INSTITUTO TECNOLÓGICO DE BUENOS AIRES

Sistemas de inteligencia artificial

Algoritmos genéticos

GRUPO 4 - Arquero 2

Alumnos

Martina Arco
Joaquín Ormachea
Lucas Emery
Diego Orlando

Fecha de Entrega

5 de Junio de 2019

Introducción

Como objetivo de este trabajo práctico, se nos propuso optimizar las combinaciones de características para los personajes de un juego de rol. Con el fin de encontrar este óptimo, desarrollamos un motor de algoritmos genéticos que las analiza.

A lo largo de este informe se va a explicar cómo desarrollamos el proyecto y las evaluaciones que realizamos para obtener conclusiones acerca de las distintas configuraciones posibles del algoritmo genético.

Estructura

Cromosoma

Diseñamos el cromosoma como una clase que cuenta con un vector llamado genes. Este consiste de 6 elementos, donde los primeros 5 son los id correspondientes a cada pieza de equipamiento, es decir, arma, casco, pechera, bota y guante, y el último es la altura donde cuando se inicializa o muta se hace entre los valores 1.3 y 2.

Población

Para la inicialización de población decidimos crear arrays aleatorios sobre los distintos genes para cada cromosoma. Esto nos permite partir de una base aleatoria sin generar ningún tipo de tendencia al aplicar el algoritmo genético.

Por otro lado, esto nos trajo un problema al querer realizar nuevas corridas con diferentes parámetros pero iguales poblaciones iniciales. Para solucionarlo, proporcionamos un parámetro semilla para que el usuario ingrese, haciendo que este sea el identificador de la población inicial. De esta forma la población es aleatoria pero se puede repetir si se desea.

La seed es un string de longitud igual al tamaño de la población donde cada carácter es utilizado como semilla para la generación de los genes de cada cromosoma. En el archivo utils se encuentra cómo generar una de acuerdo al tamaño de la población.

Implementación

Decidimos implementar el algoritmo de la forma más modularizada posible para poder dividir el trabajo y que cada uno se enfoque en implementar detalladamente cada parte.

Utilizando una larga lista de parámetros que mencionaremos a continuación, permitimos que el usuario pueda utilizar el motor de la forma que considere más

adecuada para el problema que está buscando resolver o encontrar la resultado que está esperando.

Selección

Para los algoritmos de selección tomamos de referencia los vistos en clase. Los podemos dividir en 2 tipos de algoritmos. Por un lado, los que realmente tomaban una decisión con respecto a qué cromosoma elegir y por el otro los que brindaban una nueva perspectiva de análisis para el fitness, como boltzman.

- Elite
- Roulette
- Ranking
- Universal
- Tournament
- Boltzman

Para este segundo tipo de algoritmos decidimos implementarlo como un parámetro que puede ser activado o no y dependiendo de esto se utilizan los valores que ese método propone para luego utilizar los demás algoritmos de selección.

Todos los implementamos utilizando una función wrapper que determina cual es el algoritmo que se desea utilizar en la instancia que se está ejecutando y deploya el método correspondiente. Luego el método devuelve un array de padres que fueron seleccionados para luego pasar por la etapa de cruza.

Al momento de realizar tests unitarios para los algoritmos de selección, nos encontramos con el problema de que estos utilizan variables aleatorias por lo que era complicado hacer tests sencillos. Para solucionarlo, realizamos tests reiteradas veces, demostrando que en el total, las opciones más probables siempre terminaban con mayor cantidad de resultados. Esto se puede observar por ejemplo que en varias corridas del test de ranking, el cromosoma con mayor fitness dentro de una población de 100, es elegido en el 98% de los casos en promedio.

Cruce

Siempre se generan 2 hijos a partir de 2 padres.

Implementamos:

- One point: a partir de un punto se invierten.
- Two points: Lo mismo pero con 2 límites de donde se hace el cruce.
- Uniforme: se tiene una probabilidad del 50% para hacer un swap de cada gen.
- Anular: se tiene un punto y una longitud y se hace un swap desde ese punto hasta la longitud que te dan.

Esta parte del proyecto no nos trajo problemas.

Mutación

Se implementó mutación gen, donde solo un gen de cada cromosoma puede mutar, y mutación multigen, donde todos los genes de todos los cromosomas pueden mutar. Ambos algoritmos pueden ser uniformes, donde el rate de mutación no va cambiando, o no uniformes, donde el rate de mutación va decayendo de manera exponencial.

Reemplazo

Se implementaron los tres métodos de reemplazo vistos en la teórica:

- El primer método de reemplazo simplemente deja pasar todos los hijos generados, que son N, a la nueva generación.
- El segundo método recibe todos los hijos generados, que en este caso pueden ser k < N, y los deja pasar al la próxima generación. Además, selecciona N - k de la población de padres que pasarán a la nueva generación sin modificaciones, así quedando la nueva generación con tamaño N.
- El tercer método recibe todos los hijos generados, que en este caso también pueden ser k < N, pero este selecciona, N - k de la población total y luego selecciona k de la población de N padres + la población de k hijos (N + k), asi quedando la nueva generación de tamaño N.

Los métodos 2 y 3 pueden recibir como parámetro dos tipos de selección diferentes, con el porcentaje de la muestra que seleccionara cada algoritmo de selección.

Condiciones de corte

Los criterios de corte utilizados fueron:

- Máxima cantidad de generaciones: se setea una generación máxima a la que se puede llegar y cuando se llega corta.
- Contenido: cuando se pasa de generación, se chequea si el fitness más alto no se modificó en las últimas n generaciones (definidas por parámetro) en cuyo caso se corta.
- Estructura: se analiza si los cromosomas con mayor fitness varían de una generación a la otra. Para lograr esto se pasan dos parámetros, uno determina qué porcentaje de la población se quiere considerar, de esta forma si se pasa 0.5 se va a analizar la mitad de la población, si se pasa 1 todos los individuos deben tener los mismos genes. Además de debe pasar durante cuántas generaciones se quiere mantener constante.
- Entorno a un óptimo: se corta cuando el mejor fitness de una generación llega al fitness óptimo que se pasa por parámetro. En este problema en particular

esto no aplica ya que no se tiene un óptimo porque no hay condiciones particulares que se quieran cumplir. El máximo fitness al que llegamos, que fue con una alta mutación y utilizando muchos hijos en la población, fue de 51.

Pruebas

El objetivo de las pruebas será poder comparar las distintas configuraciones en relación a una configuración por defecto. Para esto implementamos un script de bash que ejecuta nuestro programa utilizando de base los siguiente parámetros y luego iterando por las distintas variantes que queremos comparar. Tener en cuenta que todas las corridas se hacen con la misma población inicial determinada por la seed.

- **population_size** = 100
- **k** = 50
- **seed**=cRYEJJgvmksuIiEnJZJmdtBzsJjkxCHdAMvsSJkWLvEwucvJuByqumCUHKSSPhYAAbJpfcMQjEmqlazvURqjTLQDfxpMrZNGaJQU
- **stop_condition** = content
- **crossover_algorithm** = one_point
- **mutation_algorithm**=uniform_gen
- **selection_algorithm_1**=roulette
- **selection_algorithm_2**=probabilistic_tournament
- **selection_algorithm_3**=roulette
- **selection_algorithm_4**=probabilistic_tournament
- **selection_algorithm_5**=tournament
- **scaling_algorithm** = none
- **initial_temperature** = 100
- **temperature_step** = 0.1
- **replacement_method** = 1
- **percentage_for_selection** = 0.5
- **percentage_for_replacement** = 0.5
- **max_generation** = 20000
- **fitness_max** = 48
- **population_percentage_to_say_equals** = 0.7
- **generation_number_to_say_equals** = 1000
- **prob_mutation** = 0.2
- **rate_mutation** = 0.001

Algoritmo de cruza - *gráficos 1, 2, 3*

- Un punto
- Dos puntos
- Uniforme
- Anular

A partir de los gráficos podemos observar que no hay tantas diferencias solamente variando el método de cruza. Se puede destacar que con anular se obtienen los avances más prematuros y que con dos puntos se acerca rápidamente

al óptimo. Podemos adjudicar estos desarrollos a que anular y dos puntos son los métodos más aleatorios por lo que pueden encontrar un camino orientado al maximo correcto desde el principio ya que de genera una mayor variación de individuos.

Algoritmo de mutación - *gráficos 4, 5, 6*

- Gen uniforme
- Gen no uniforme
- Multigen uniforme
- Multigen no uniforme

En esta prueba observamos que la mutación gen no uniforme se encontró con un máximo local. Esto es debido a la disminución de la probabilidad de mutación por lo que a medida que avanzaron las generaciones, dejó de mutar y se estancó en un tipo de gen. Esto se puede ver muy claro en el gráfico de dispersión ya que, termina prácticamente nula mientras que los demás no. Además, el hecho de que la mutación sea de un sólo gen y no de múltiples también hace que disminuya la diversidad.

Por otro lado, se puede ver que multigen no uniforme encuentra rápidamente un camino correcto al tipo de cromosoma “óptimo” y se concentra en esos, dejando de mutar y generando rápidamente una población “óptima”. Encontrar el cromosoma óptimo se debe a que en este método la probabilidad de mutación se aplica en más de un gen a la vez.

En cuanto a los uniformes, se puede observar que a veces llegan a tener una línea horizontal constante en el fitness pero luego vuelven a subir. No llegan tan rápido como no uniforme multi gen debido a que la probabilidad de mutación no baja, pero logra seguir subiendo y no estancarse como gen no uniforme.

Algoritmo de escalamiento - *gráficos 7, 8, 9*

- None
- Boltzmann

Estas pruebas se realizaron con selección Roulette para que el escalamiento del fitness tenga un efecto apreciable. Se puede observar cómo con Boltzmann la diversidad inicial es mayor, ya que la temperatura es alta y la presión baja, y cómo a medida que pasan las generaciones y la temperatura disminuye, y por ende la presión aumenta, la diversidad es menor que sin escalamiento.

Algoritmo de selección - *gráficos 10, 11, 12*

- Elite
- Roulette
- Universal
- Tournament
- Probabilistic_tournament
- Ranking

Para esta prueba se analizó la utilización de diferentes algoritmos de selección para elegir qué individuos van a generar hijos. Se puede ver que torneo probabilístico fue el que más perduró ya que es el que contiene mayor variación al seleccionar, es decir, no se hace exclusivamente por fitness sino que es aleatorio. Además, fue el que llegó a un fitness más alto ya que esta variación le permitió salir de los máximos locales y seguir explorando.

Ruleta y universal muestran un comportamiento similar y cortan en el mismo punto aproximadamente. Son los que consiguen fitness más bajo. Esto puede ser porque en ruleta, si hay un fitness muy superior al resto, este cromosoma se va a repetir muchas más veces, disminuyendo la variación de cromosomas en la población.

Por último, elite tornamento y ranking son los que más rápido llegan a un máximo local y consecuentemente terminan. Sin embargo, no alcanzan su máximo potencial de fitness ya que llegan a un máximo local y se quedan estancados.

Algoritmo de reemplazo - *gráficos 13, 14, 15*

- Método 1
- Método 2
- Método 3

Con estos gráficos se puede ver claramente cómo el método 1 es el que más tarda en cortar y el que más variación de cromosomas tiene. Esto ocurre porque los hijos, que ya sufrieron una cruza y posible mutación, son los únicos que pasan a la siguiente generación. En cambio, en los otros pueden pasar algunos padres inalterados. El punto medio de esto sería el método 2 donde pasan todos los hijos generados y se seleccionan algunos padres. En el 3, se eligen los mejores de entre todos los individuos, por eso tiende más a quedarse en un máximo local.

Algoritmo de selección con reemplazo 2 - *gráficos 16, 17, 18*

- Elite
- Roulette
- Universal
- Tournament
- Probabilistic_tournament
- Ranking

En estos gráficos se puede apreciar el efecto de los diferentes métodos de selección de cromosomas que pasan a la siguiente generación sin modificar. Los métodos mas con mas pendiente fueron Ranking, tournament y elite pero se terminan estancando en máximos locales y son alcanzados por los demás métodos. Sin embargo, a la larga elite y ranking son los métodos que obtienen mejor fitness, por lo que parecerían ser buenas opciones.

Ruleta y universal suben muy lentamente, esto se debe a que repiten los cromosomas elegidos y en el caso de ruleta si hay uno con fitness muy superior a otro, este cromosoma puede repetirse varias veces al elegirse y se reduce la diversidad.

Algoritmo de selección de parejas - *gráficos 19, 20, 21*

- Elite

- Roulette
- Universal
- Tournament
- Probabilistic_tournament
- Ranking

Estos gráficos se generaron variando el algoritmo de selección en la elección de los dos padres para cruzar. Se puede apreciar cómo torneo probabilístico fue el primero en llegar a un fitness elevado, es el que tiene mayor pendiente y es el que mayor fitness tiene. Esto es porque es el que cuenta con mayor aleatoriedad, de forma que se crea más variación y así se genera una mayor posibilidad para llegar a un fitness elevado. Elite también fue muy performante, ya que llegó a un fitness alto, el problema fue que se estancó en un máximo local del cual no sale. Esto se explica porque siempre se van a elegir los mejores padres para cruzar pero no se genera variedad. Ranking muestra un comportamiento similar.

Torneo probabilístico es que mayor variación de fitness tiene ya que es el más aleatorio de todos.

Finalmente, ruleta y universal son parecidos ya que son un intermedio entre los muy orientados a fitness como el primer grupo y los muy aleatorios como torneo probabilístico. Así, logran tener variedad pero su pendiente no es tan alta.

Anexo

Prueba de cruza

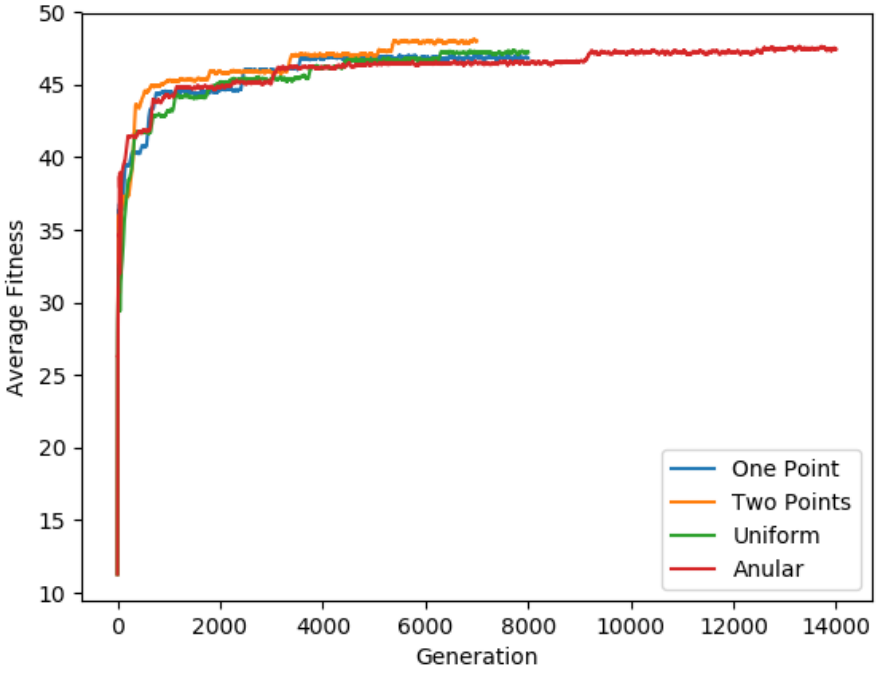

Gráfico 1

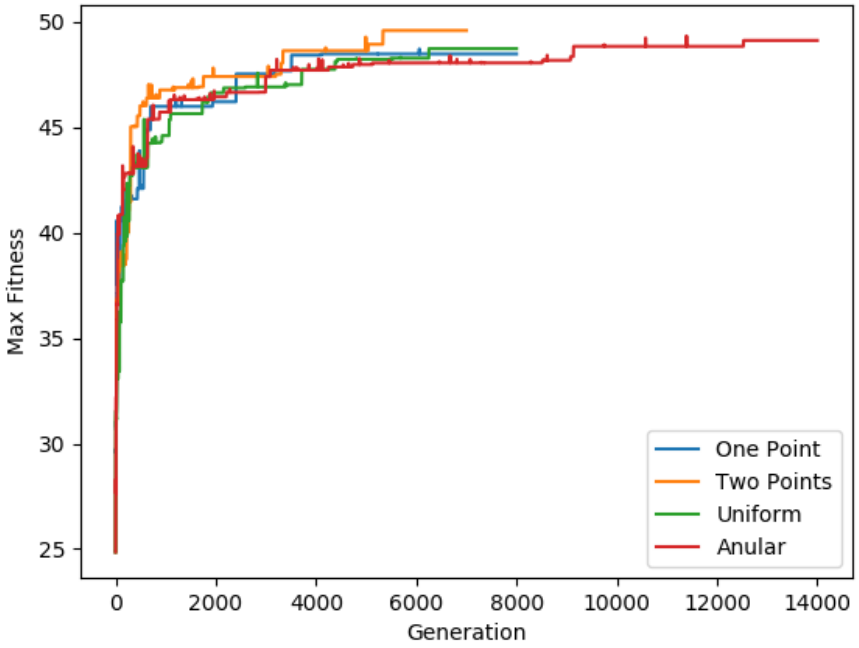

Gráfico 2

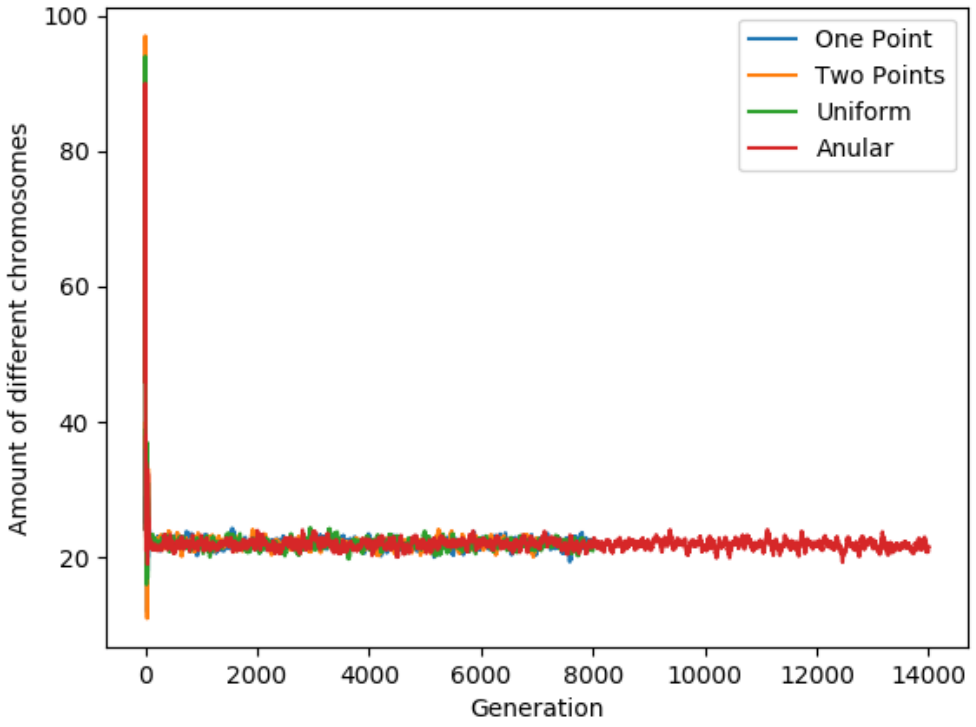

Gráfico 3

Prueba de mutación

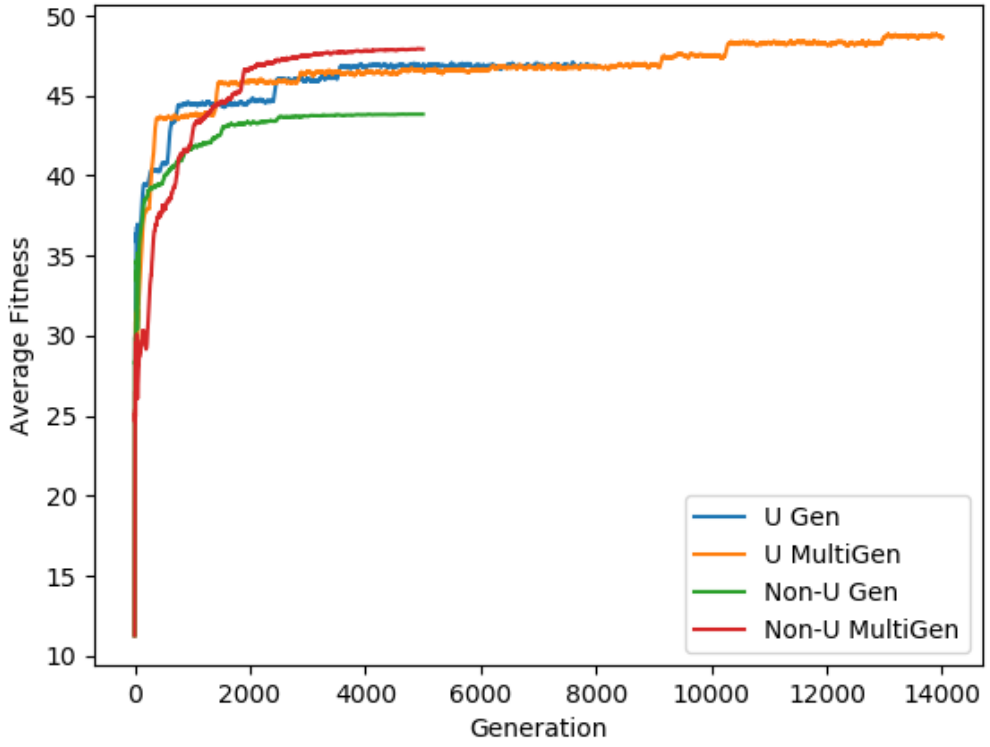

Gráfico 4

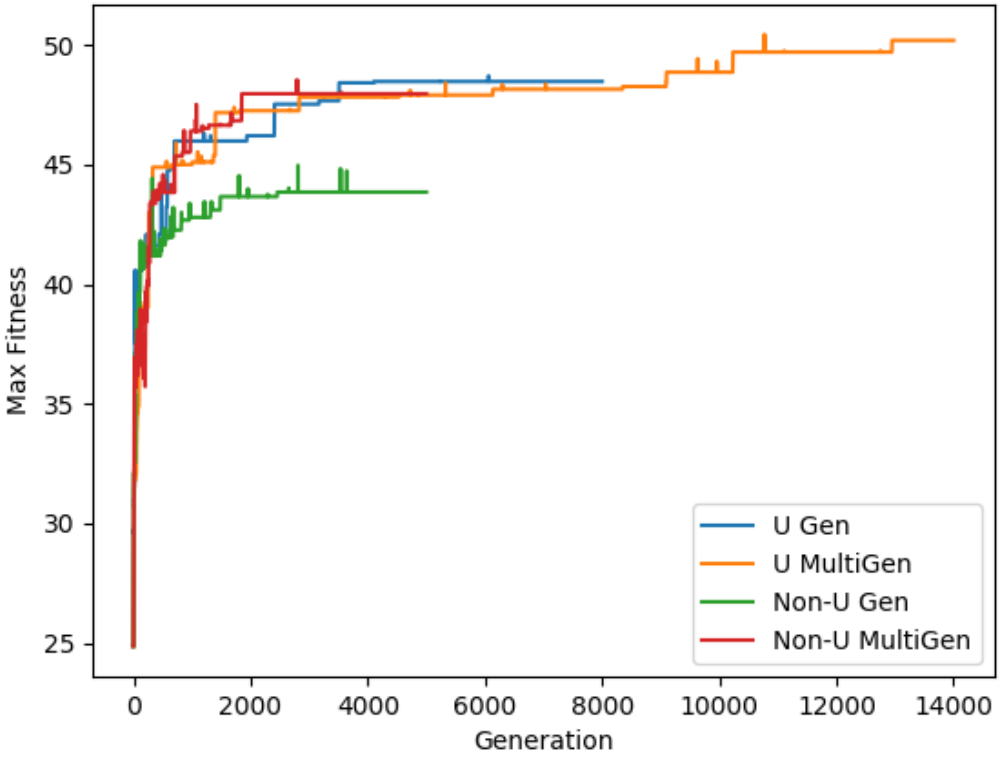

Gráfico 5

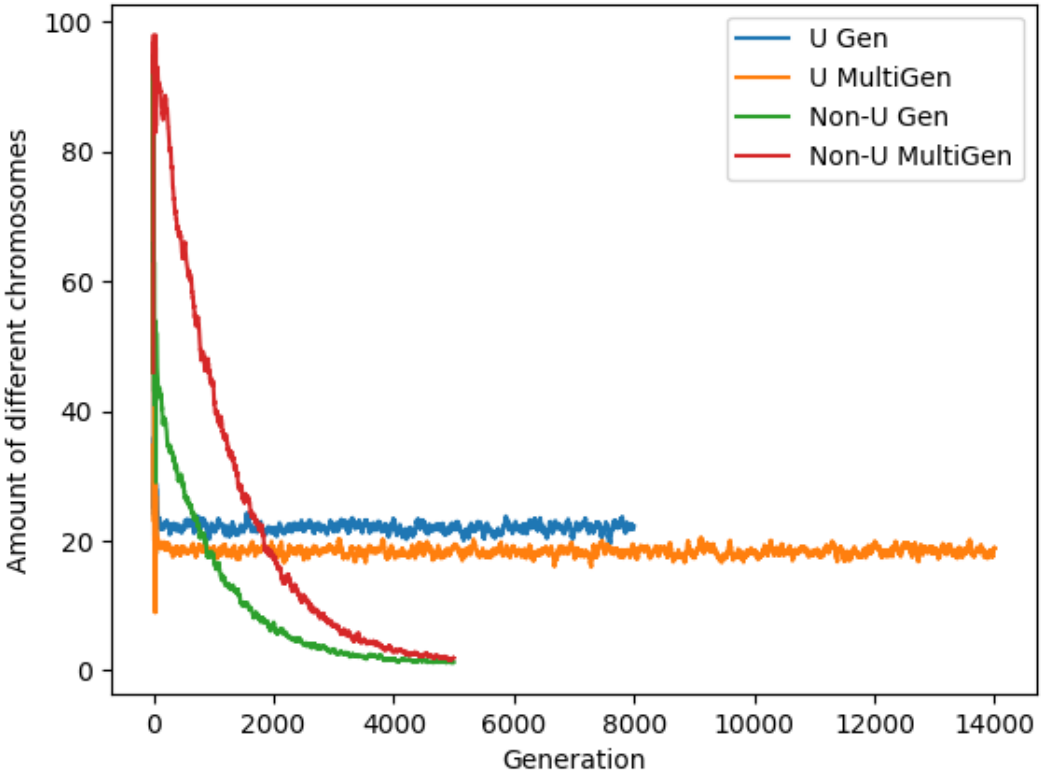

Gráfico 6

Prueba de scaling

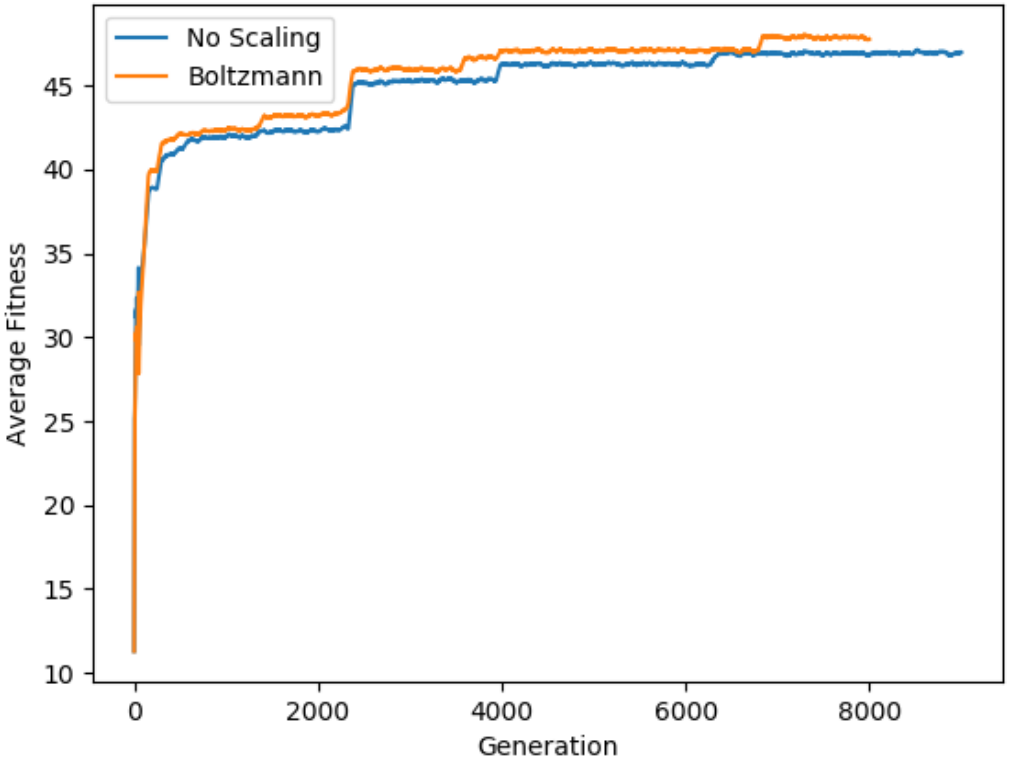

Gráfico 7

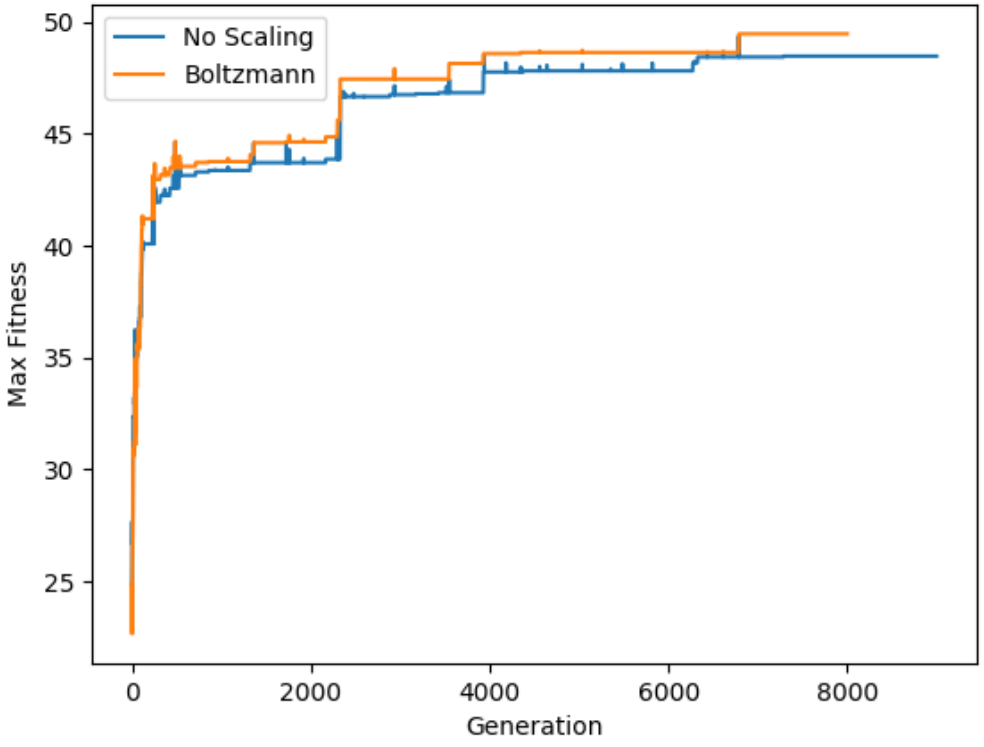

Gráfico 8

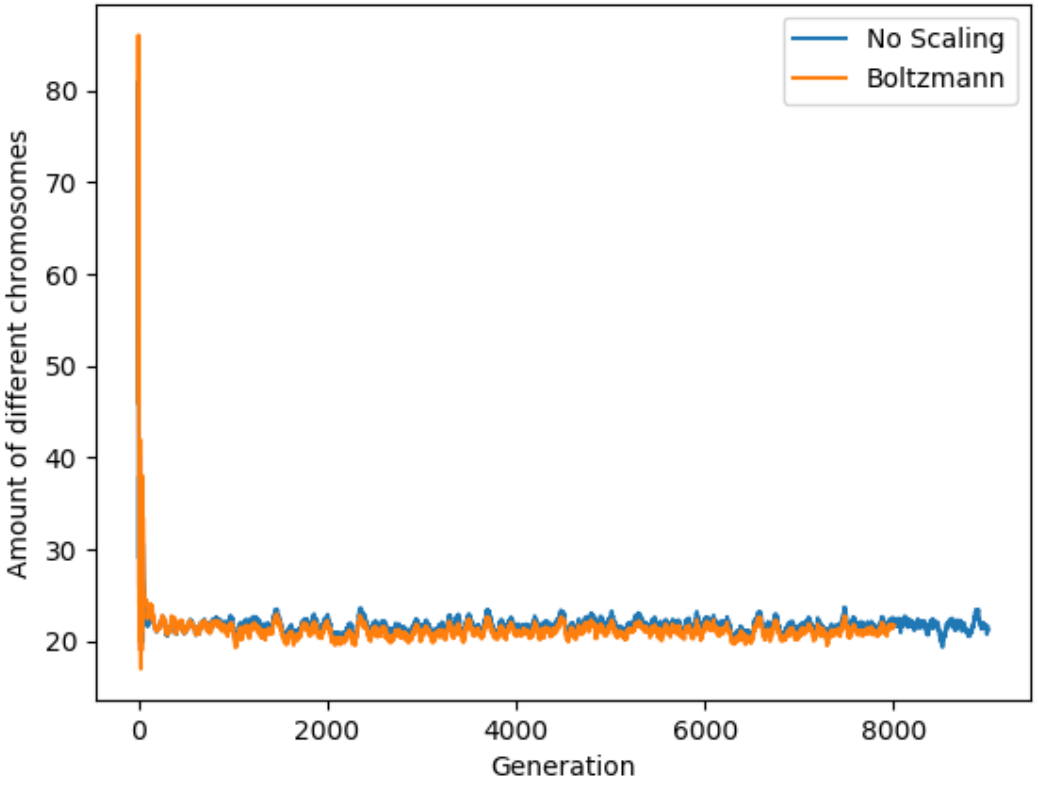

Gráfico 9

Prueba de selección

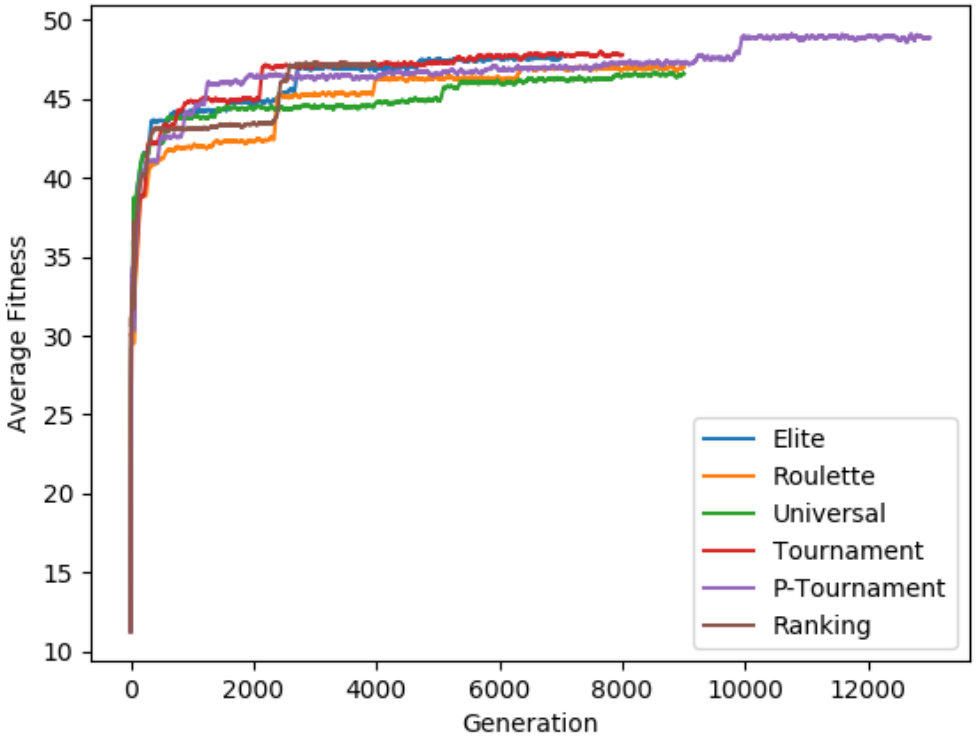

Gráfico 10

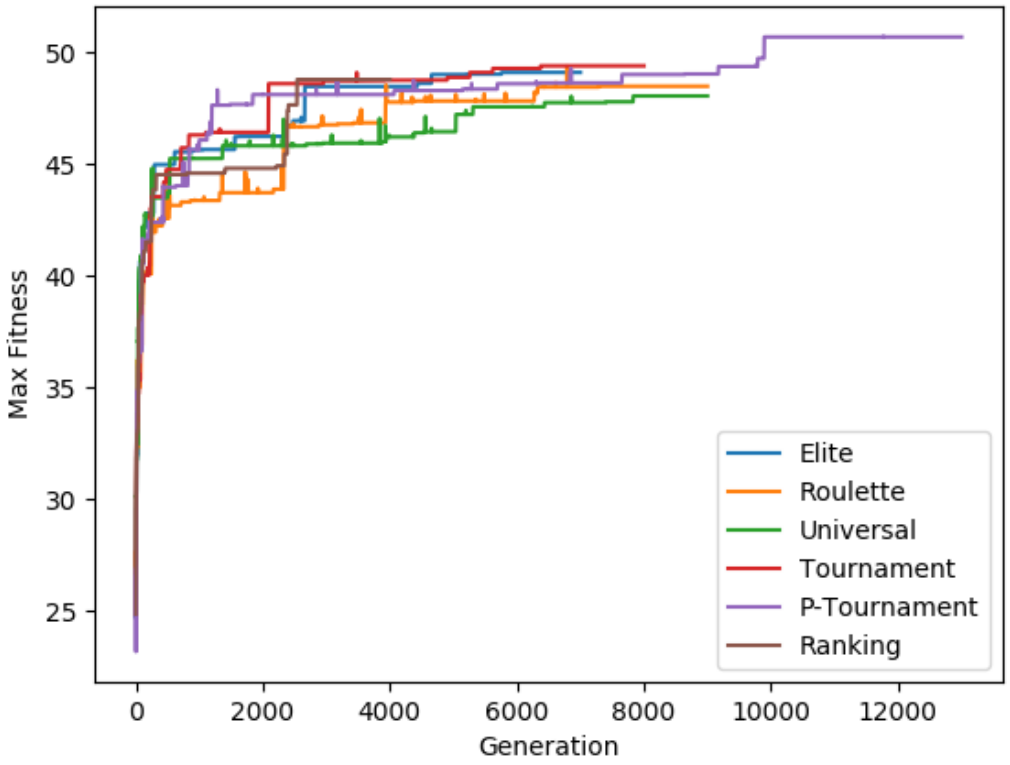

Gráfico 11

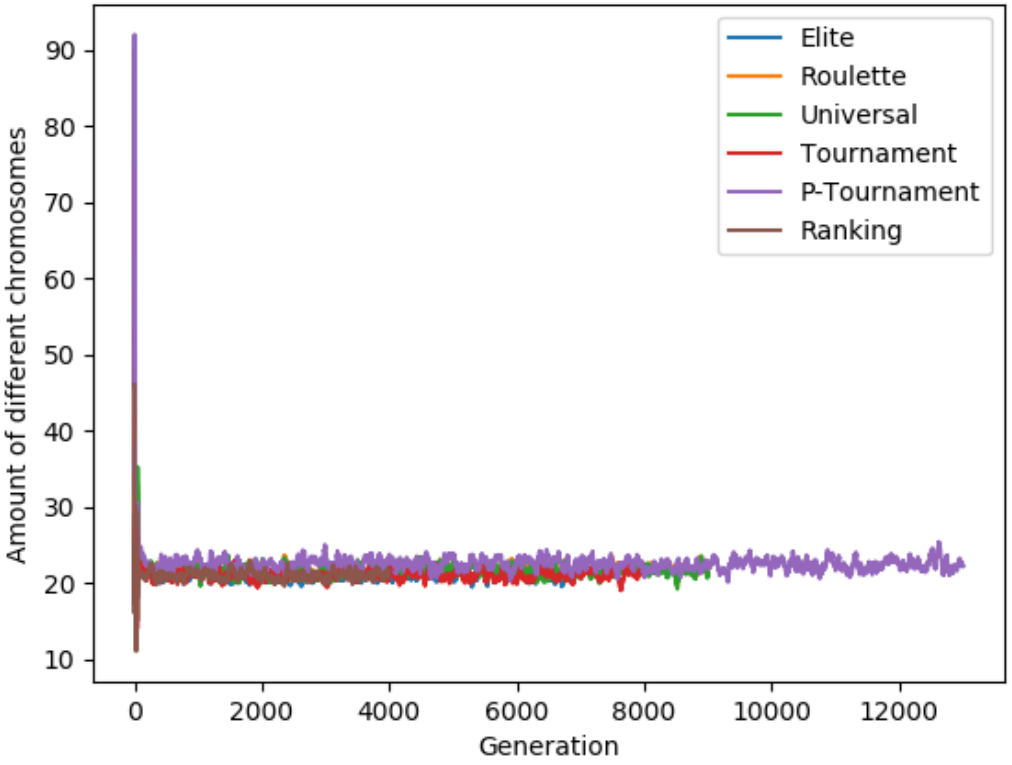

Gráfico 12

Prueba de reemplazo

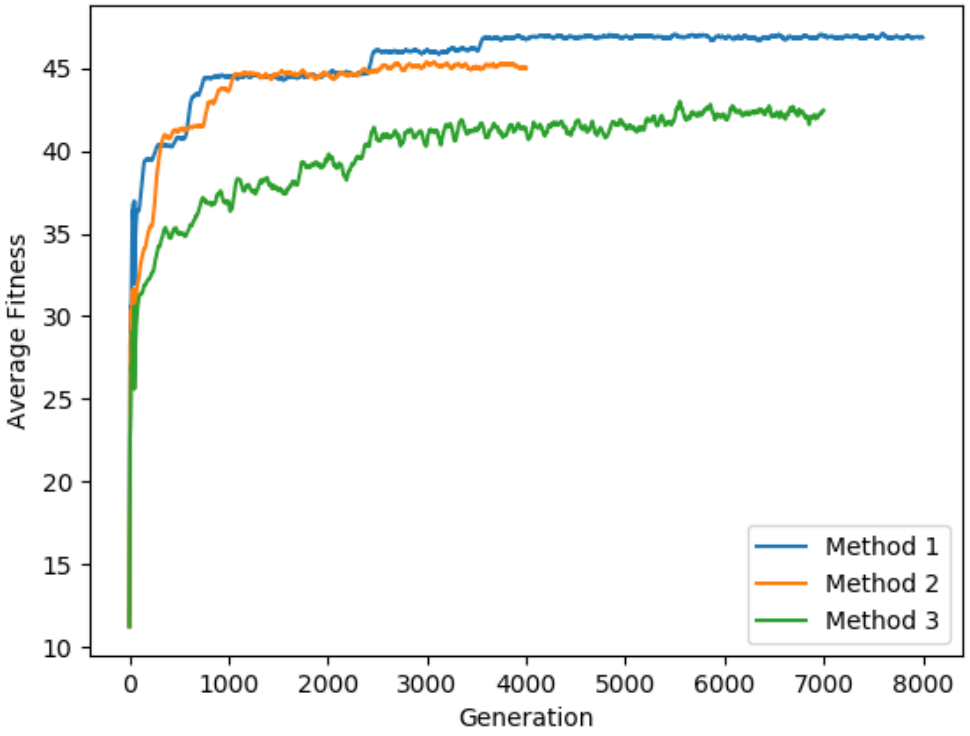

Gráfico 13

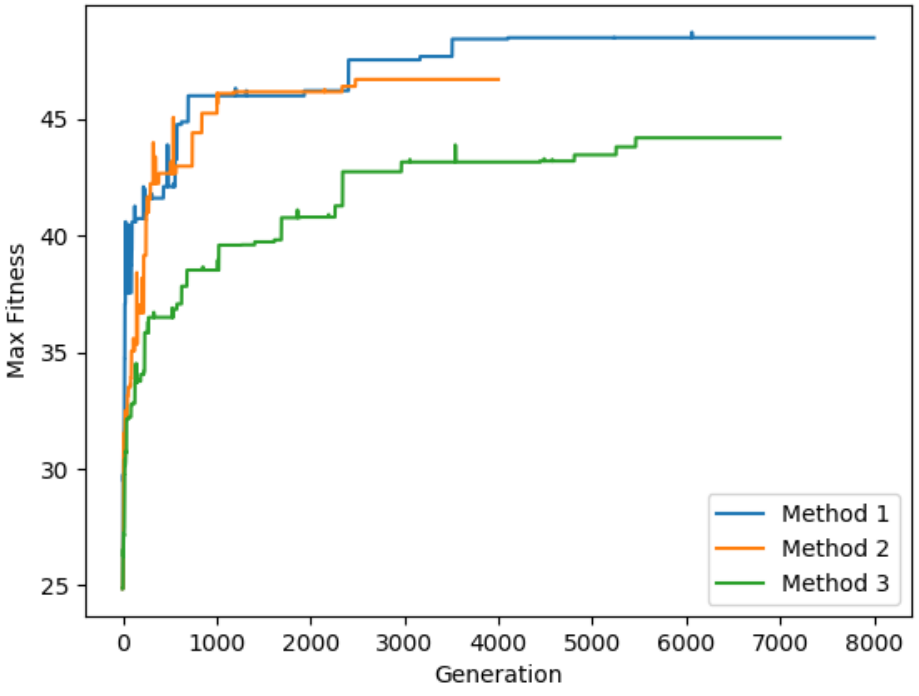

Gráfico 14

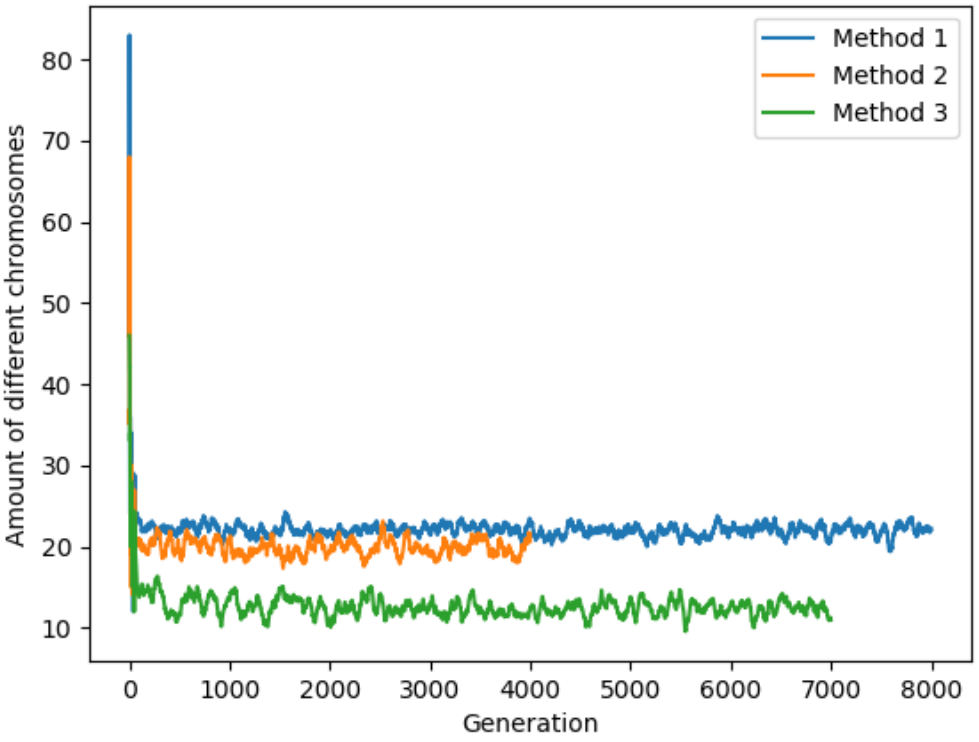

Gráfico 15

Prueba de selección en reemplazo con método 2

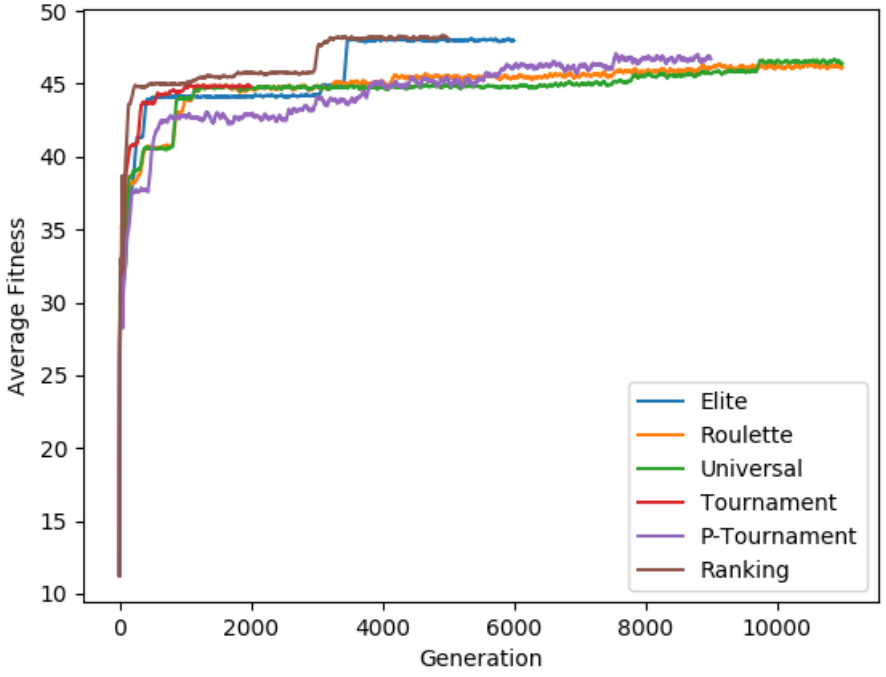

Gráfico 16

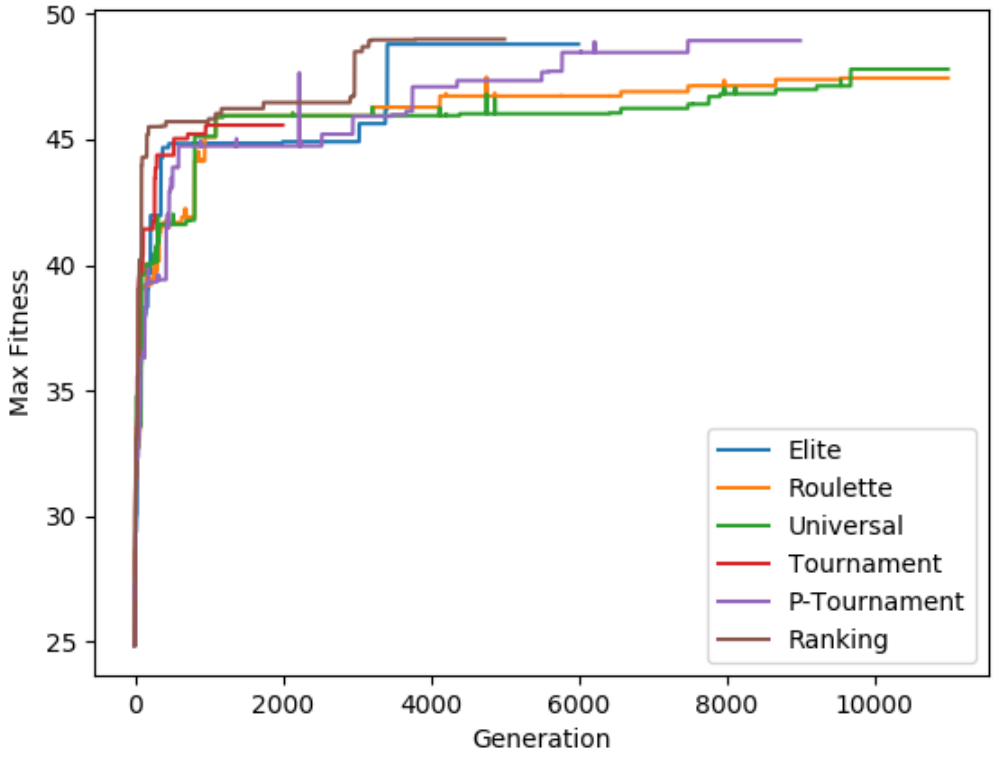

Gráfico 17

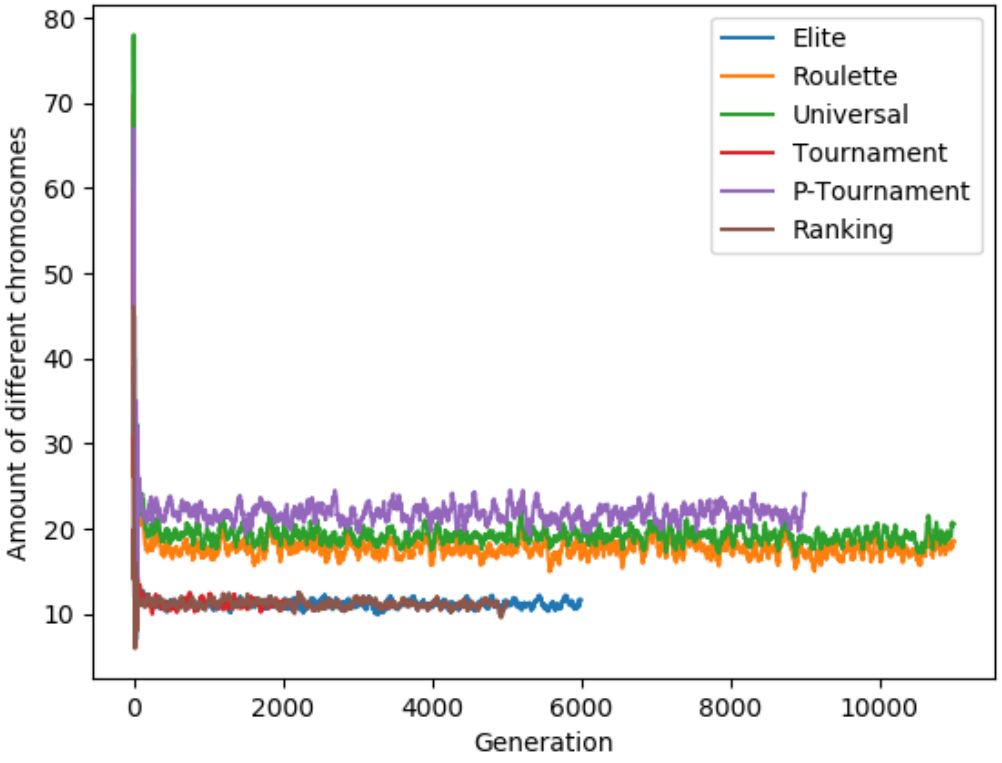

Gráfico 18

Pruebas de selección de parejas para crossover

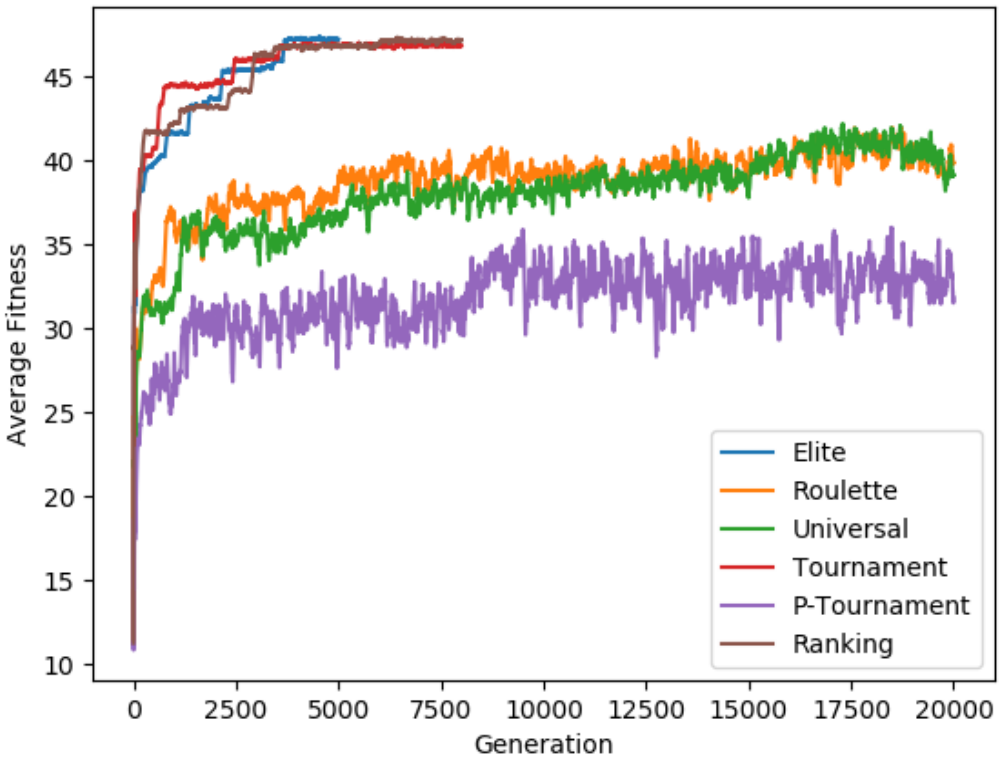

Gráfico 19

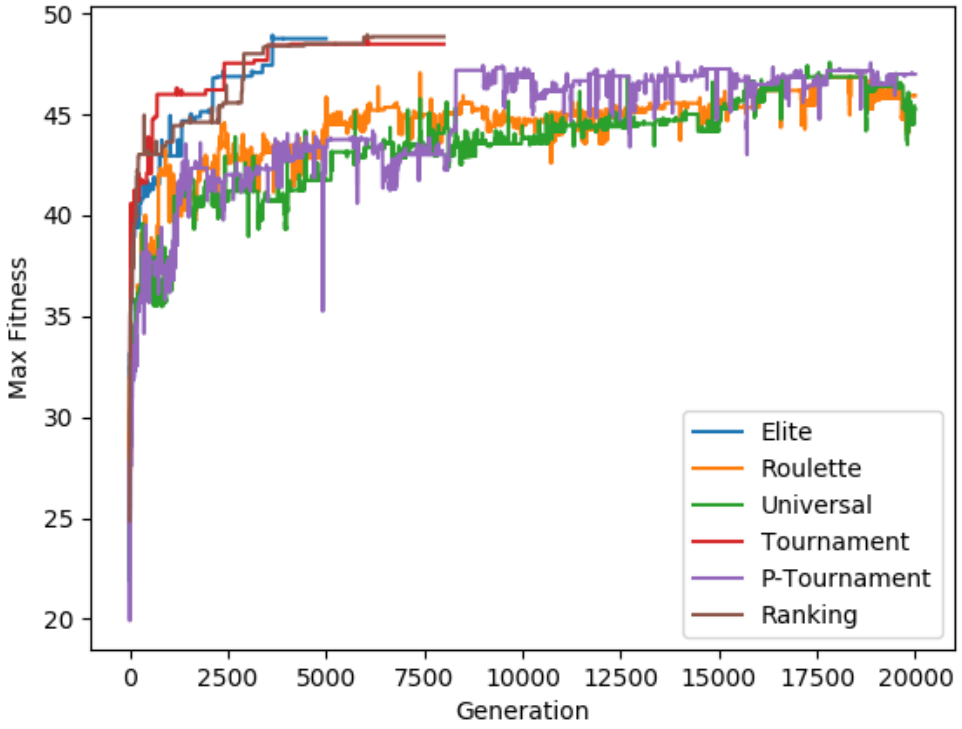

Gráfico 20

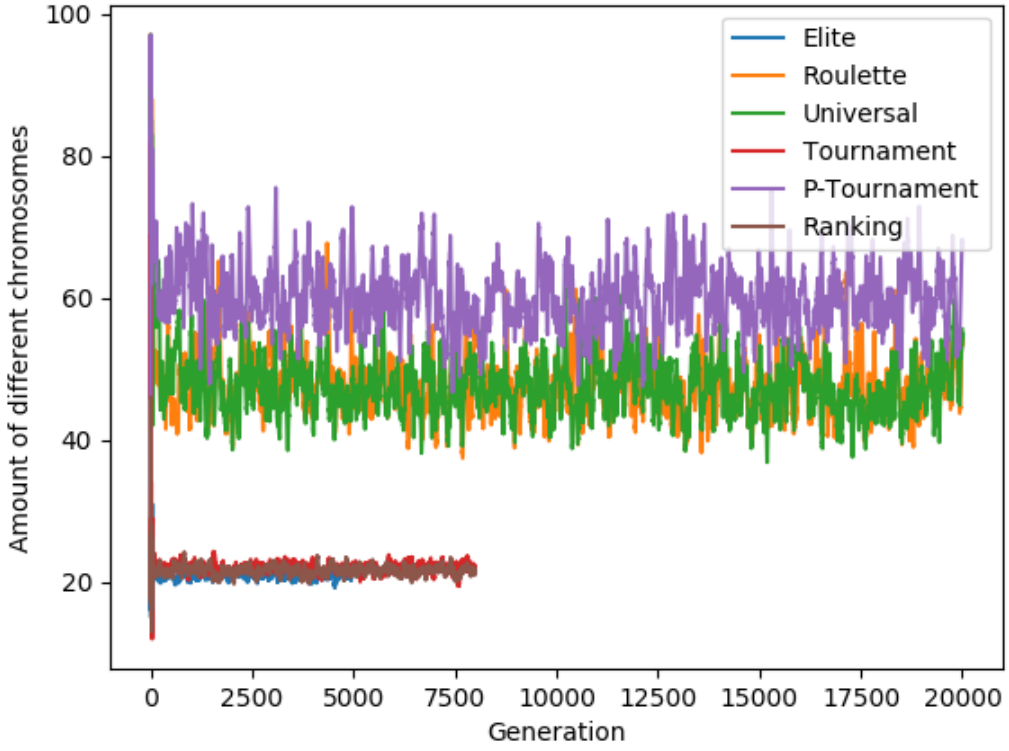
Elite
Roulette
Universal
Tournament
P-Tournament
Ranking
Amount of different chromosomes
100
80
60
40
20
0
2500
5000
7500
10000
12500
15000
17500
20000
Generation

Gráfico 21