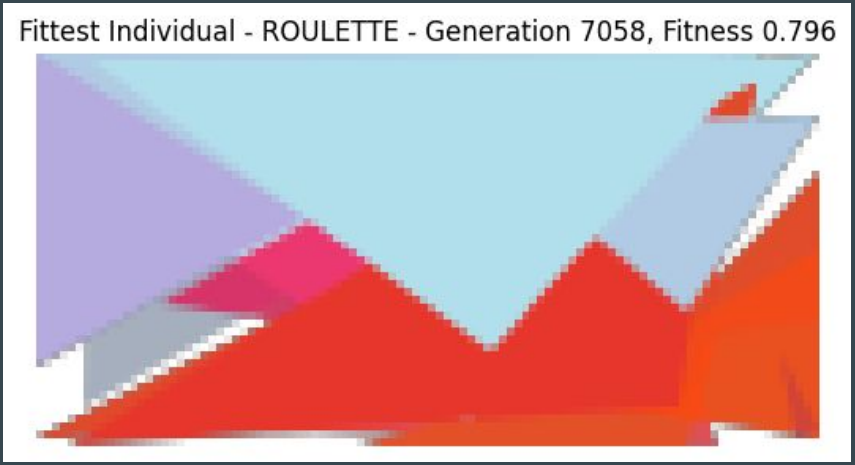
Selección Ruleta



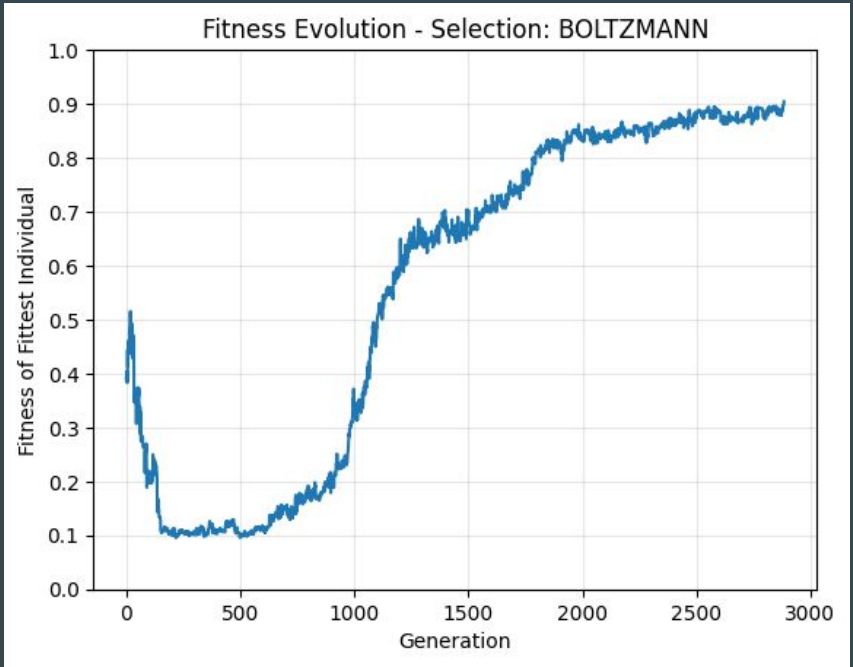
Selección Universal



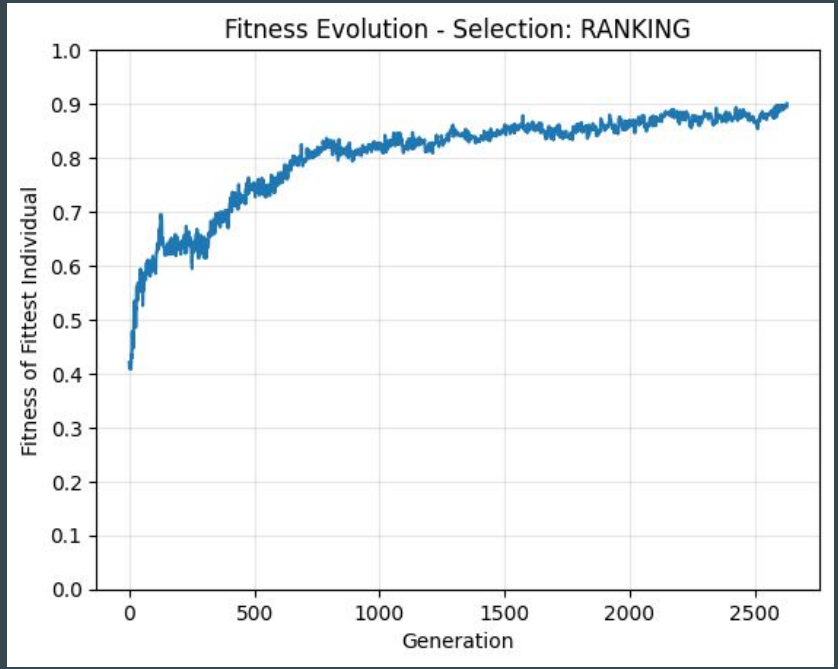
Selección Ruleta



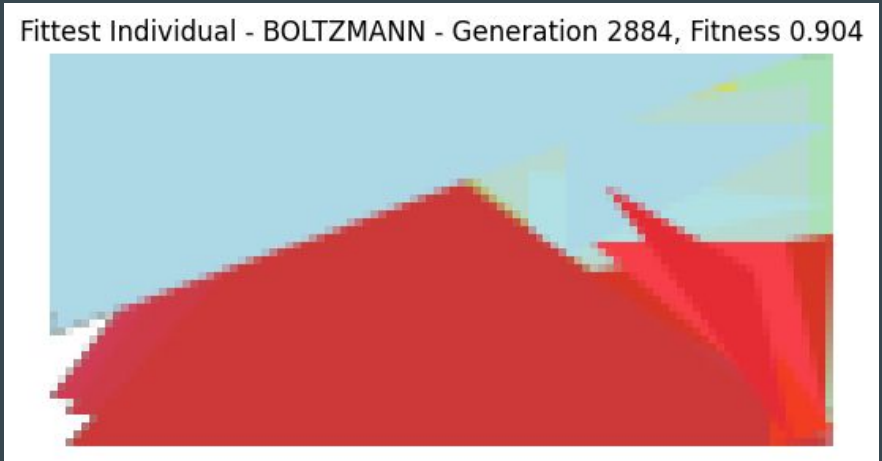
Selección Boltzmann



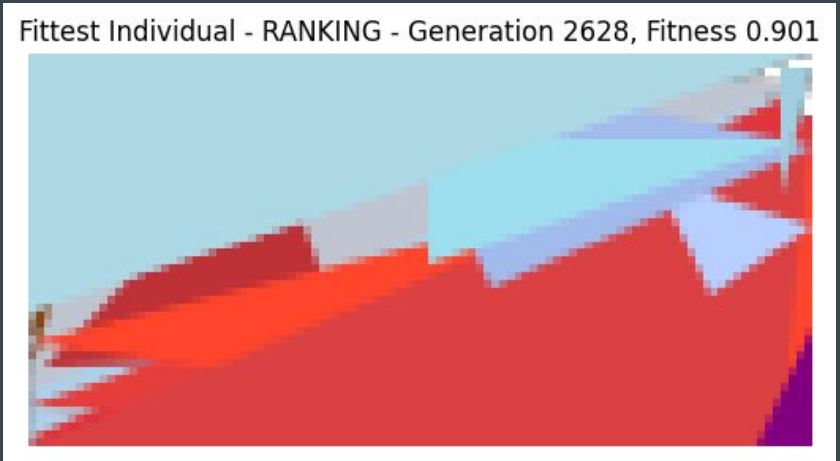
Selección Ranking



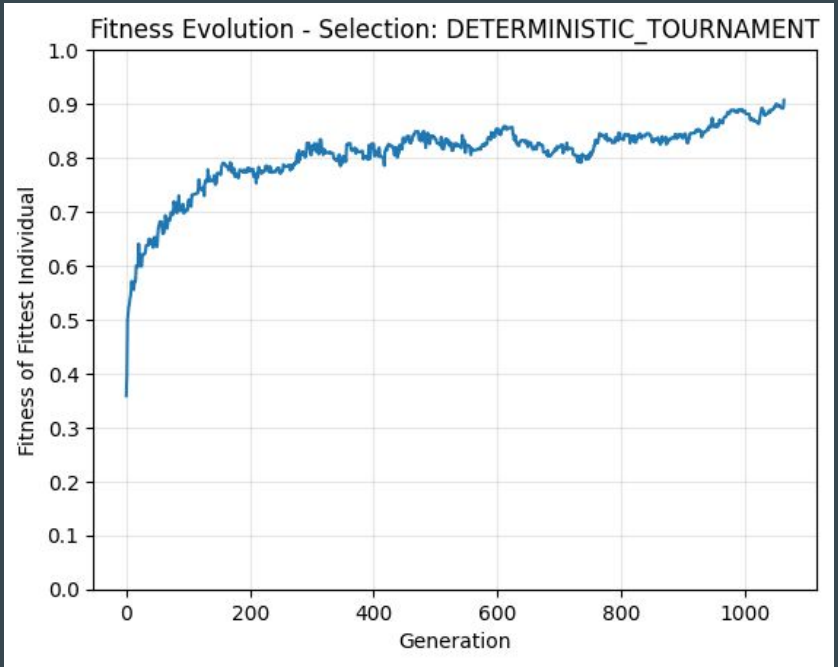
Selección Boltzmann



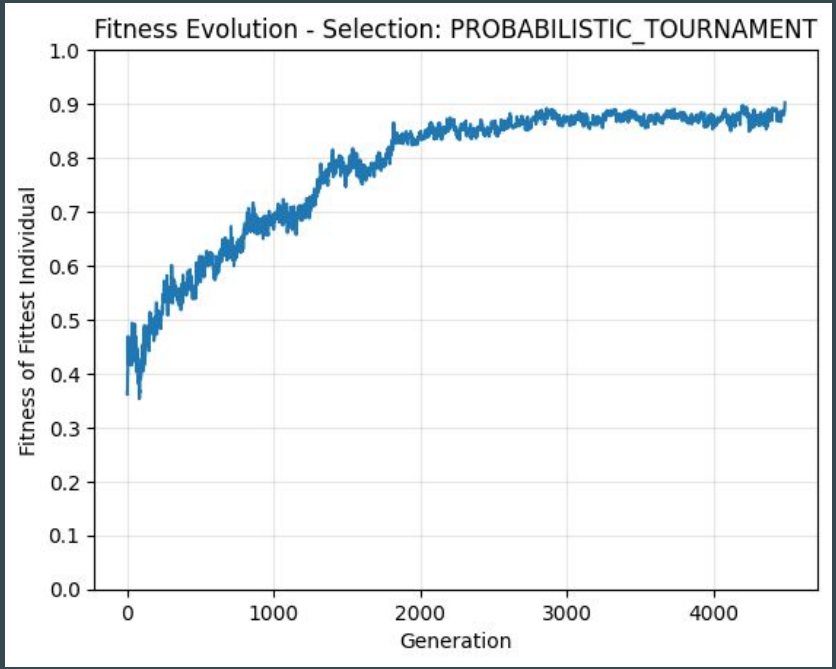
Selección Ranking



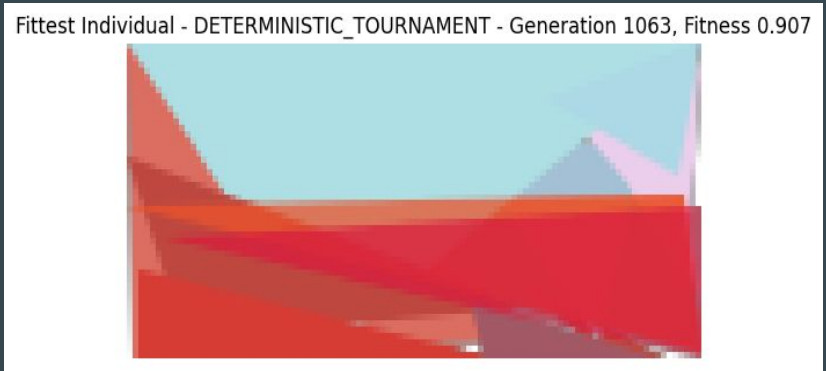
Selección Torn. Determinístico



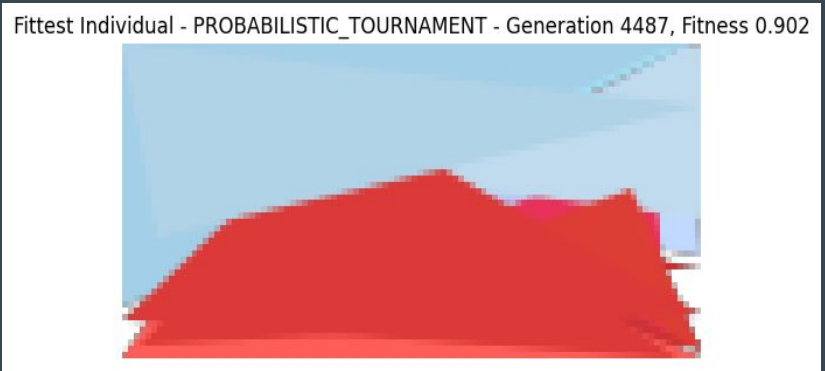
Selección Torn. Probabilístico



Selección Torn. Determinístico



Selección Torn. Probabilístico



Crossover

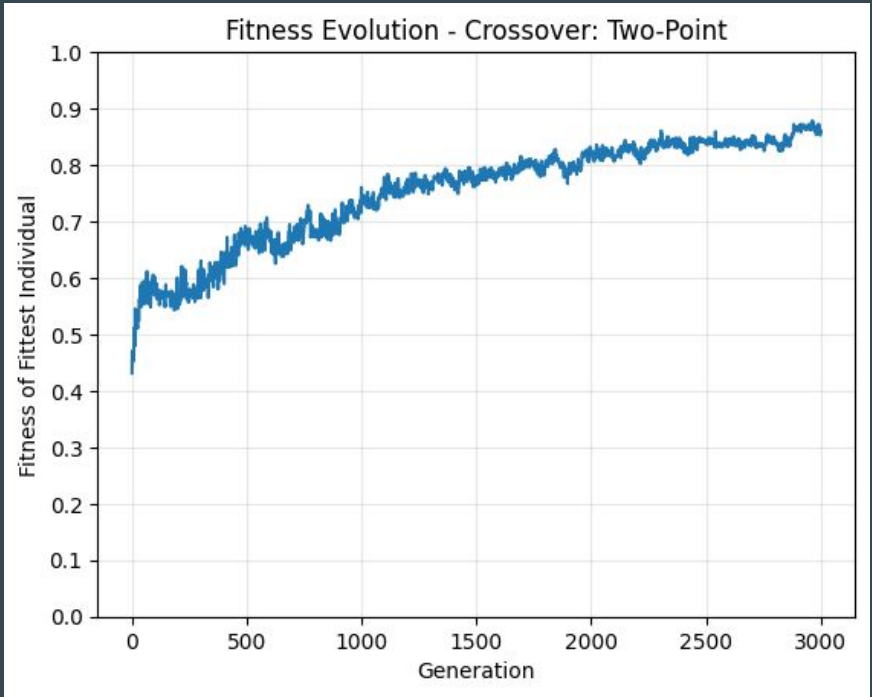
Para nuestros crossover elegimos:

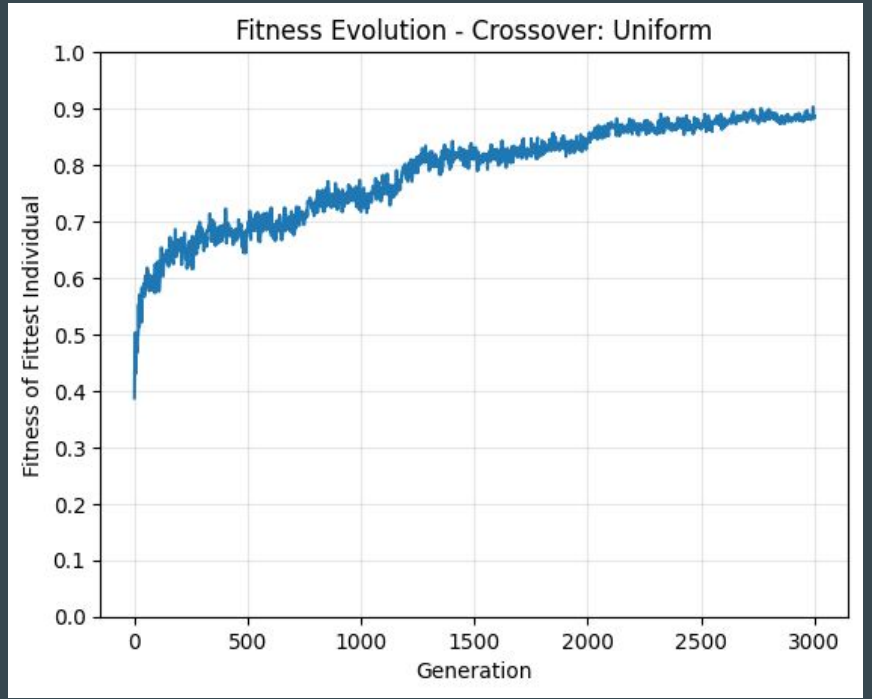
- Cruza en dos puntos
- Cruza Uniforme

Decidimos tomar estos métodos de cruza para garantizar una herencia variada de las figuras en los hijos que generemos, y también tener diversidad en los métodos de cruza pues consideramos que son los dos métodos más distintos de las opciones dadas.

Análisis de distintos métodos de crossover

- polonia-mini.png





Mutación

Para las mutaciones de figuras se aplicó el siguiente criterio:

- Probabilidad de mutación que varía en base a una temperatura, la fórmula es:

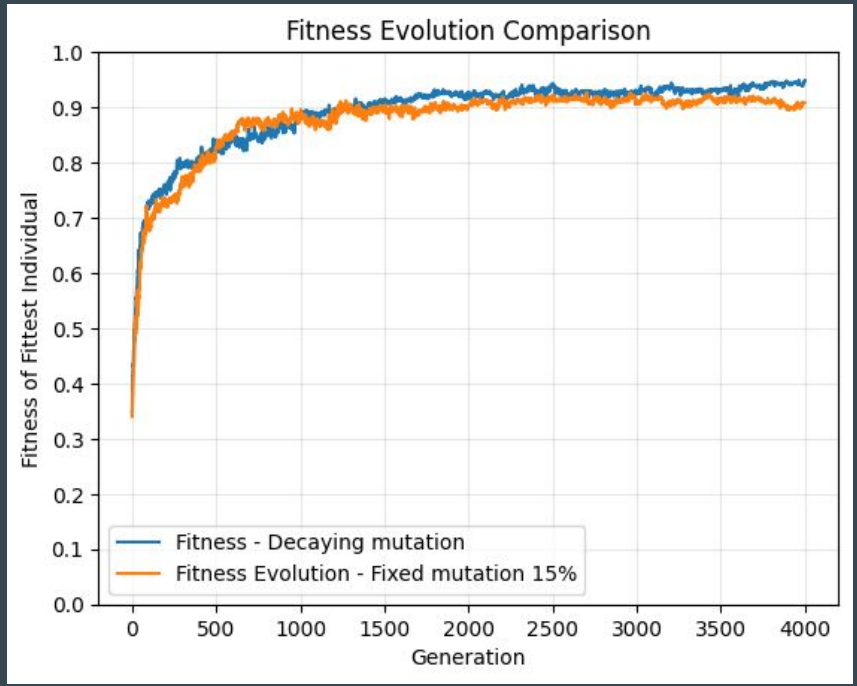
$$T(gen) = T_f + (T_0 - T_c)e^{-k\,gen}$$

- Si un gen (o figura en nuestro caso) **va** a mutar se lanza una ruleta y sucede uno de los siguientes casos:
  - 50% de probabilidad de cambiar su forma física (Se explica en la siguiente diapositiva)
  - 25% de probabilidad de enviar la figura a la capa del fondo
  - 25% de probabilidad de enviar la figura a la capa del frente

Mutación - Temperatura variable

Fue para small-argentina.png

Se utilizó elite, two_point, uniform y young_bias



Mutación (cont.)

En el caso de que mute la forma física de la figura se lanza una ruleta y sucede uno de los siguientes casos:

- 30% de probabilidad de cambiar el color en un delta próximo al actual
- 10% de probabilidad de cambiar el color a un color base (Rojo, Cyan, etc)
- 10% de probabilidad de cambiar la transparencia a 100%, 75% o 50%
- 30% de probabilidad de cambiar la posición de cada vértice en un delta
- 20% de probabilidad de achicar el triángulo y cambiar su posición

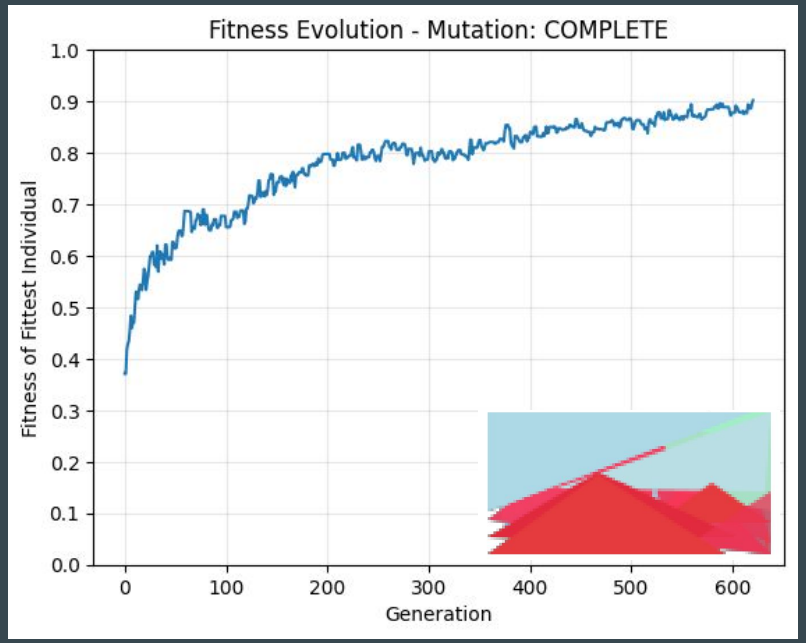
Mutación (cont.)

Se optó por este tipo de mutación estilo “ruleta” para que cuando varíe un gen, no cambie completamente. Favoreciendo que cuando una figura está “cerca” de una condición favorable, si llega a mutar pueda cambiar a dicha condición en lugar de variar completamente el gen.

Utilizamos los siguientes algoritmos:

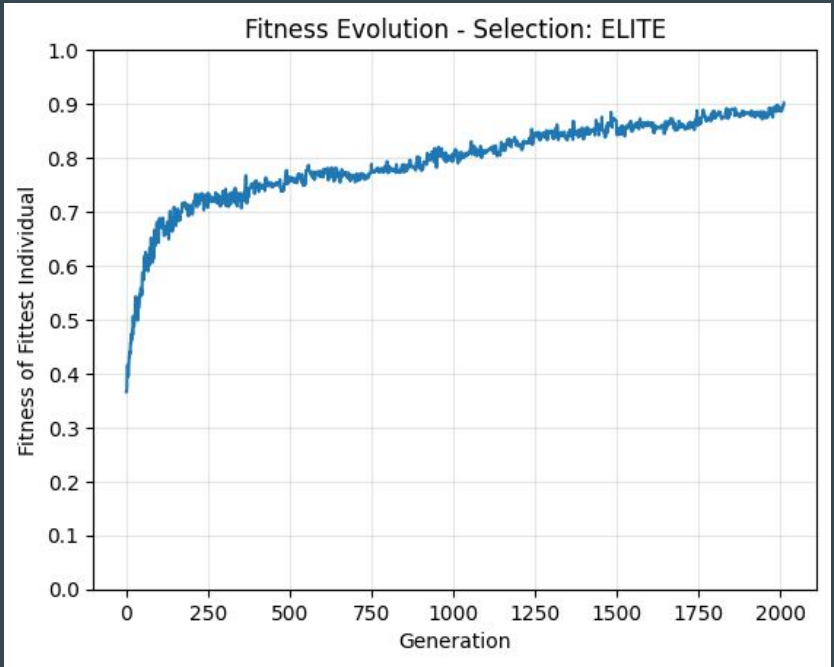
- Mutación Uniforme
- Mutación Completa

Análisis de distintos métodos de mutación

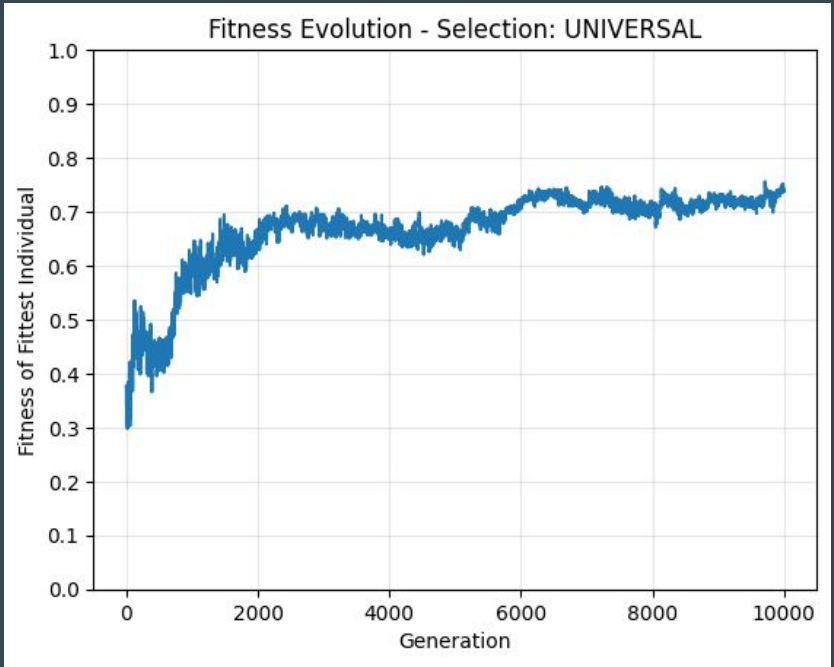


small-argentina.png

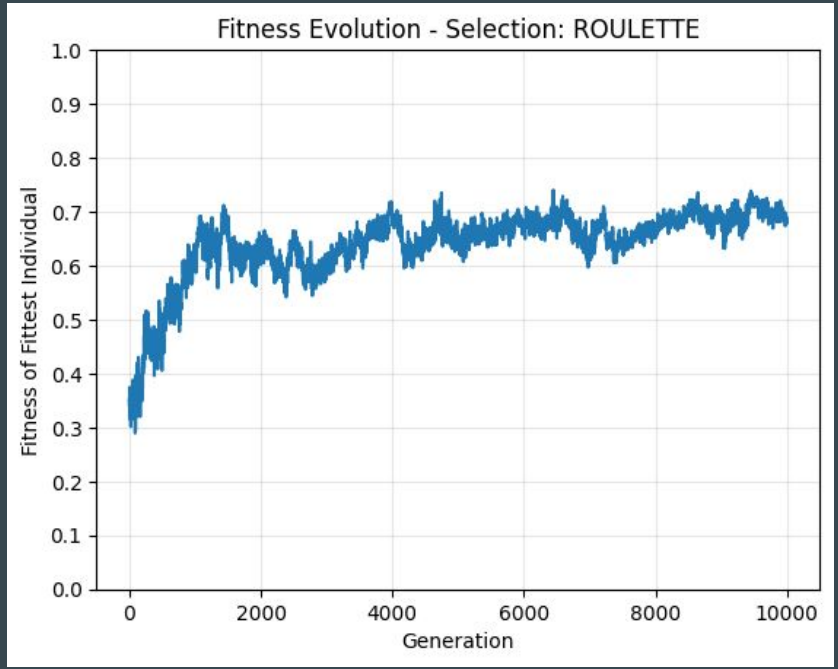
Selección Elite



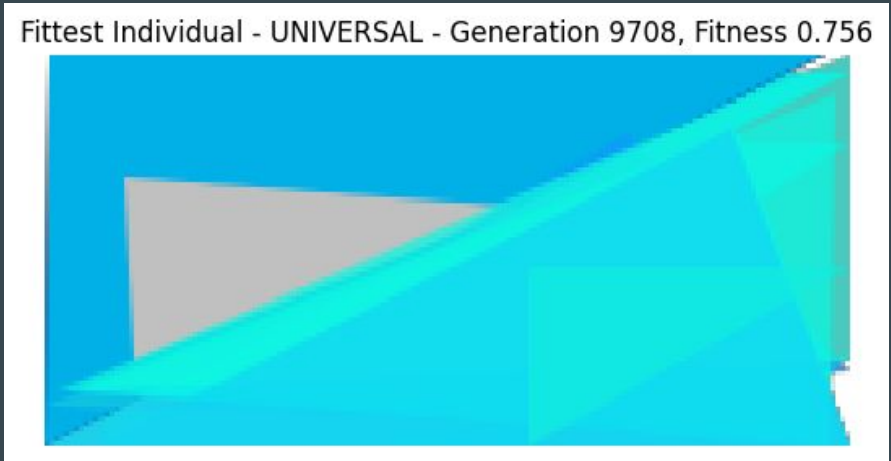
Selección Universal



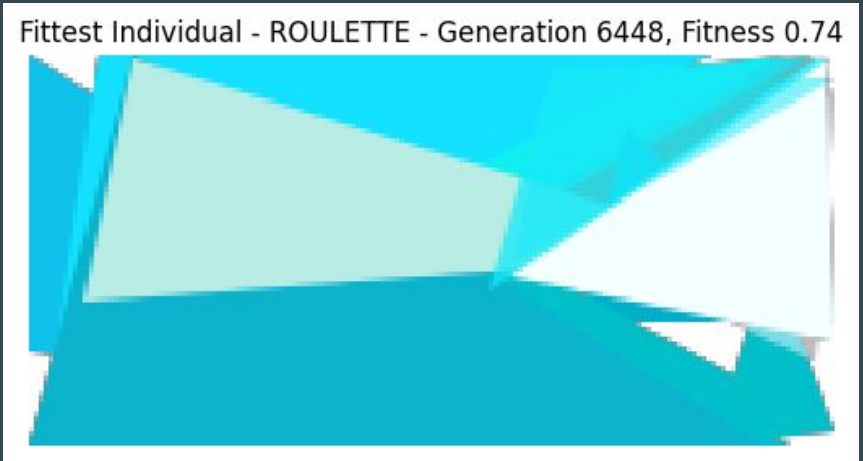
Selección Ruleta



Selección Universal

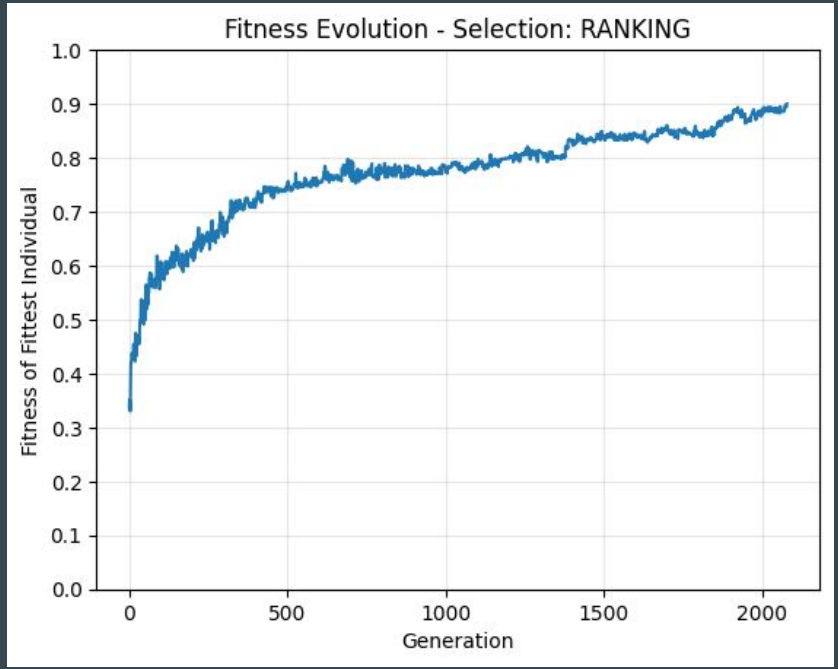


Selección Ruleta



Selección Boltzmann

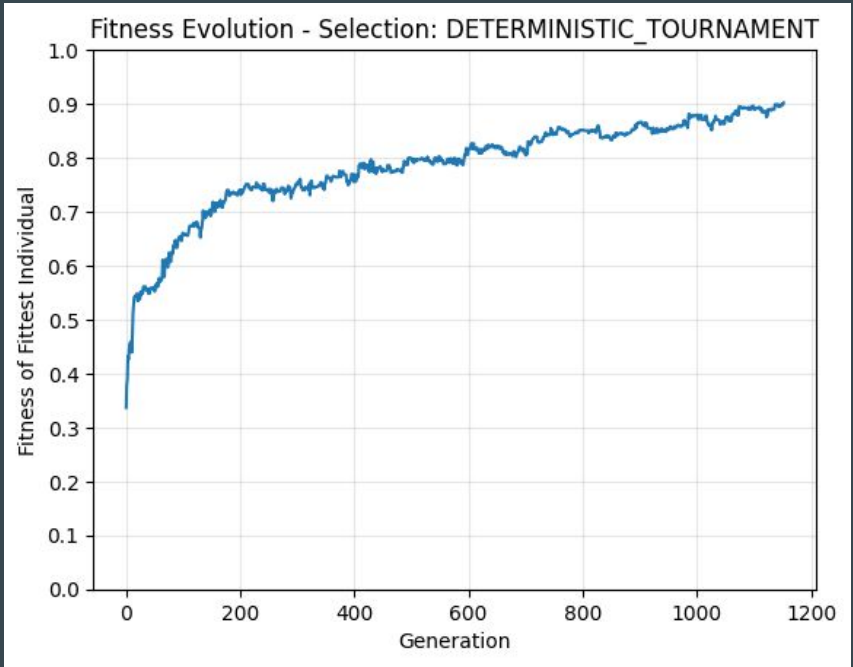
Selección Ranking



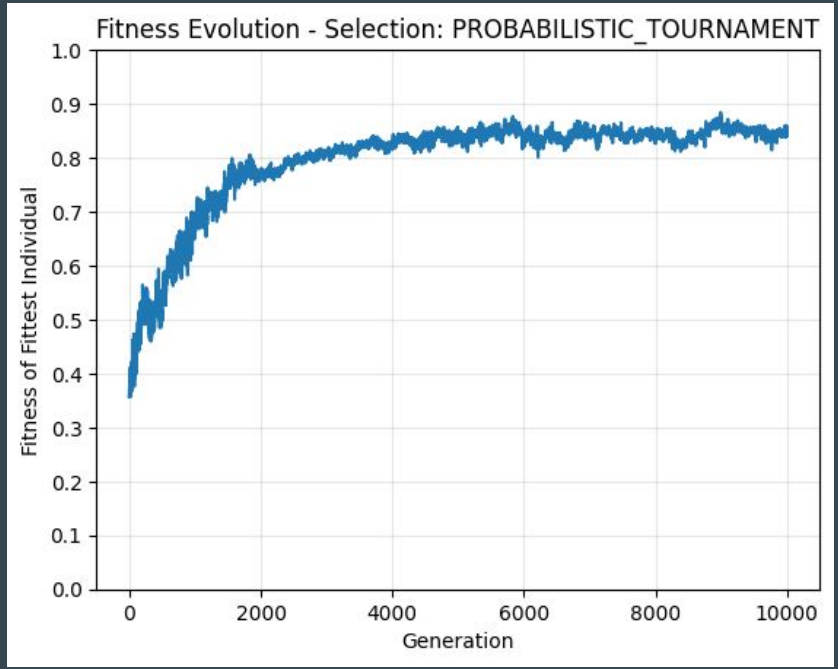
Selección Boltzmann

Selección Ranking

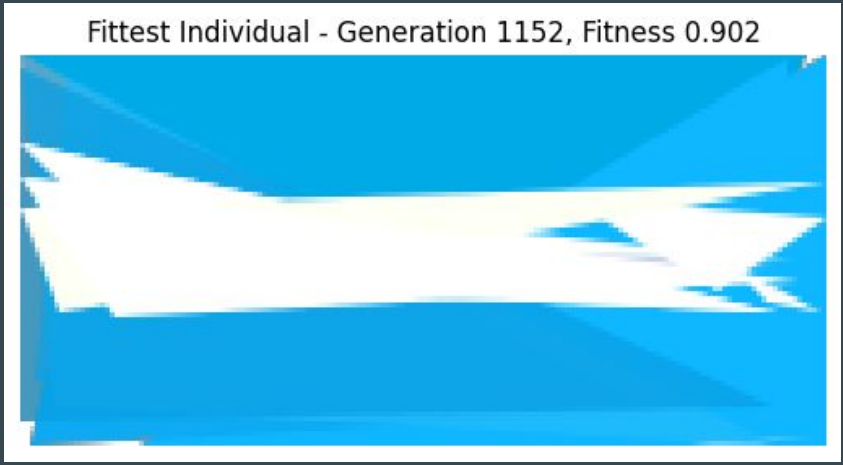
Selección Torn. Determinístico



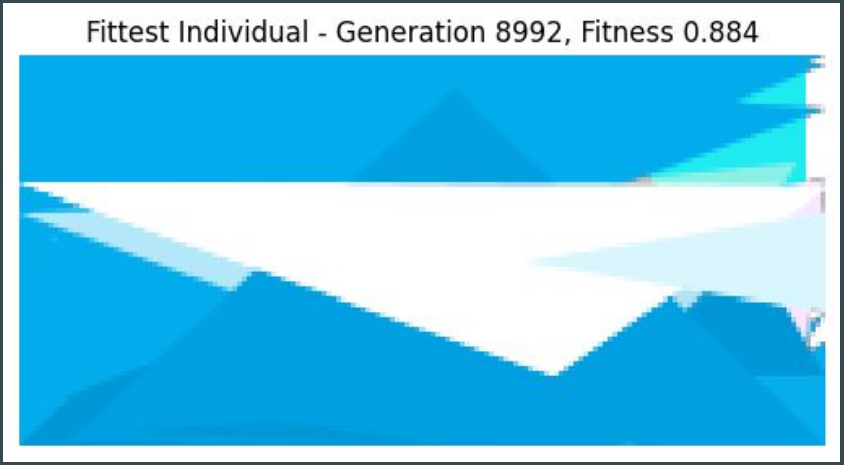
Selección Torn. Probabilístico



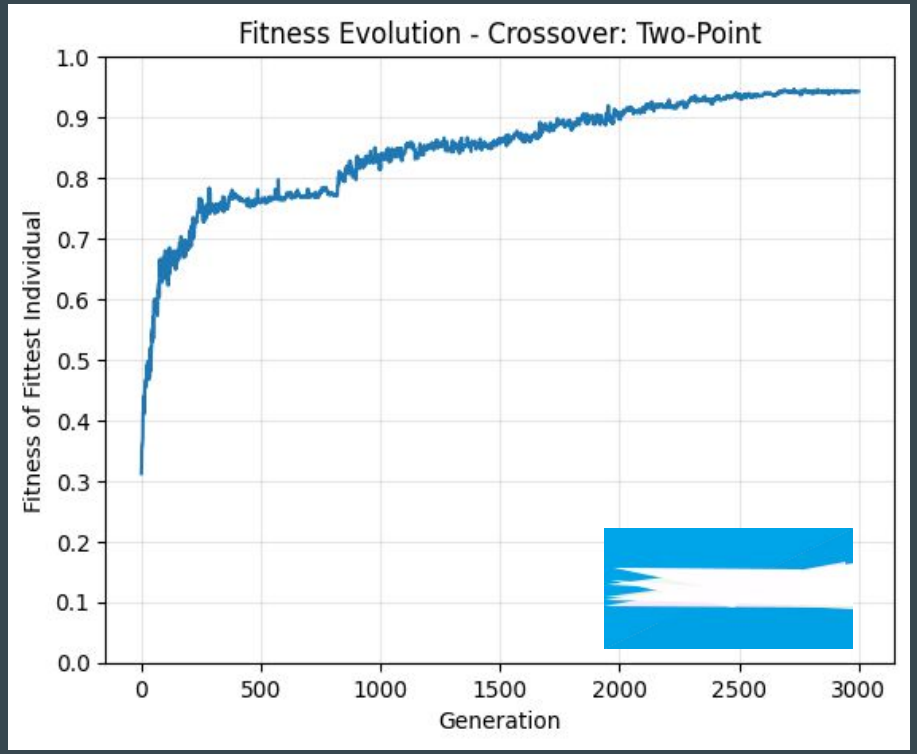
Selección Torn. Determinístico

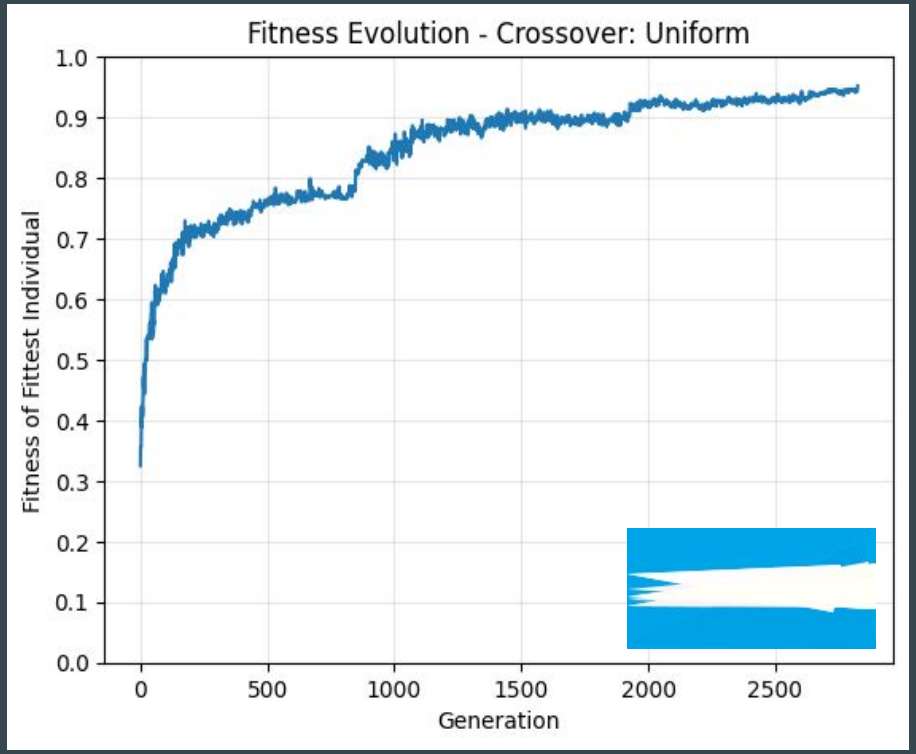


Selección Torn. Probabilístico

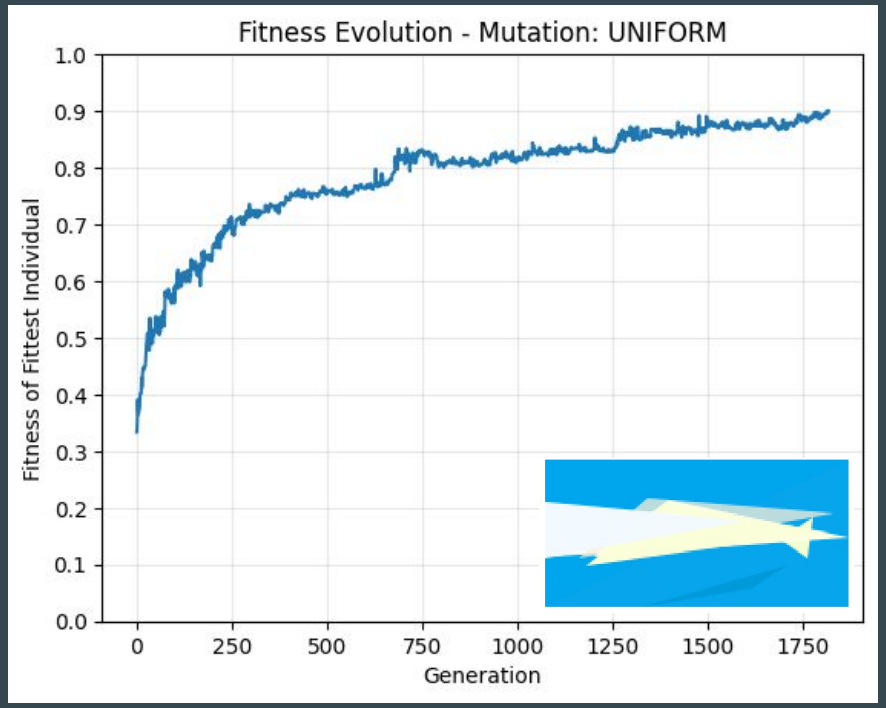


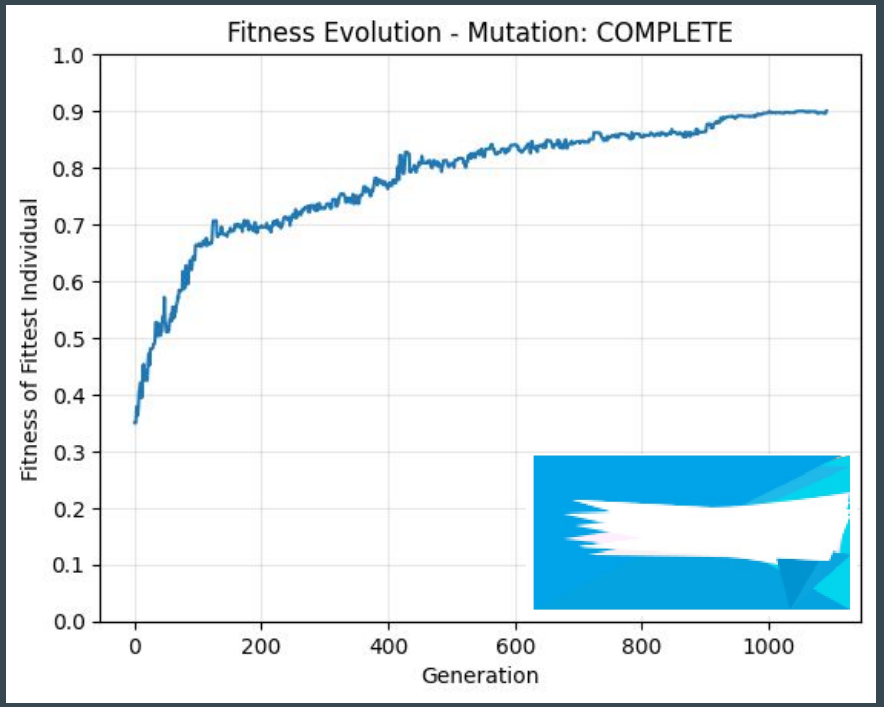
Análisis de distintos métodos de crossover





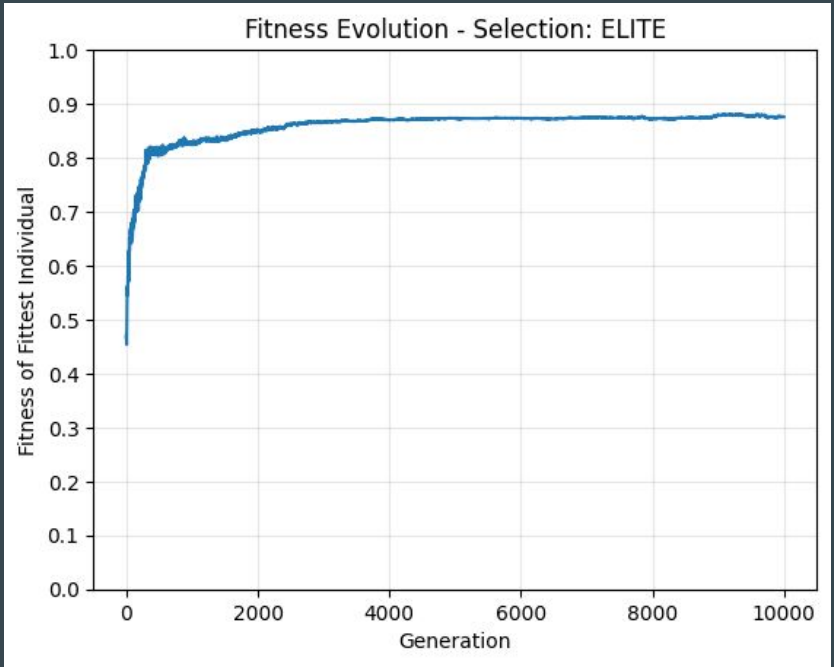
Análisis de distintos métodos de mutación



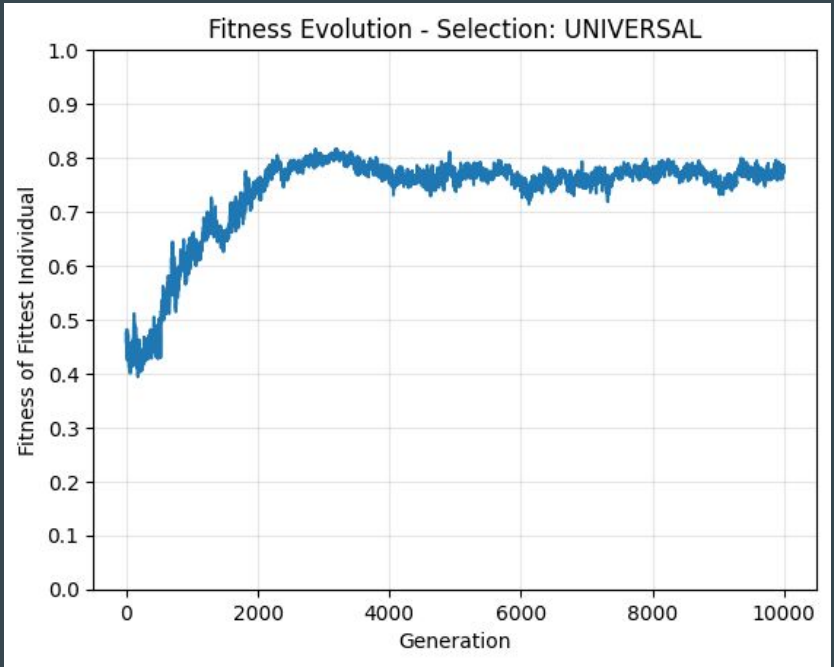


starry-night-small.png

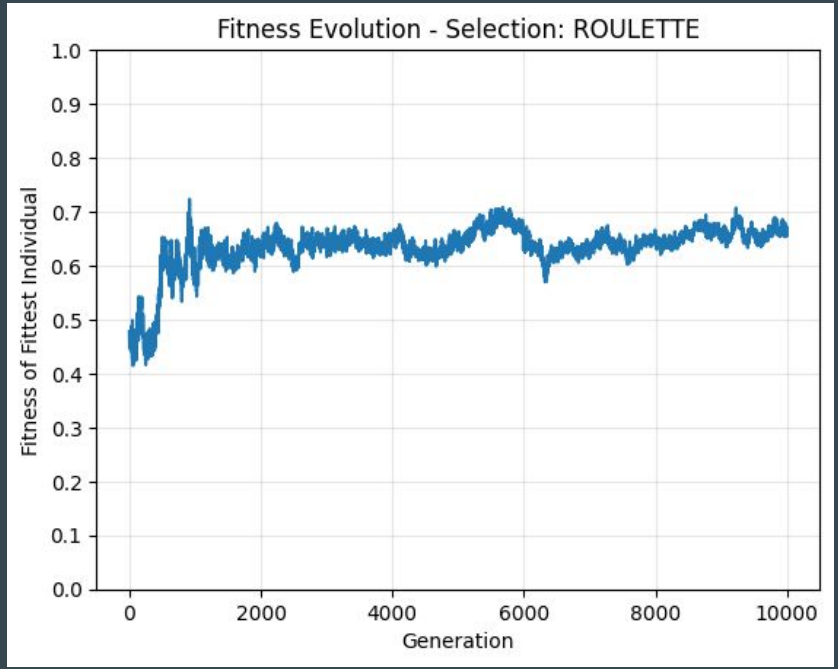
Selección Elite



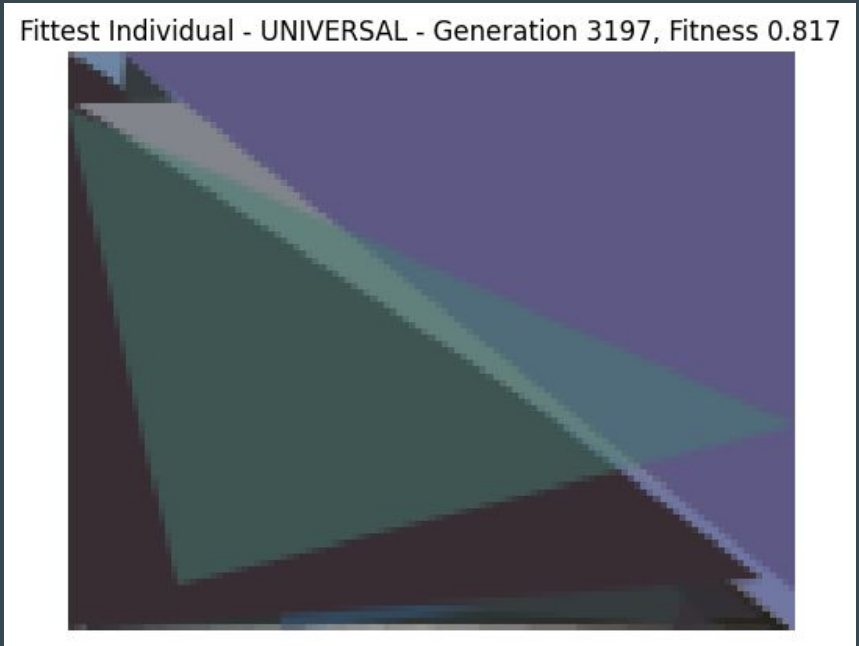
Selección Universal



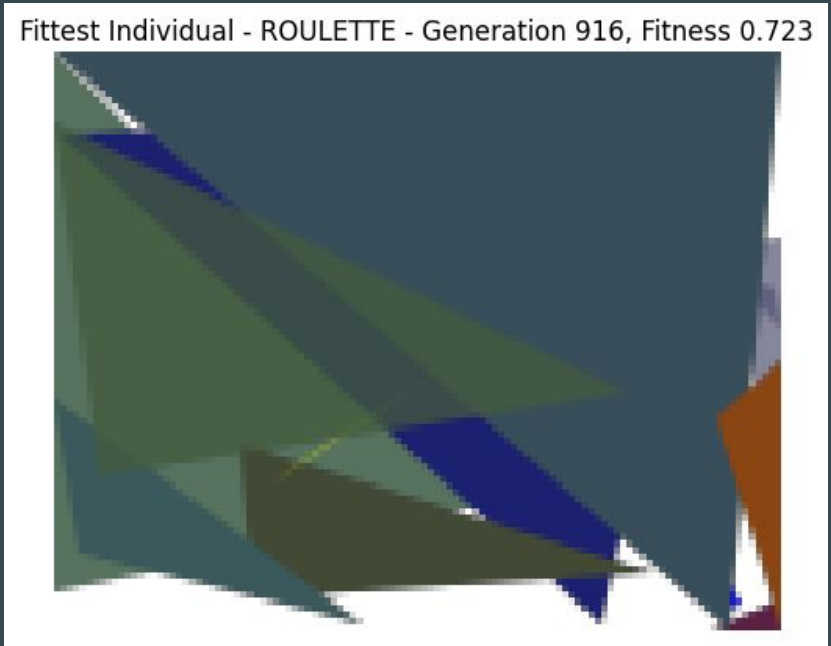
Selección Ruleta



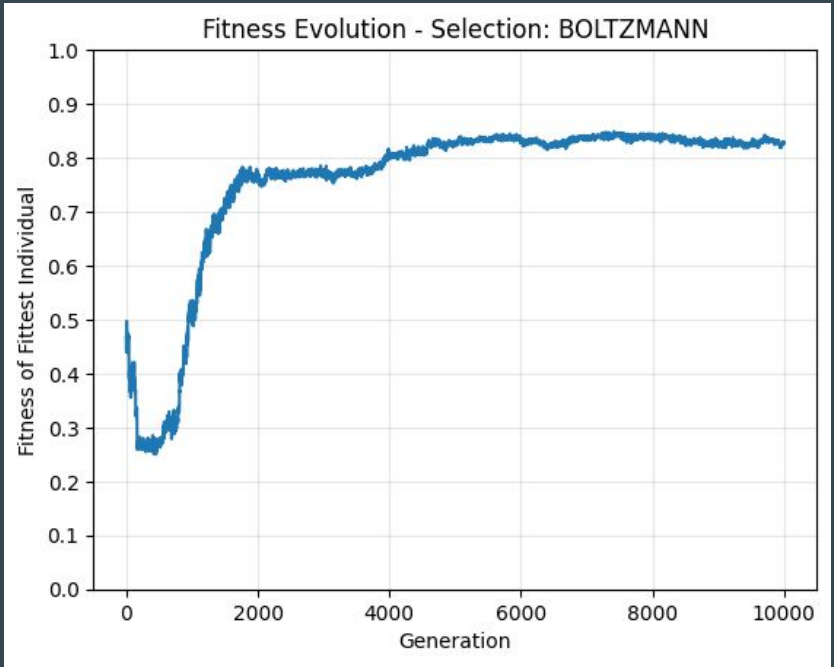
Selección Universal



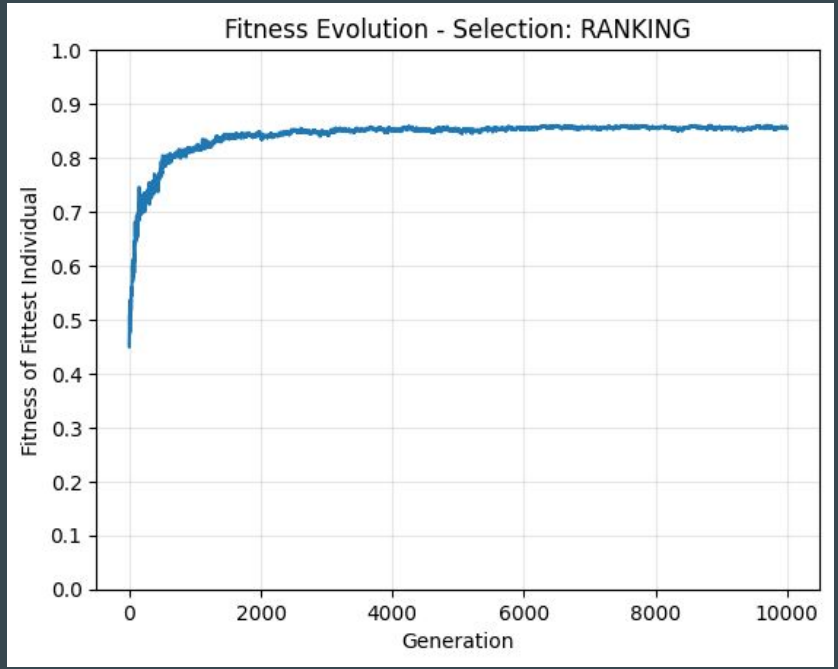
Selección Ruleta



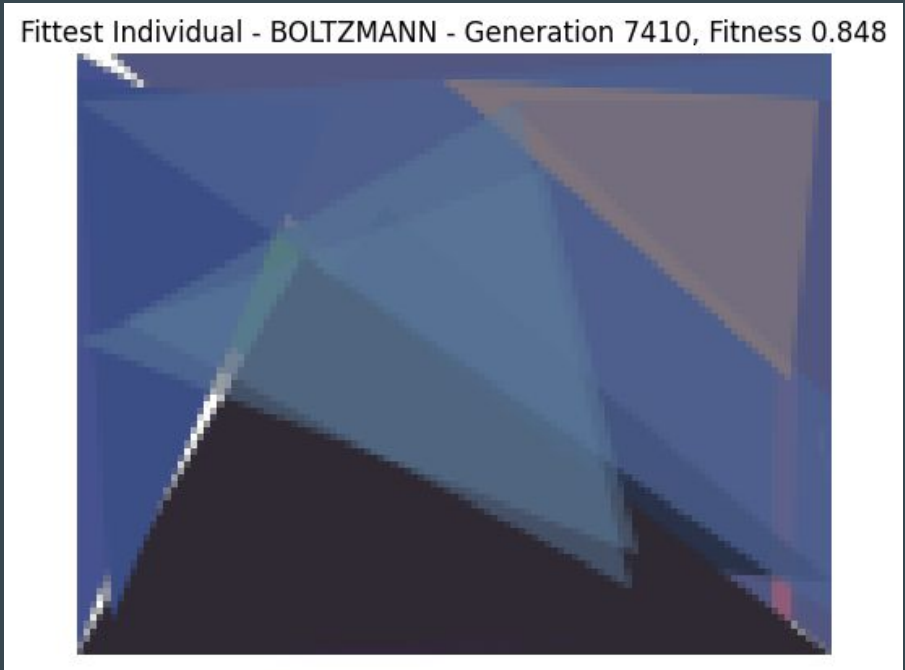
Selección Boltzmann



Selección Ranking

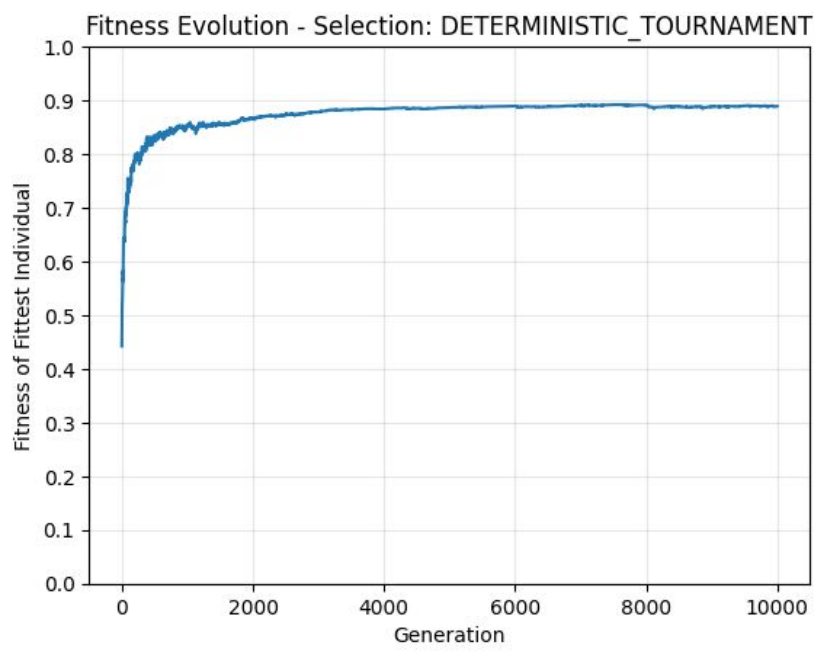


Selección Boltzmann

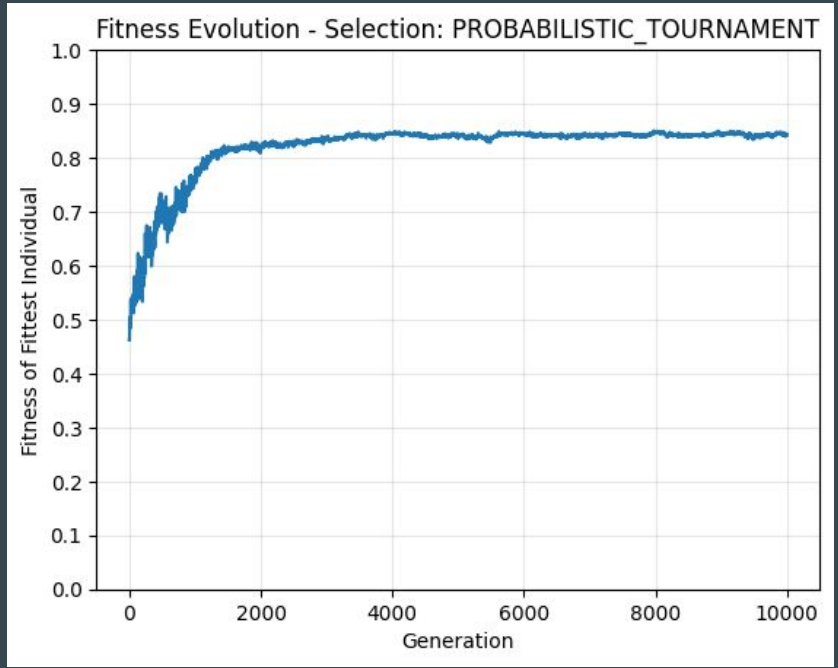


Selección Ranking

Selección Torn. Determinístico

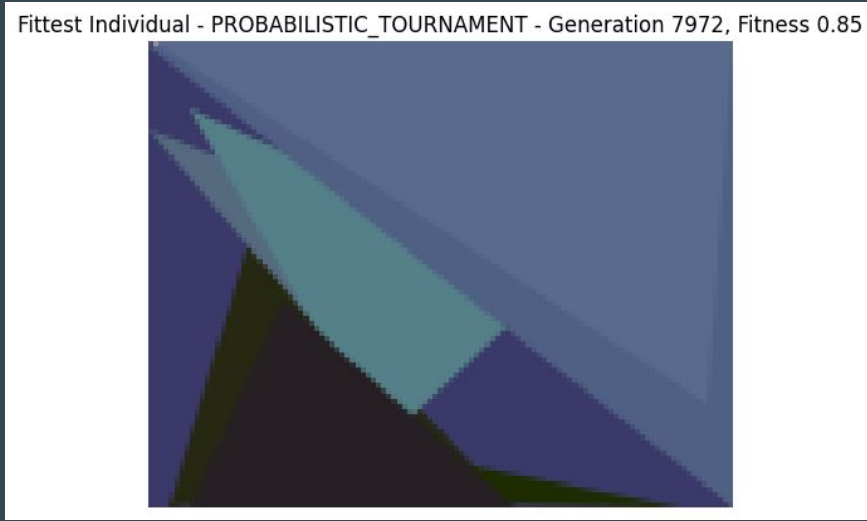


Selección Torn. Probabilístico

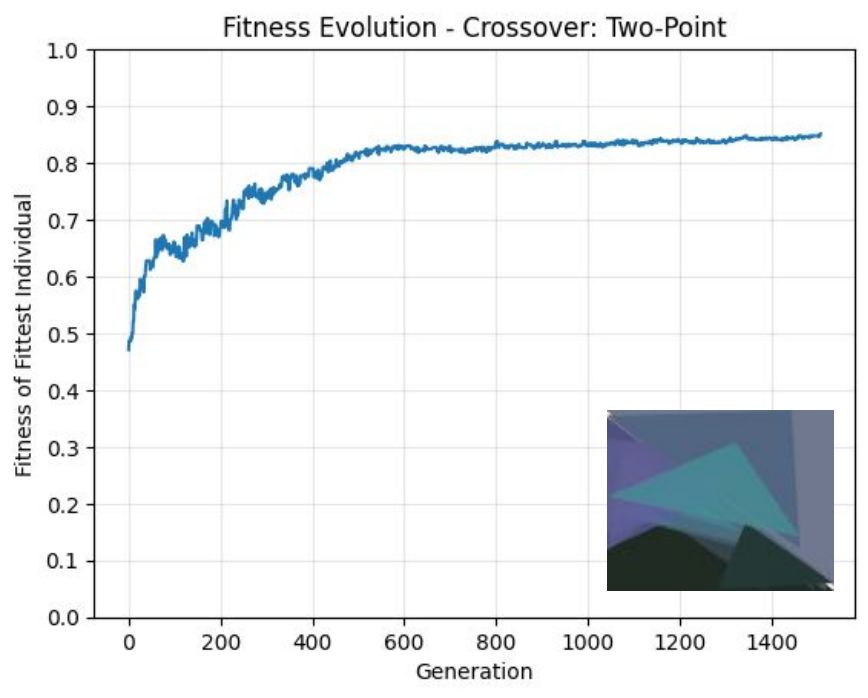


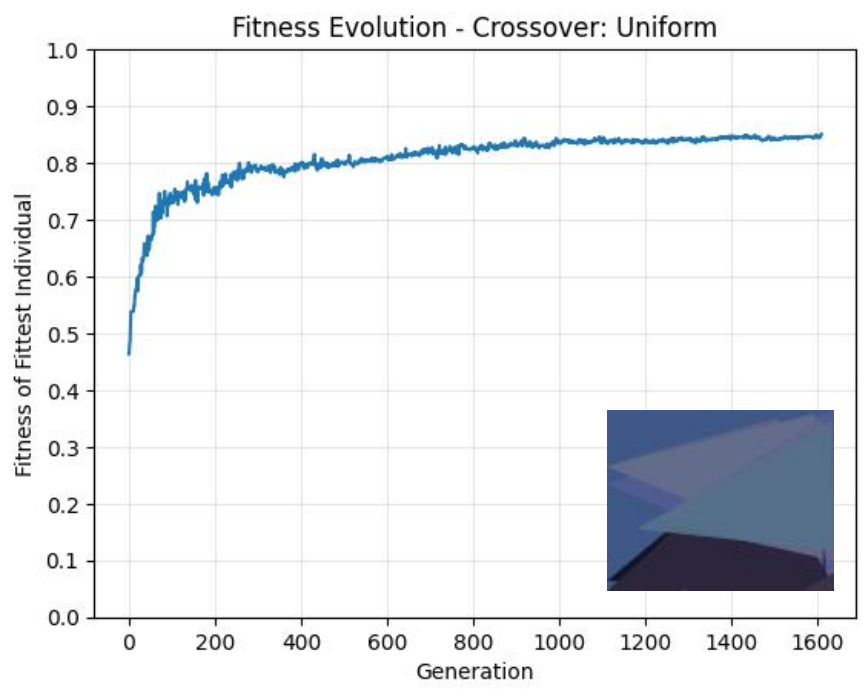
Selección Torn. Determinístico

Selección Torn. Probabilístico

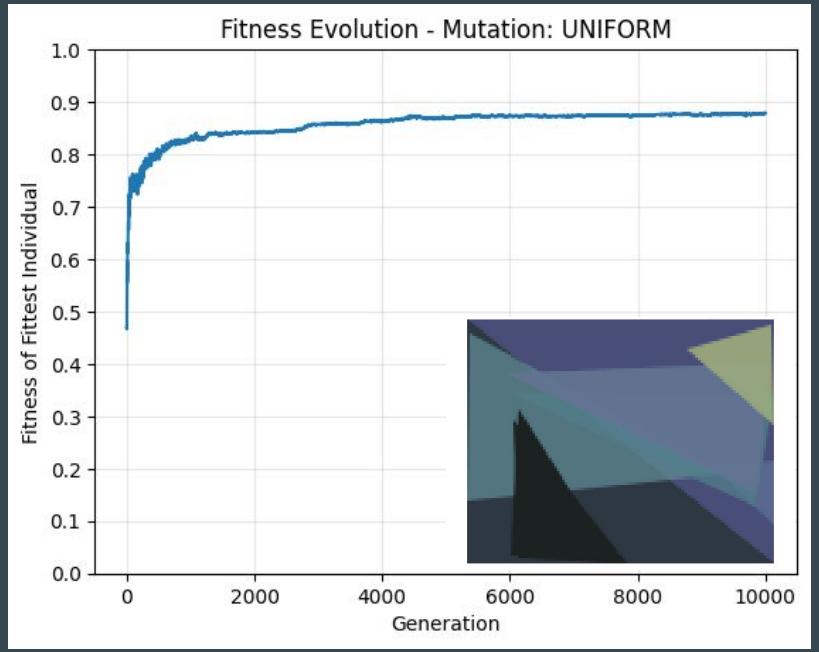


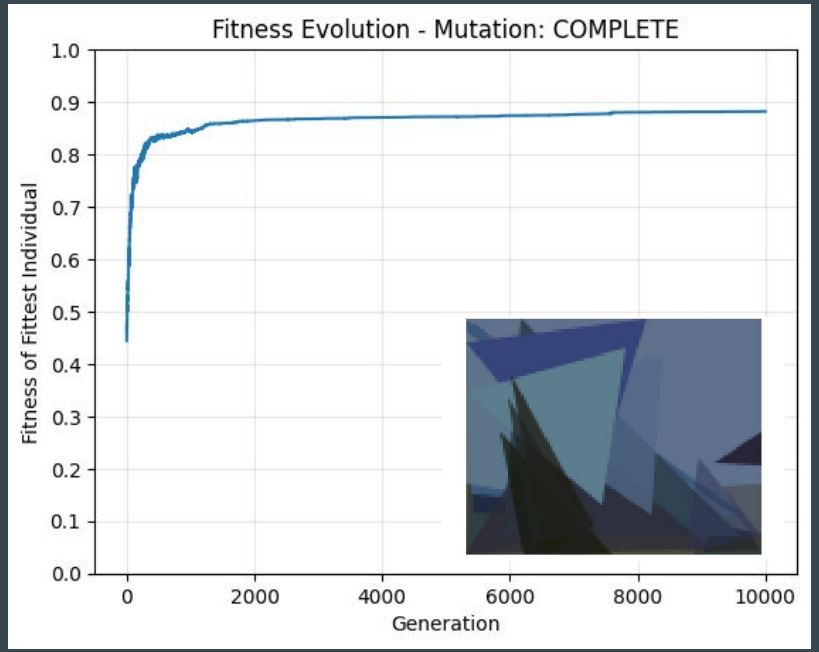
Análisis de distintos métodos de crossover





Análisis de distintos métodos de mutación





Conclusiones - Selecciones azarosos vs Selecciones no azarosas

Mirando los resultados obtenidos, se puede hacer una diferenciación entre los algoritmos de selección azarosos y no azarosas:

- Los algoritmos azarosos otorgan una probabilidad a los distintos individuos en base a su fitness, estos se puede observar que en general no logran una solución lo suficientemente aceptable por su alta aleatoriedad, se pueden alejar de los mejores fitness. Ejemplos son Ruleta y Universal
- Los algoritmos no azarosas priorizan seleccionar a los individuos con mayor fitness y no se guían por el azar, esto se puede observar debido a que logran una mejor solución en menos generaciones. Ejemplos son Elite, Ranking, Boltzmann, Torneo Determinístico y Torneo Probabilístico

Conclusiones - Crossover y Mutaciones

- Con los resultados obtenidos se puede observar que el crossover uniforme converge ligeramente más rápido a una solución aceptable que el crossover en dos puntos.

- Por otro lado, en los distintos métodos de mutación (uniforme y completa) no se logra notar una diferencia apreciable.