



SIA - TP2

Algoritmos Genéticos

Grupo 7

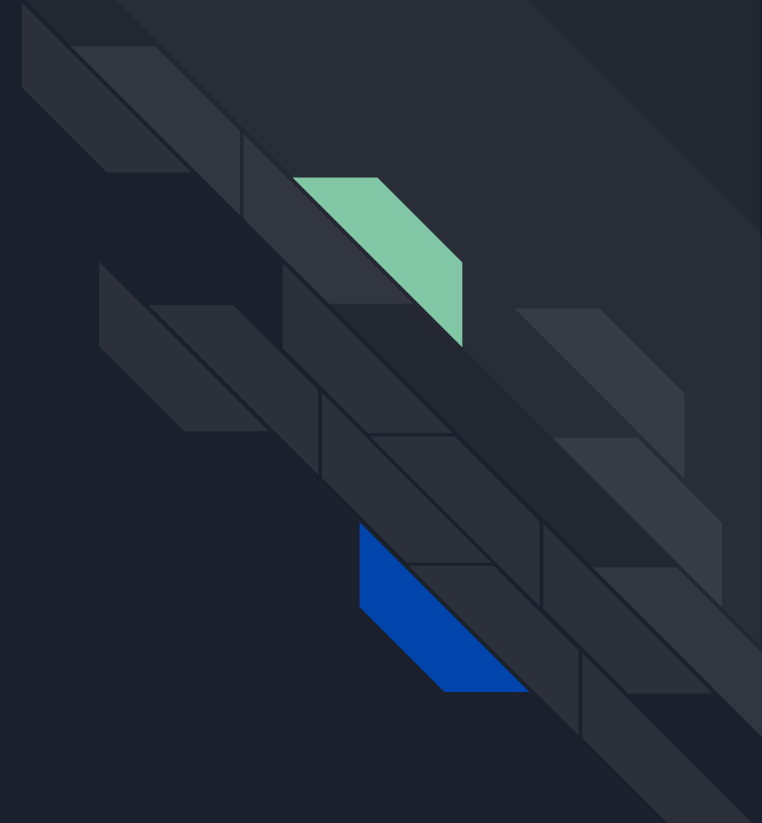
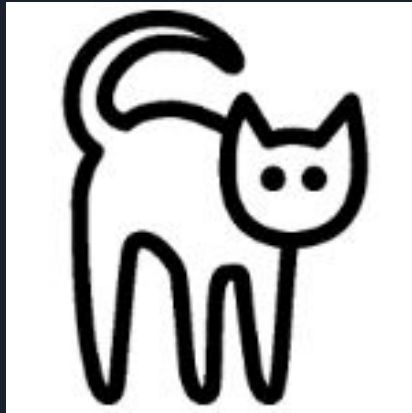
Tomás Scheffer - 63393

Matías Sapino - 61067

Tobías Pugliano - 62180

Luca Bloise - 63004

Ejercicio 1





Estructura

- **Individuo:** Matriz $N \times N$ de caracteres ASCII. Cada carácter se corresponde con una sección de la imagen original.
- **Genes:** Cada gen representa un caracter ASCII de la matriz, teniendo entonces cada individuo N^2 genes.
- **Generación 0:** Se obtienen individuos con caracteres ASCII aleatorios.



Función de Fitness

- A cada carácter ASCII de un individuo le asociamos una densidad entre 0 y 1 (" tiene densidad 0, '#' tiene densidad 1).
- Para cada sección de la imagen original se calcula su densidad en base al nivel de oscuridad (0 si es todo blanco, 1 si es todo negro).
- Se comparan las densidades entre cada carácter ASCII y cada sección de la imagen y se calcula el error.
- Para cada individuo, se calcula el fitness de la siguiente manera:
 - $f(i) = 1/(1+\text{suma errores})$



Función de Cruza

Cruce Uniforme

- No mantiene la correlación posicional de los alelos.
- Aumenta la variedad.



Criterios de Corte

- Cantidad de generaciones.
- Estructura: Una parte relevante de la población no cambia en una cantidad de generaciones.
- Contenido: El mejor fitness no cambia en una cantidad de generaciones.

Ejercicio 2





Estructura

- **Individuo:** Aproximación de la imagen objetivo.
- **Genes:** Triángulos de colores.
- **Generación 0:** Se generan individuos con triángulos generados aleatoriamente (vértices y colores aleatorios).



Función de Fitness

- Se compara al individuo con la imagen objetivo.
- Para la comparación, se utiliza el espacio de color CIELAB el cual se basa en la percepción del ojo humano para comparar colores.
- El Fitness toma el error obtenido a partir de la diferencia de los valores CIELAB.

$$\text{fitness} = \exp \left(-\frac{\text{error}}{\text{escala}} \right)$$



Métodos de selección implementados

- Elite
- Ruleta
- Universal
- Boltzmann
- Torneo determinístico
- Torneo probabilístico
- Ranking




Métodos de cruce implementados

- Cruce de un punto
 - Cruce de dos puntos
 - Cruce uniforme
 - Cruce anular
-
- A partir de K padres se generan K hijos.
 - Dentro de los K padres escogidos, se los aparean de forma aleatoria.
 - Por cada par de padres se generan 2 hijos.
 - Se cruzan los genes, en este caso los triángulos.



Métodos de mutación implementados

- **Gen**
 - **MultiGen**
 - **Uniforme**
 - **Completa**
-
- Se mutan los genes del individuo, en este caso sus triángulos.
 - Al mutar un triángulo se muta su color, uno de sus vértices o ambas características.
 - Al mutar un vértice, se lo reemplaza por uno nuevo generado de forma aleatoria.
 - Al mutar el color, se modifica ligeramente el color actual mediante un delta (valor aleatorio entre -10 y 10).



Métodos para elegir nuevas generaciones implementados

- Tradicional
- Sesgo Joven

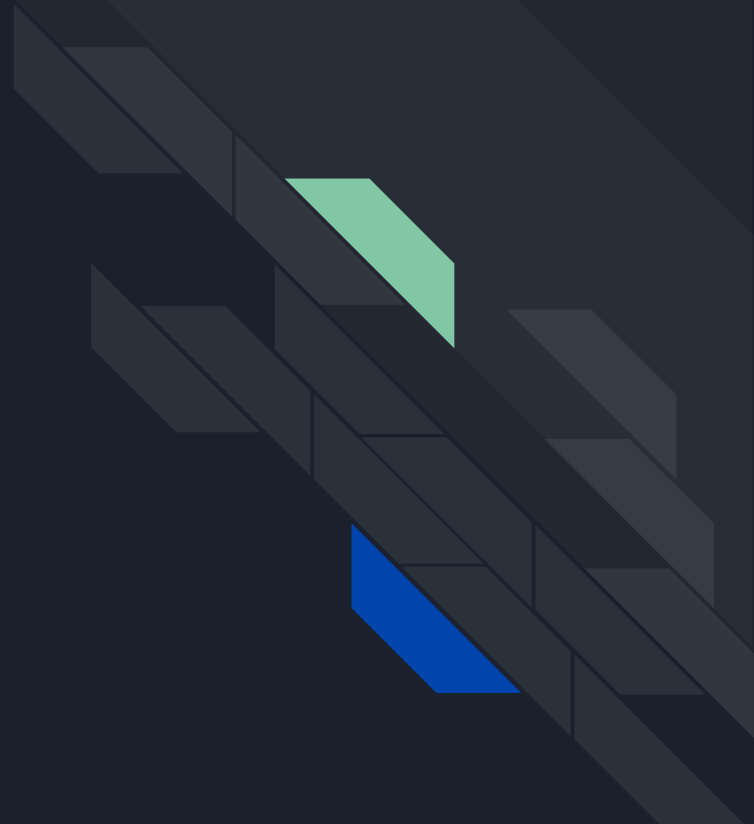


Condiciones de corte implementadas

- Tiempo
- Cantidad de generaciones
- Solución aceptable
- Estructura
- Contenido

Resultados

Para todos los resultados se utilizaron 200 triángulos para cada individuo

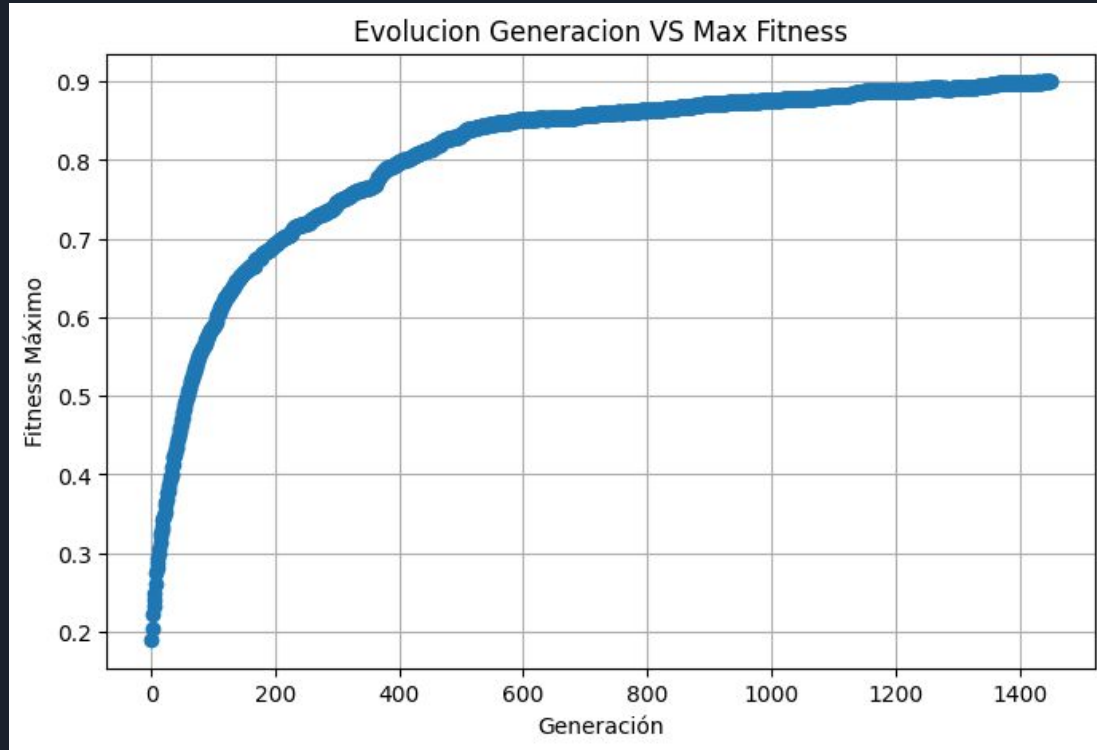


Probando el algoritmo

```
{  
  "n_population_size": 500,  
  
  "selection_method": "deterministic_tournaments",  
  "k_selection_size": 600,  
  "m_selection_size": 100,  
  "threshold": 0.75,  
  
  "crossover_method": "uniform",  
  "crossover_probability": 1,  
  "p_uniform": 0.9,  
  
  "mutation_method": "multigen_uniform",  
  "mutation_probability": 0.1,  
  "mutation_M": 5,  
  
  "implementation": "young-bias",  
  "stop_condition": "acceptable_solution",  
  "stop_condition_acceptable_solution": 0.90  
}
```



Probando el algoritmo



Evolución del individuo con máximo fitness





Modificando hiperparametros





Tamaño de población

```
{
  "n_population_size": 10,

  "selection_method": "elite",
  "k_selection_size": 2,

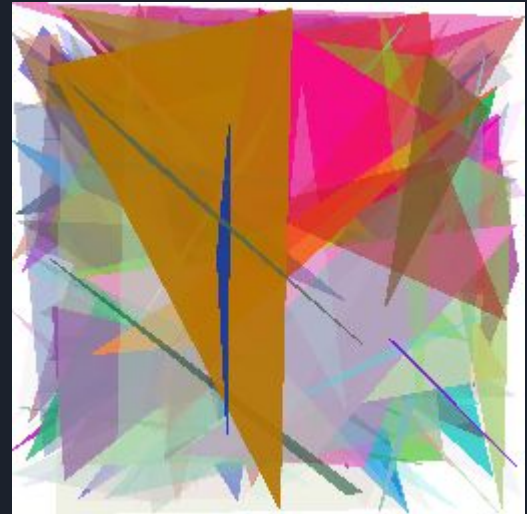
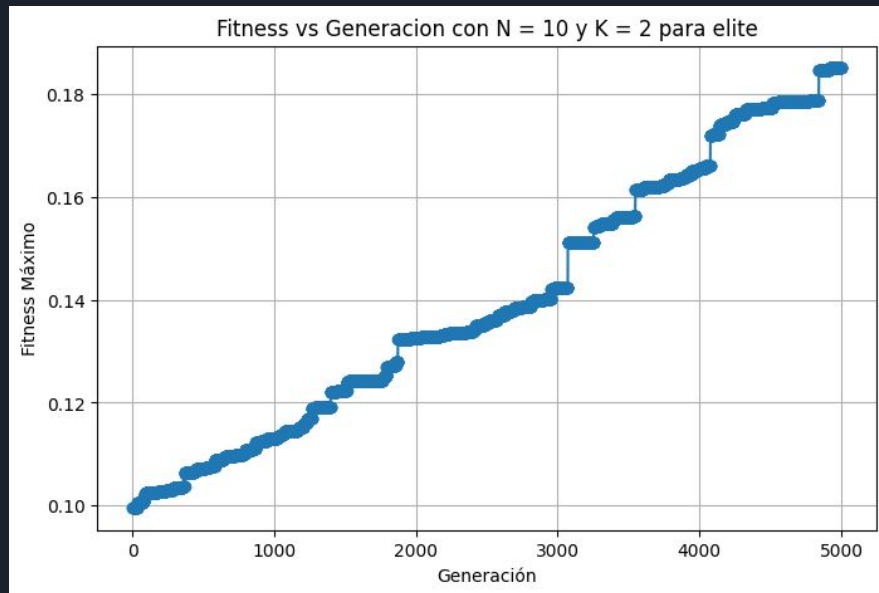
  "crossover_method": "uniform",
  "crossover_probability": 1,
  "p_uniform": 0.9,

  "mutation_method": "gene",
  "mutation_probability": 0.1,

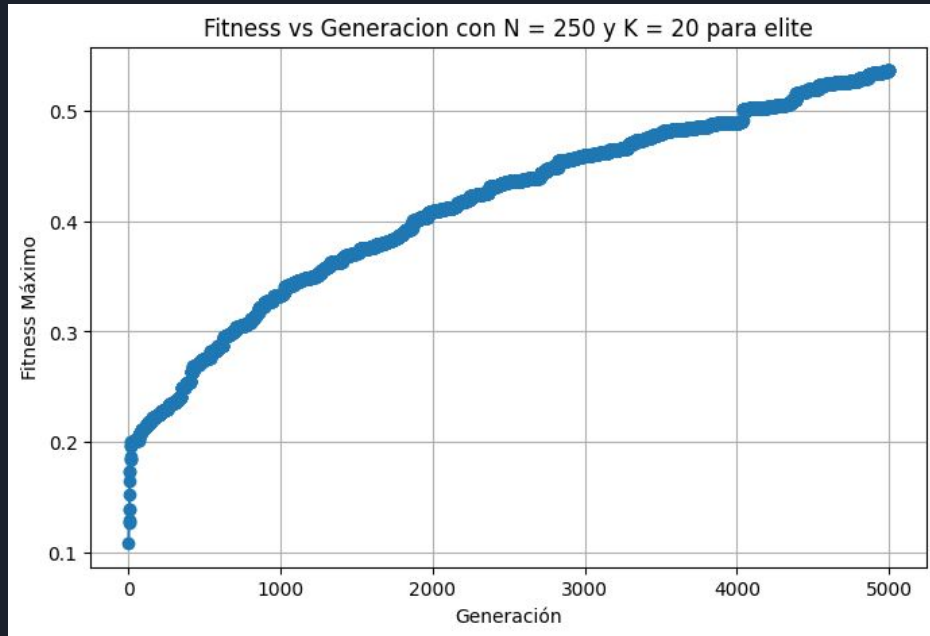
  "implementation": "traditional",

  "stop_condition": "max_generations",
  "stop_condition_max_generations": 5000
}
```

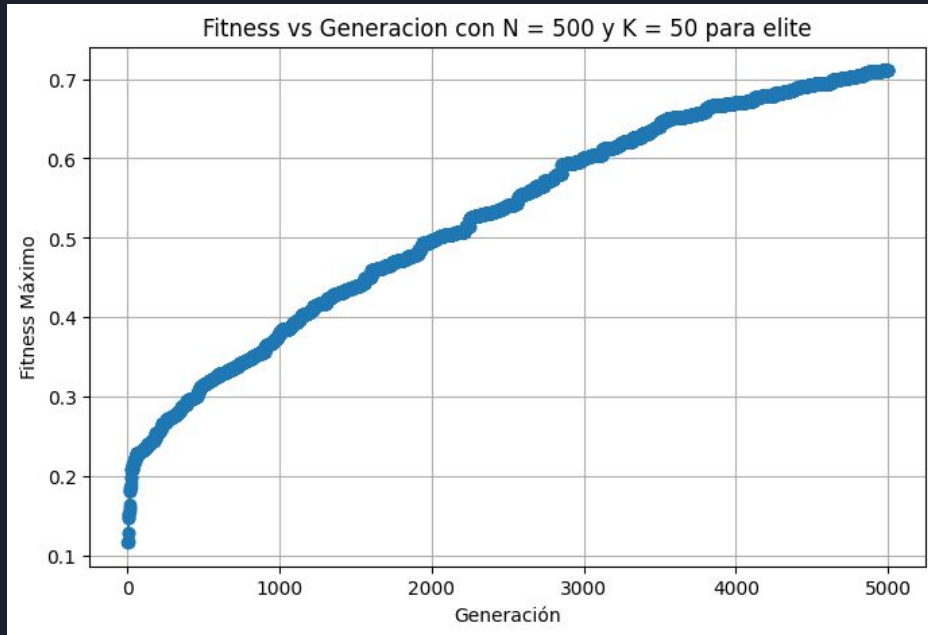
Tamaño de población



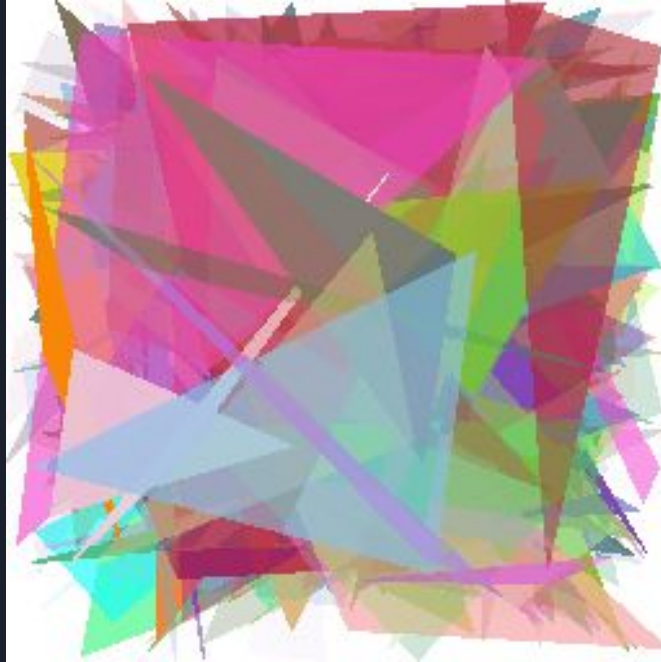
Tamaño de población



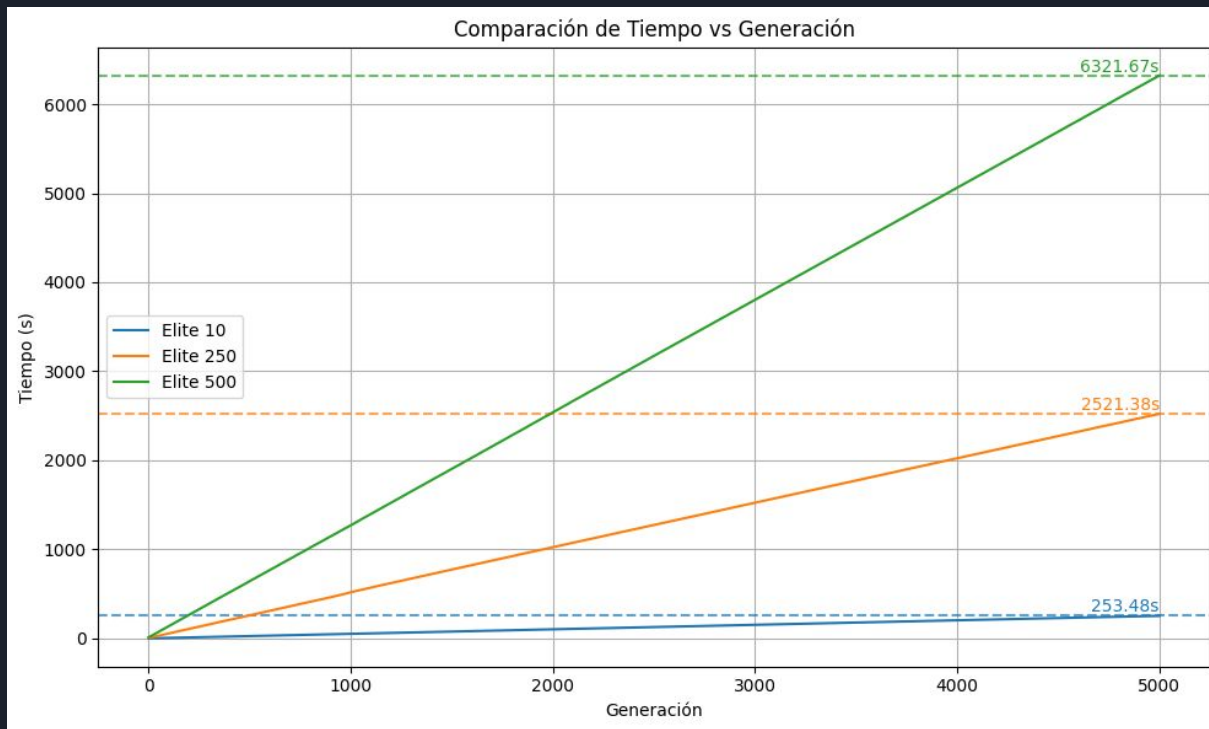
Tamaño de población



Tamaño de la población



Tradeoff





Modificando el método de cruza

```
{
  "n_population_size": 500,

  "selection_method": "deterministic_tournaments",
  "k_selection_size": 600,
  "m_selection_size": 100,

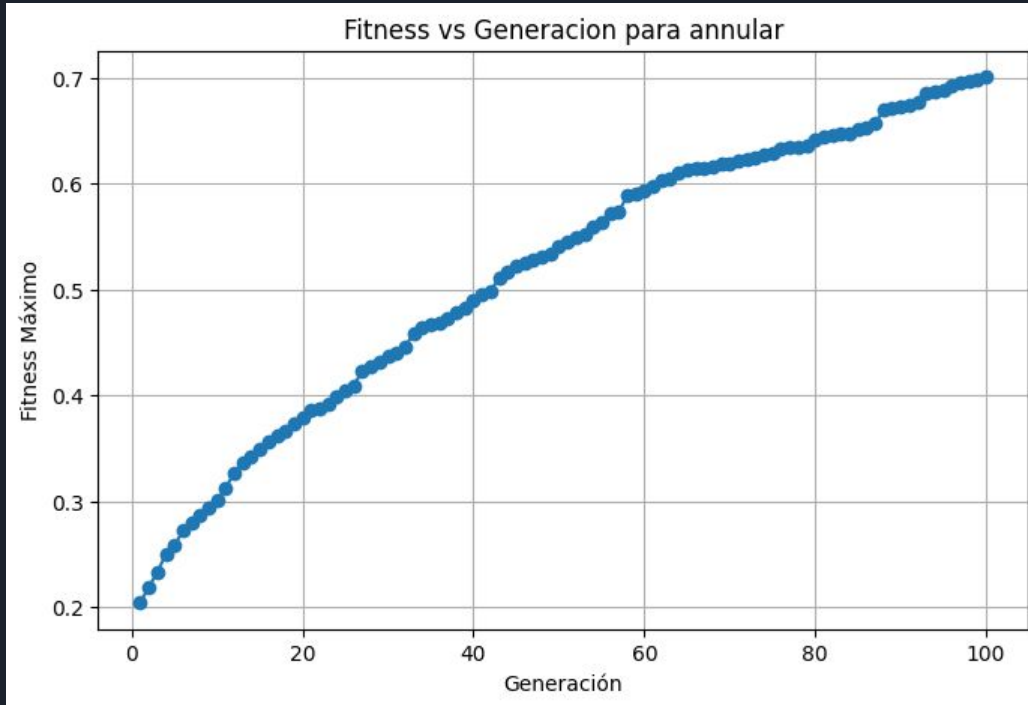
  "crossover_method": "annular",
  "crossover_probability": 0.9,

  "mutation_method": "multigen_uniform",
  "mutation_probability": 0.1,
  "mutation_M": 5,

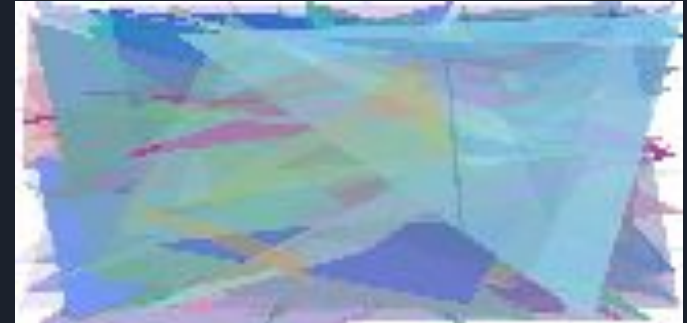
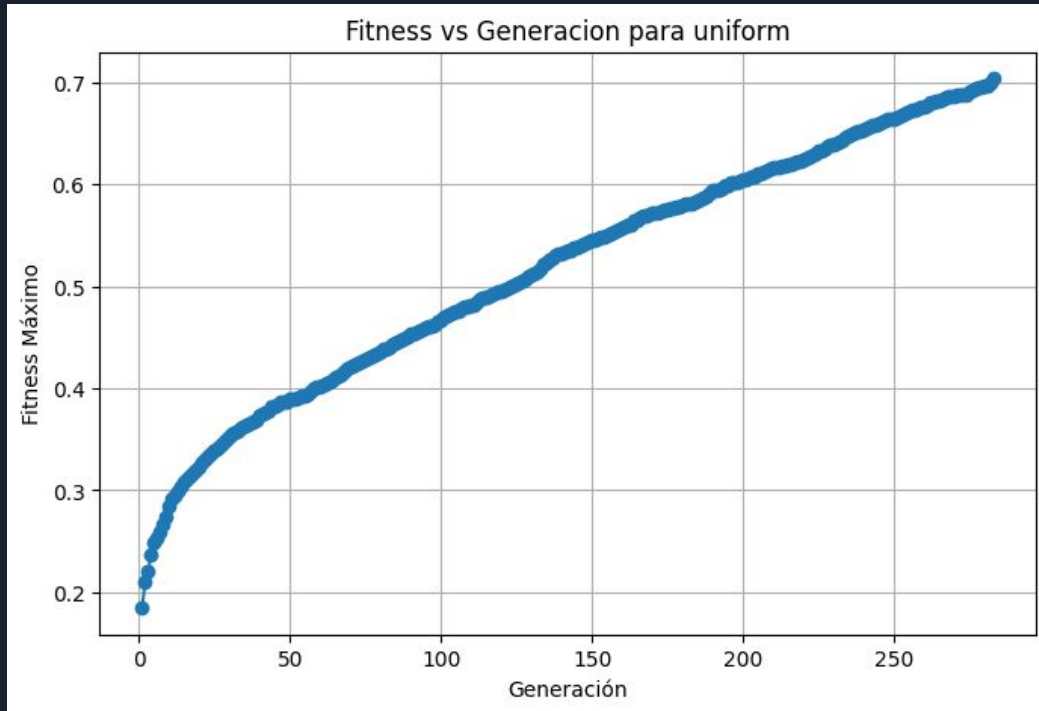
  "implementation": "young-bias",

  "stop_condition": "acceptable_solution",
  "stop_condition_acceptable_solution": 0.70,
  "stop_condition_max_time_seconds": 1500,
}
```

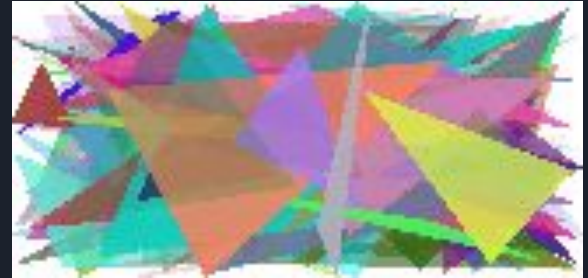
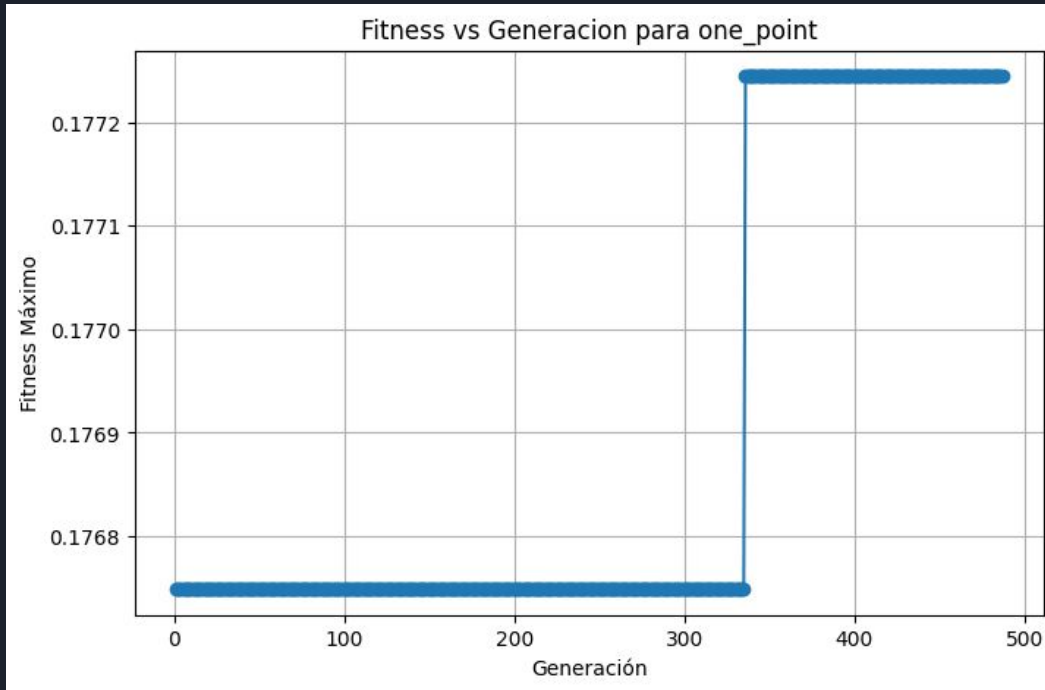
Modificando el método de cruza Anular



Modificando el método de cruce Uniforme



Modificando el método de cruza Un Punto



Un Punto con diferentes hiperparametros

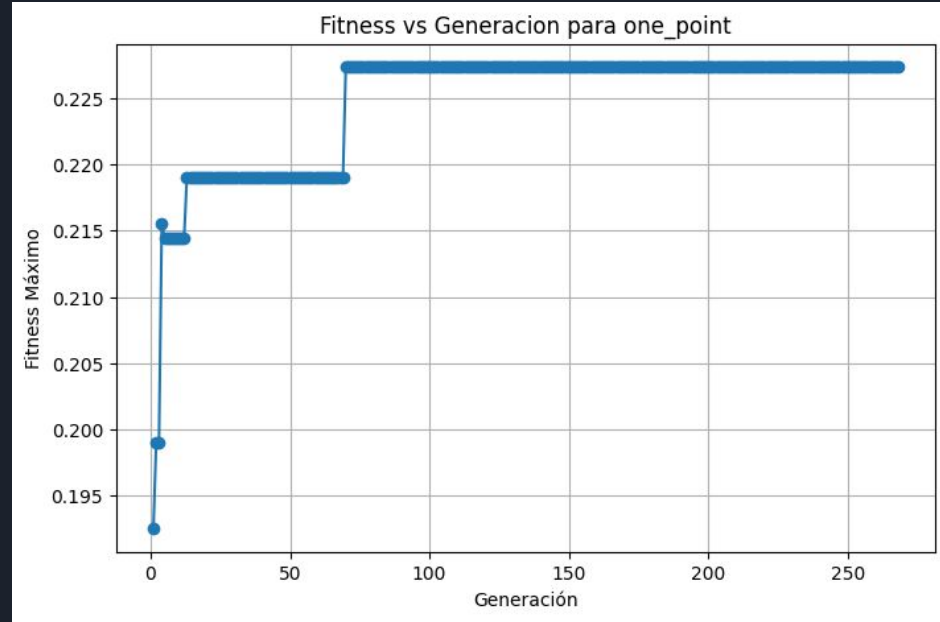
```
{
  "n_population_size": 500,

  "selection_method": "probabilistic_tournaments",
  "k_selection_size": 300,
  "m_selection_size": 100,
  "threshold": 0.8,
  "crossover_method": "one_point",
  "crossover_probability": 0.55,

  "mutation_method": "multigen_uniform",
  "mutation_probability": 0.1,
  "mutation_M": 5,

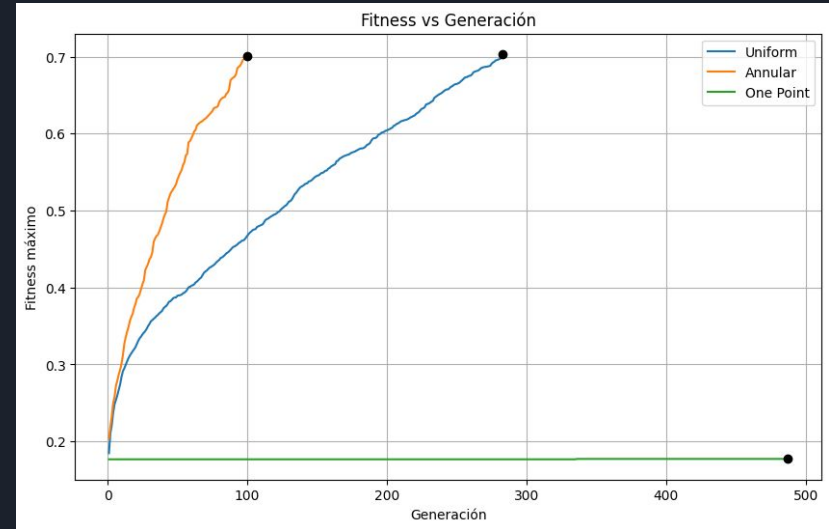
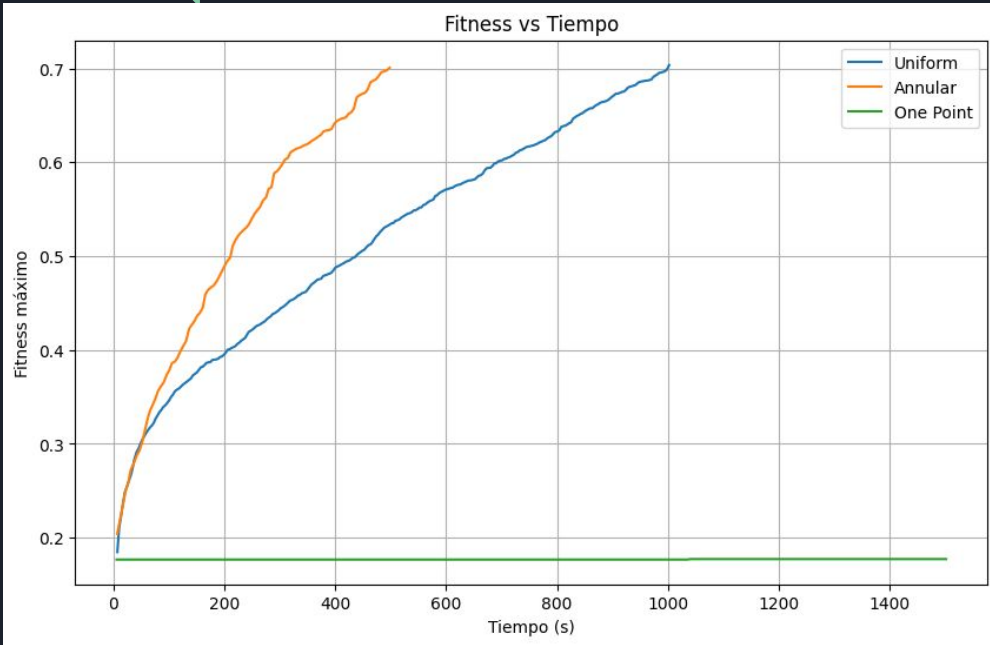
  "implementation": "traditional",

  "stop_condition": "max_time_seconds",
  "stop_condition_max_time_seconds": 1500
}
```



Modificando el método de cruce

Comparación





Modificando el Método de Selección

```
{
  "n_population_size": 500,

  "selection_method": "elite",
  "k_selection_size": 600,

  "crossover_method": "uniform",
  "crossover_probability": 1,
  "p_uniform": 0.9,

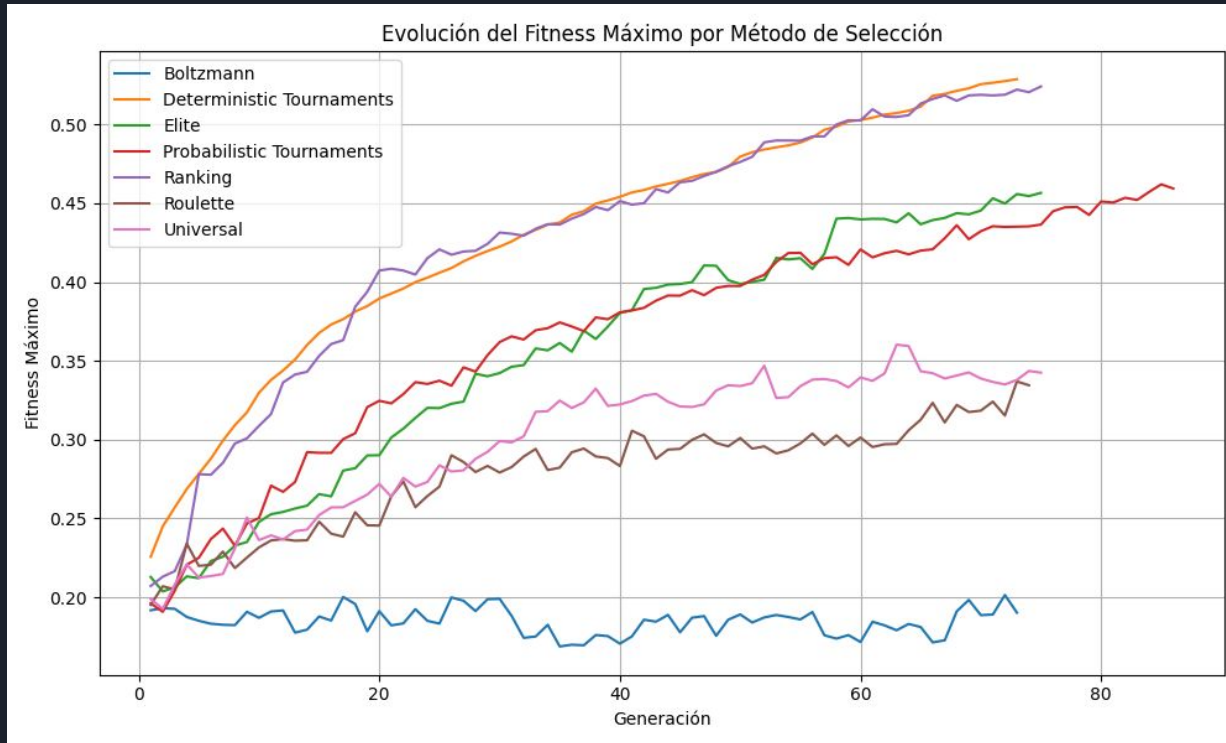
  "mutation_method": "multigen_uniform",
  "mutation_probability": 0.1,

  "implementation": "young-bias",

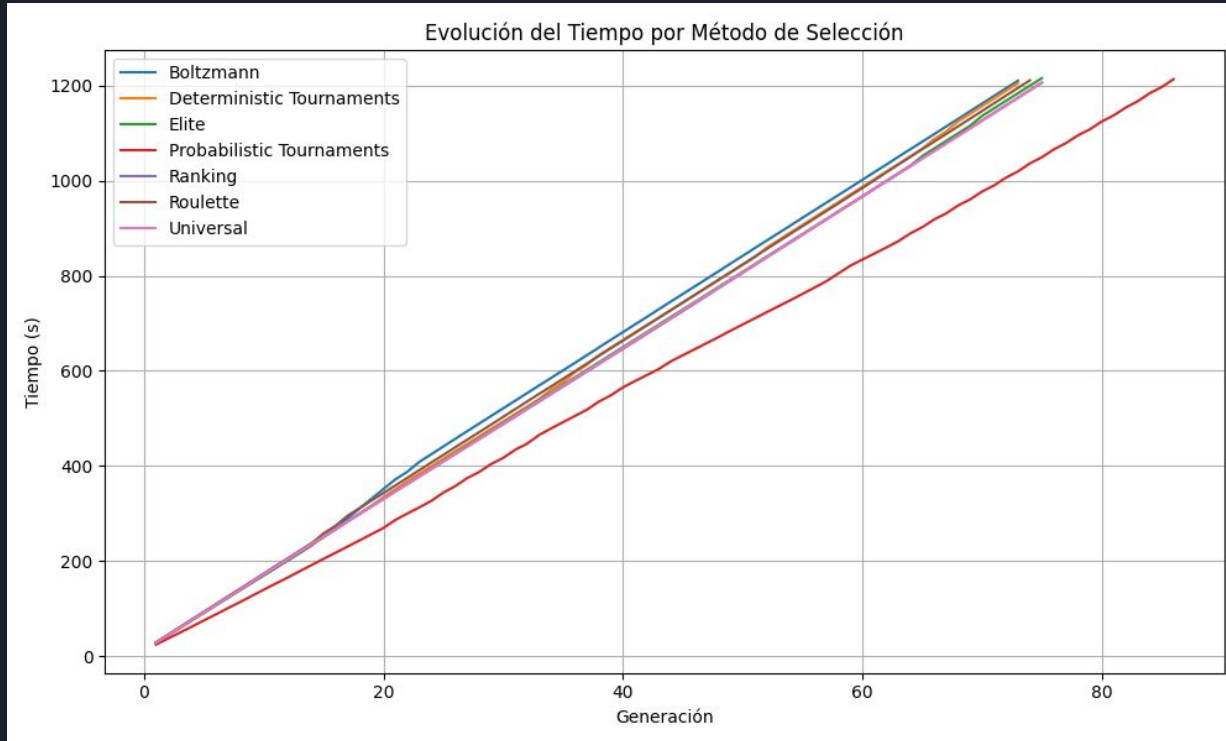
  "stop_condition": "max_time_seconds",

  "stop_condition_max_time_seconds": 1200
}
```


Modificando el Método de Selección



Modificando el Método de Selección





Opcionales




Paralelizando el cálculo de fitness



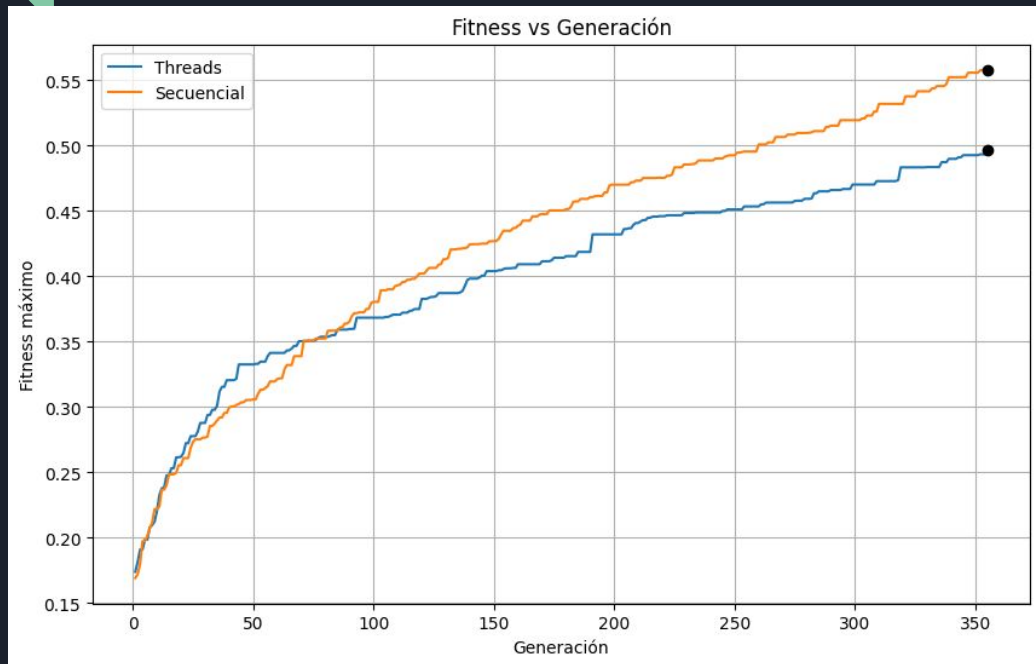


Usono ProcessPoolExecutor

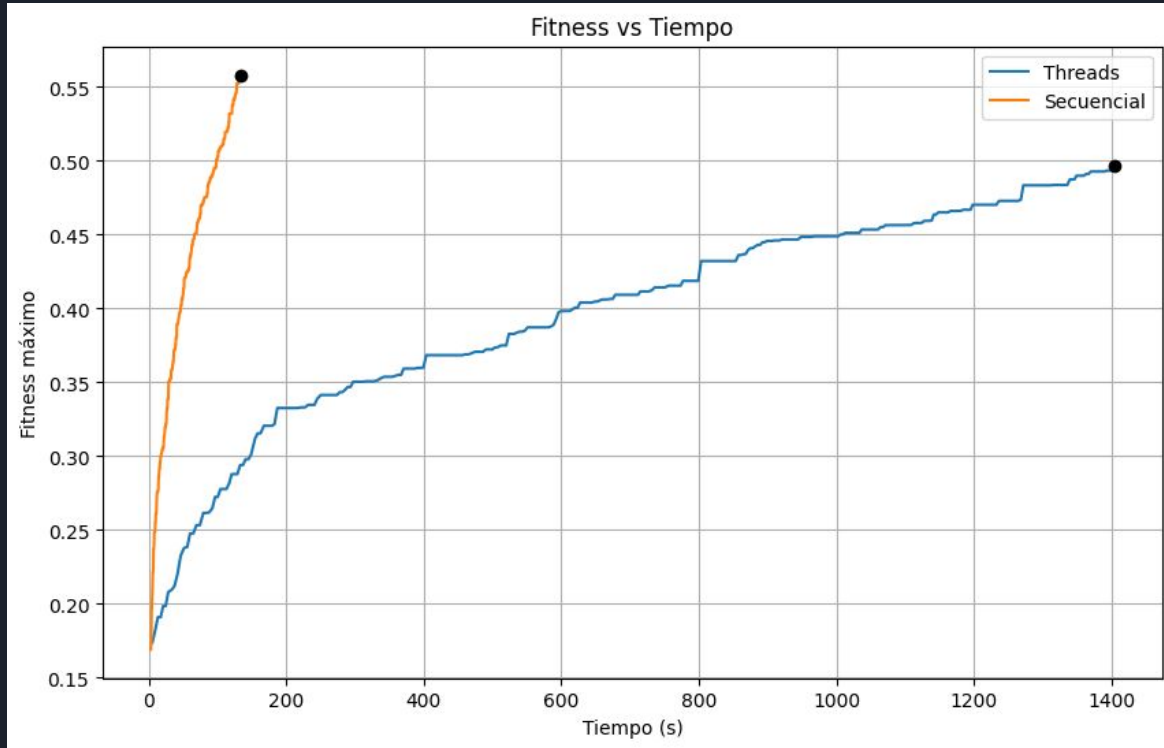


```
1  def elite(self, k_selection_size):
2      n_population_size = len(self.population)
3
4      if self.use_threads:
5          tasks = [(ind, self.fitness_obj, i) for i, ind in enumerate(self.population)]
6          with ProcessPoolExecutor(max_workers=4) as executor:
7              fitness_list = list(executor.map(evaluate_individual, tasks))
8      else:
9          fitness_list = [(individual, self.fitness_obj.fitness(individual), i)
10                          for i, individual in enumerate(self.population)]
```

Usando ProcessPoolExecutor



Usando ProcessPoolExecutor





Conclusiones

- Con un N pequeño la imagen resultante tiene un fitness relativamente bajo (así como su percepción visual) pero un tiempo de ejecución menor que utilizar un N grande para una imagen resultado mejor.
- En nuestro caso, el método de cruza anular resultó ser el más rápido manteniendo un buen fitness.
- No logramos obtener los resultados esperados con la paralización del cálculo del fitness, pero creemos que en un lenguaje en el que lo manejamos con mayor facilidad lo hubiésemos logrado (Java), o quizás teniendo más tiempo para investigar en Python.
- Es muy importante comprender la intervención de cada hiper parámetro para poder generar una imagen resultado lo suficientemente fiel en un tiempo de cómputo y coste computacional relativamente bajo.
- No podemos estimar fehacientemente el resultado de un método de selección (con sus hiperparametros) hasta no efectivamente correrlo.