Instituto Tecnológico de Buenos Aires

Sistemas de Inteligencia Artificial

Ingeniería en Informática

Algoritmos Genéticos

Tercer Trabajo Práctico Especial

Titular: Parpaglione, Cristina

Semestre: 2018A

Grupo: 4

Repositorio: https://bitbucket.org/itba/sia-2018-04

Fecha: 30 de mayo de 2018

Entrega: 30 de mayo de 2018 a las 15:00hs

Autores: Buscaglia, Matías Alejandro (#53551)

Costesich, Pablo Alejandro (#50109)

Delgado, Francisco (#57101)

Gonzalez, Lautaro Nicolás (#54315)

1. Introducción

2. Motor

El motor está compuesto de Darwin, el "pegamento.[en]tre las partes, e implementaciones de:

- Combiner
- Selector
- Replacer
- EvolutionaryTarget
- Mutator
- Species

La mayoría del motor hace uso extensivo de streams y paralelización, además de inmutabilidad, para lograr grandes velocidades y un buen aprovechamiento de los recursos computacionales.

2.1. Darwin

Separamos el motor de la implementación del juego en sí. El motor, llamado Darwin, define un pipeline de ejecución que llama a las distintas etapas de un algoritmo genético. Las mismas fueron definidas en base a criterios de inmutabilidad, por lo que el pipeline aprovecha el CPU al máximo paralelizando cada paso.

El pipeline tiene tres grandes etapas:

- Selección de Padres (getParents): se genera una lista en base al primer selector.
- Procreación: se genera un stream de parejas, las cuales son cruzadas, y sus hijos mutados.
- Reemplazo (mix): se genera una lista final a partir de las dos anteriores utilizando el algoritmo correspondiente y un selector.

2.2. Replacer

Los replacers son los encargados de definir el método en el que se obtendrán los padres y cómo se generará la descendencia. Para ello, implementan los métodos getParents y mix que usará Darwin.

2.2.1. NewGenerationReplacer

Implementa un reemplazo total de población. El factor de brecha generacional es 1 (100 %), y por lo tanto no realiza una selección de padres o de hijos.

2.2.2. KeepSomeReplacer

Implementa un reemplazo de brecha generacional k donde N - k padres sobrevivirán (seleccionados mediante el segundo Selector). Se le agregarán k hijos, que resultan de seleccionar padres con el primer Selector.

2.2.3. MixAllReplacer

Implementa un reemplazo de brecha generacional k donde, de la suma de padres e hijos, sobrevivirán N utilizando el segundo Selector. Los k hijos resultan de seleccionar padres con el primer Selector.

2.3. EvolutionaryTarget

Los targets son las condiciones de continuación del motor. Pueden ser por cantidad de iteraciones, estructura, y otras.

2.3.1. StructuralTarget

Para detectar la estructura de la población que puede sobrevivir una gran cantidad de generaciones, el StructuralTarget toma como parámetro el porcentaje de individuos inmortales y las generaciones que StructuralTarget recordará.

Para evaluar la cantidad de inmortales, tomamos los individuos de la generación actual que son distintos, y para cada uno buscamos el mínimo de veces que aparece en todas las generaciones (incluyendo la corriente). Esa lista de números significa la cantidad de individuos que sobrevivieron en todas las generaciones. Al sumar dicha lista, el resultado es la cantidad de individuos que sobrevivieron. Si el número de inmortales es menor al de target (porcentaje por tamaño de población), el motor continúa buscando soluciones.

2.3.2. OptimumTarget

Al igual que StructuralTarget, OptimumTarget se basa en una memoria generacional. Sin embargo, sólo almacena el desempeño promedio de todos los individuos por generación. El target calcula el máximo desempeño promedio y el mínimo para la memoria generacional. Si ocurre que la relación entre el mínimo y máximo desempeño es menor a la condición de corte, detiene el motor ya que el mismo se ha estancado en una "banda"de fitness.

3. Juego

3.1. Species (DNDCharacter)

El character fue definido como un objeto inmutable que precalcula todos los parámetros. El mismo es una agregación de la altura y los Items, que también son definidos inmutables (y referenciados desde un repositorio central para ahorrar memoria y ganar velocidad).

Tanto el array de items como la altura son considerados el cromosoma (único) de un character.

3.2. Combiners

Se definieron una serie de combiners, que dependen de los métodos de un-alelo, n-alelos y anular. Los mismos intercambian una parte, n partes o un segmento de los cromosomas entre padres para generar hijos.

El proceso además contempla la existencia de una función que, dada una probabilidad, decide si el par actual debe tener hijos o sobrevivir intacto.

3.3. Mutators

Similares a los combiners, estos generan con probabilidad uniforme o no uniforme mutaciones sobre los individuos. De la misma forma que los Combiners, contempla la existencia de una función que, dada una probabilidad, decide si el individuo actual debe o no mutar.

4. Conclusiones

4.1. Análisis de resultados

A continuación se exponen los aspectos más significativos del problema y sus resultados:

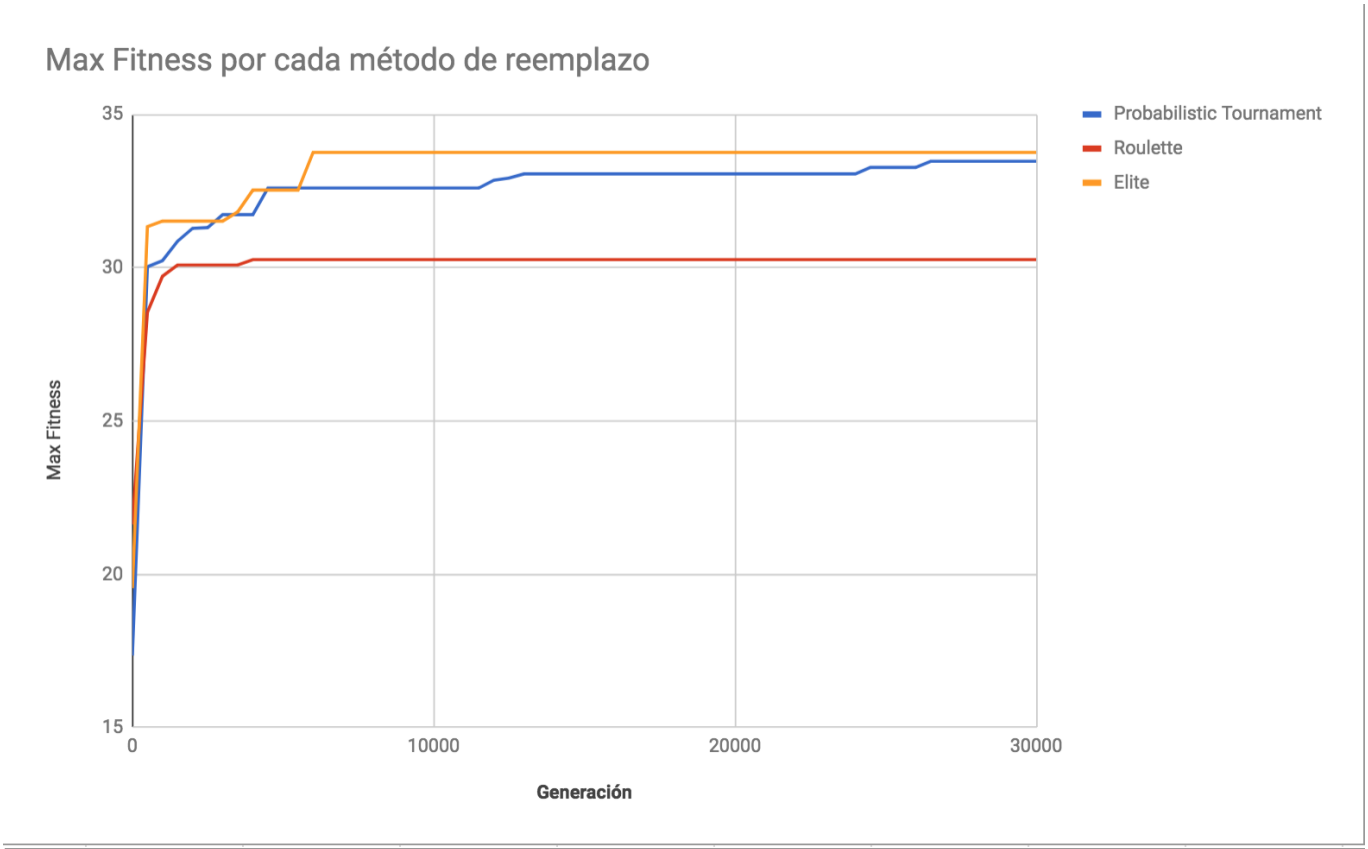
- Los métodos de cruza producen soluciones con fitnesses muy similares. Sin embargo, se observan leves diferencias en la varianza que producen sobre la población, esto es probable debido a que si bien todos los métodos modifican la misma cantidad de individuos, estos cruzan a los individuos de distinta manera pudiendo producir más o menos varianza en los hijos.

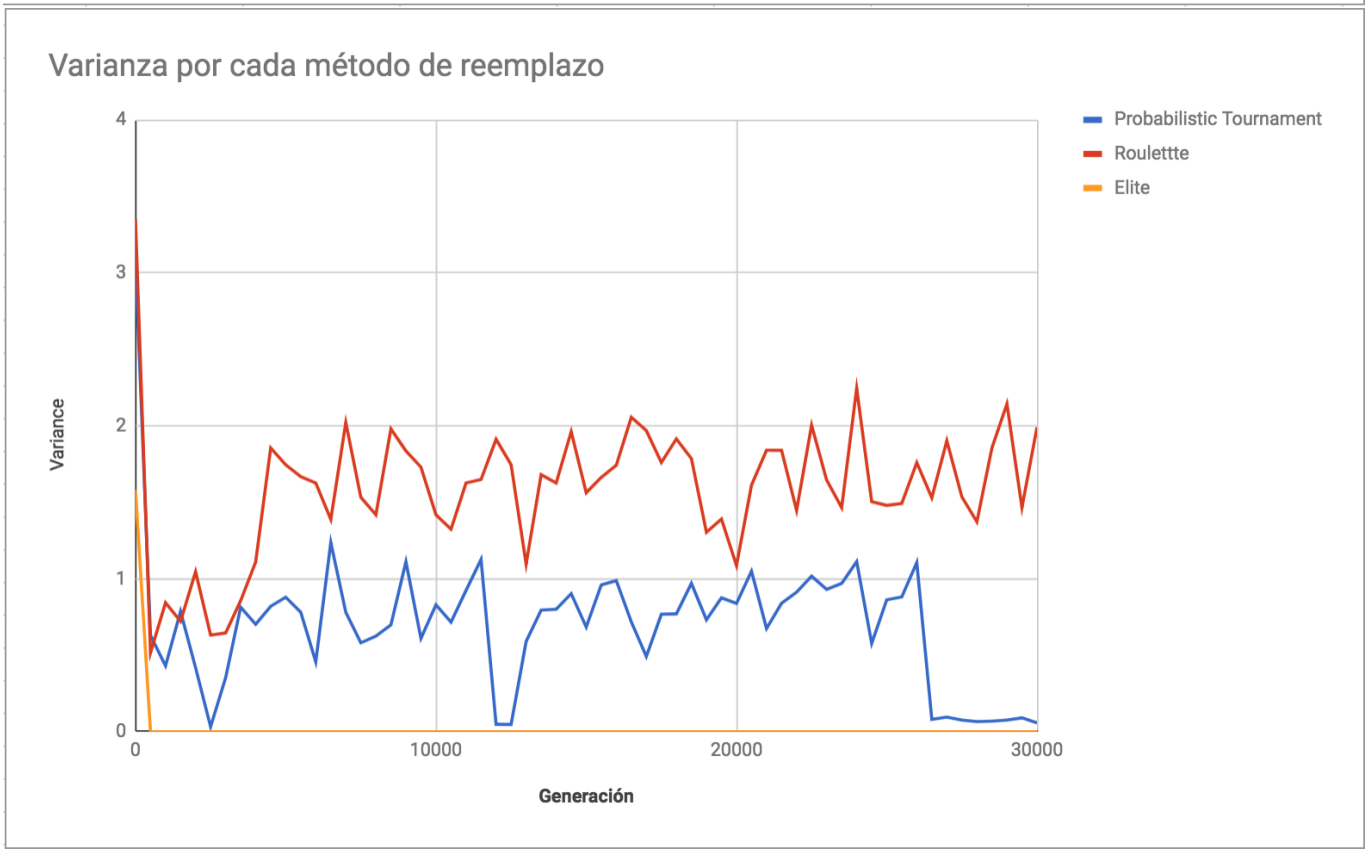
- El método de reemplazo que es claramente inferior es aquel que reemplaza totalmente toda la población. Entre los otros dos metodos se observa marginalmente mejor al método de reemplazo que selecciona N-K padres y pasa los hijos directamente.

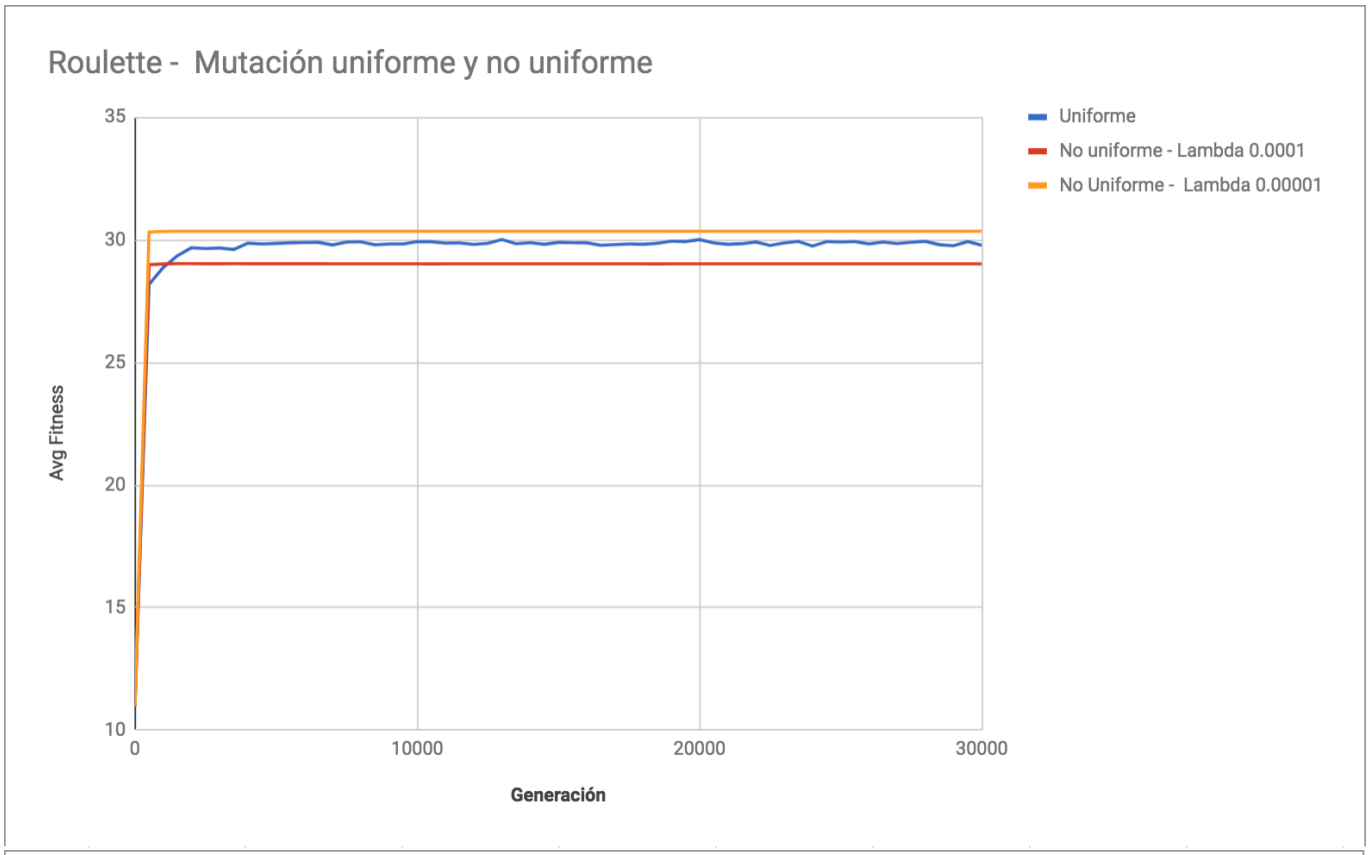
- Como era de esperarse la mutacion uniforme produce una varianza que no se correlaciona con las generaciones mientras que la mutacion no uniforme decrece a medida que avanzan las generaciones. Una posible mejora al trabajo es alterar los parametros de la función de probabilidad de mutación ya que observamos que la varianza se redujo muy rápidamente. Observamos una leve mejora en la cantidad de generaciones que se necesitan para llegar a la meceta de fitness, en donde la mutacion no uniforme posee esta leve ventaja sobre la uniforme.

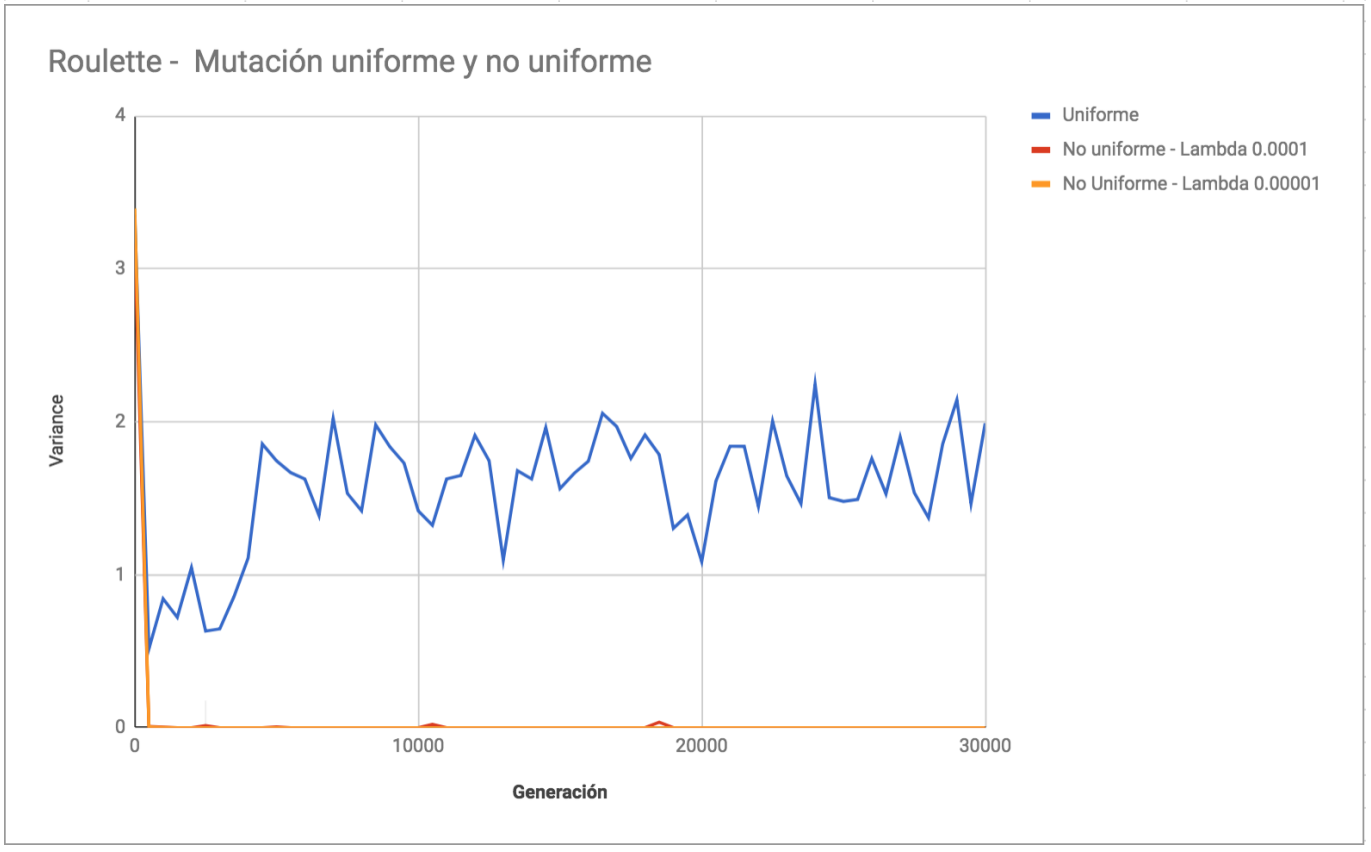
- Los métodos de selección tuvieron un desempeño similar excepto por ranking el cual no mostró mejoras en el fitness de la población. Con respecto a la varianza como es de esperarse todos los metodos producen medidas similares excepto por elite que logicamente al seleccionar solo los individuos con mayor fitness tiende a homogeneizar la población

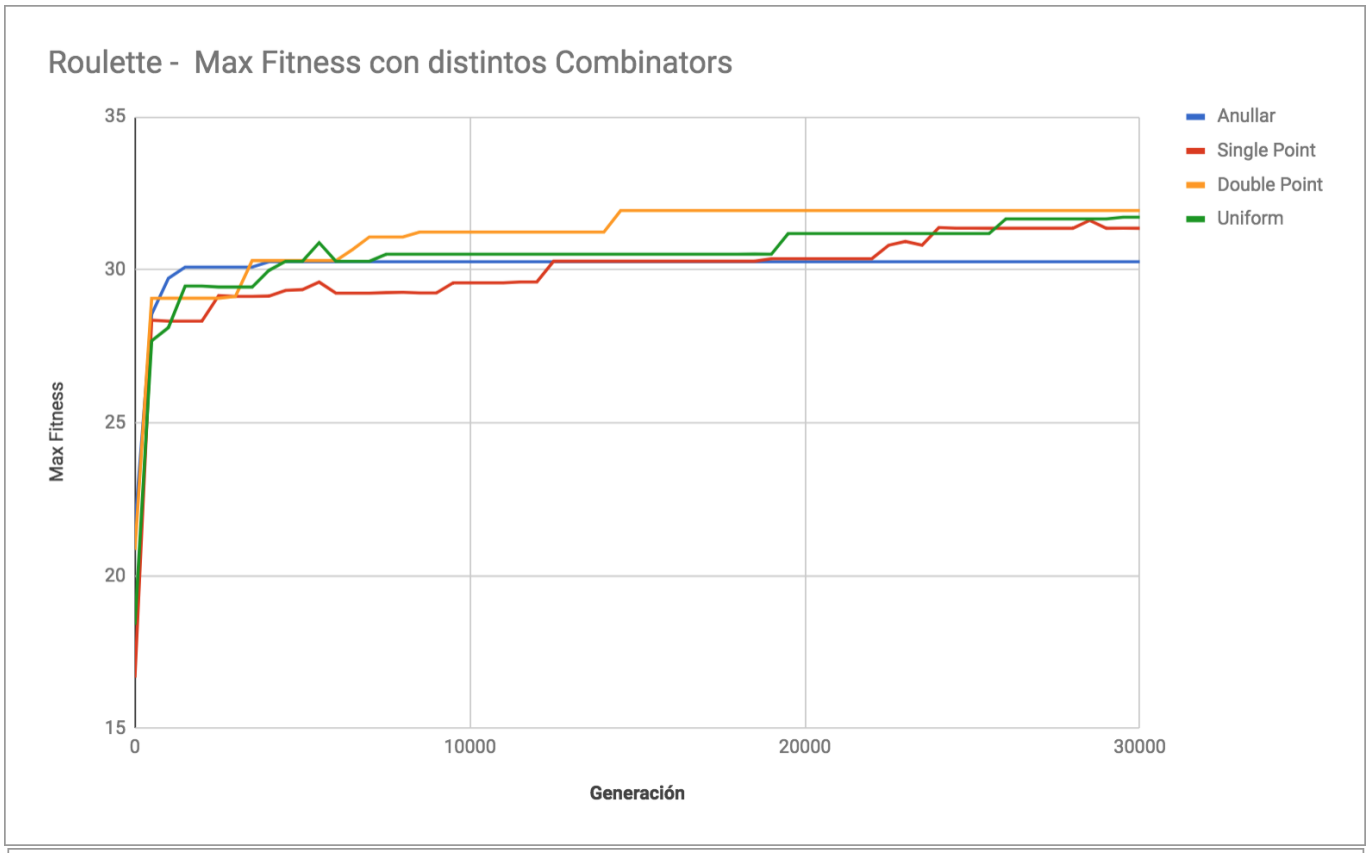
Apéndice

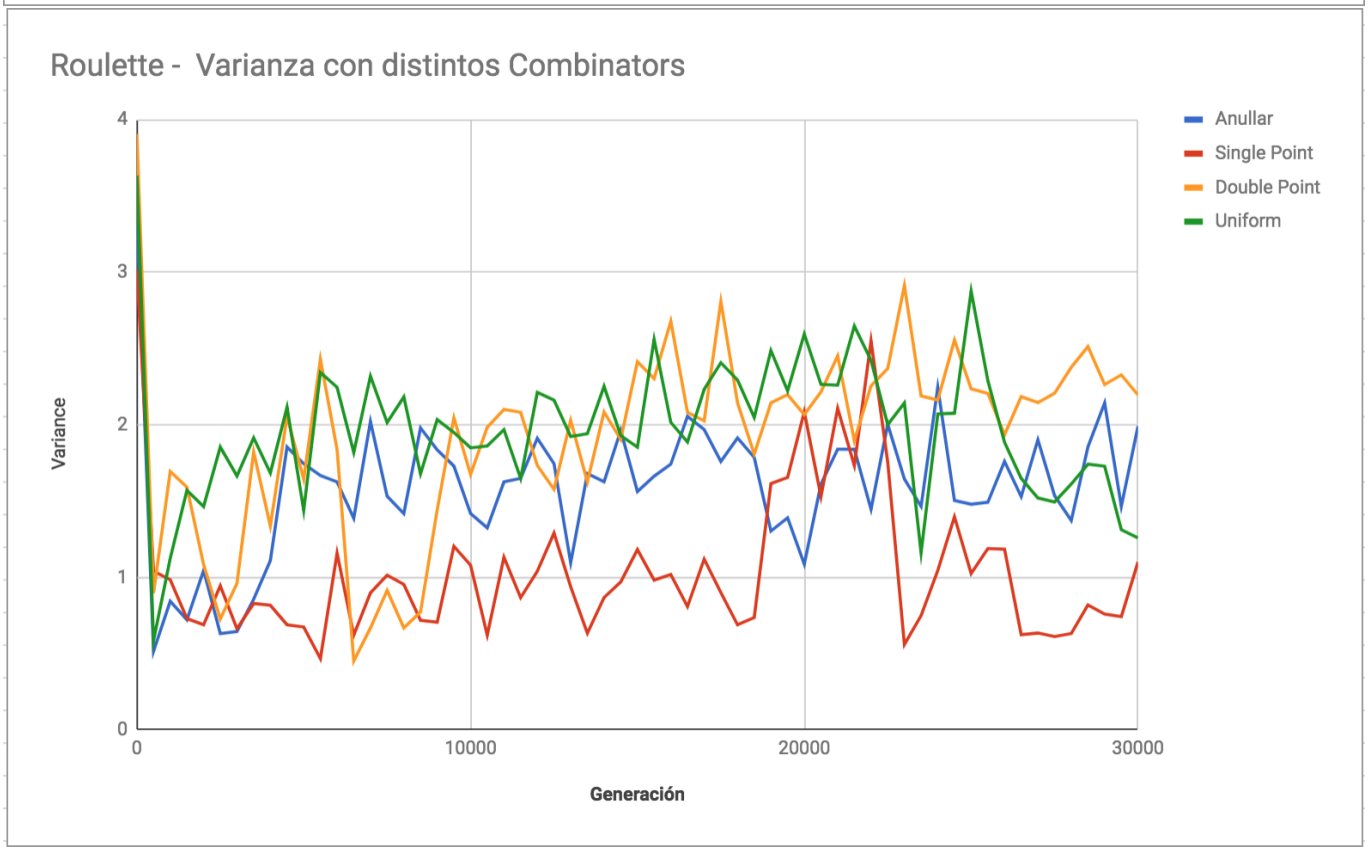

Max Fitness por cada método de reemplazo
Probabilistic Tournament
Roulette
Elite
Max Fitness
35
30
25
20
15
0
10000
20000
30000
Generación

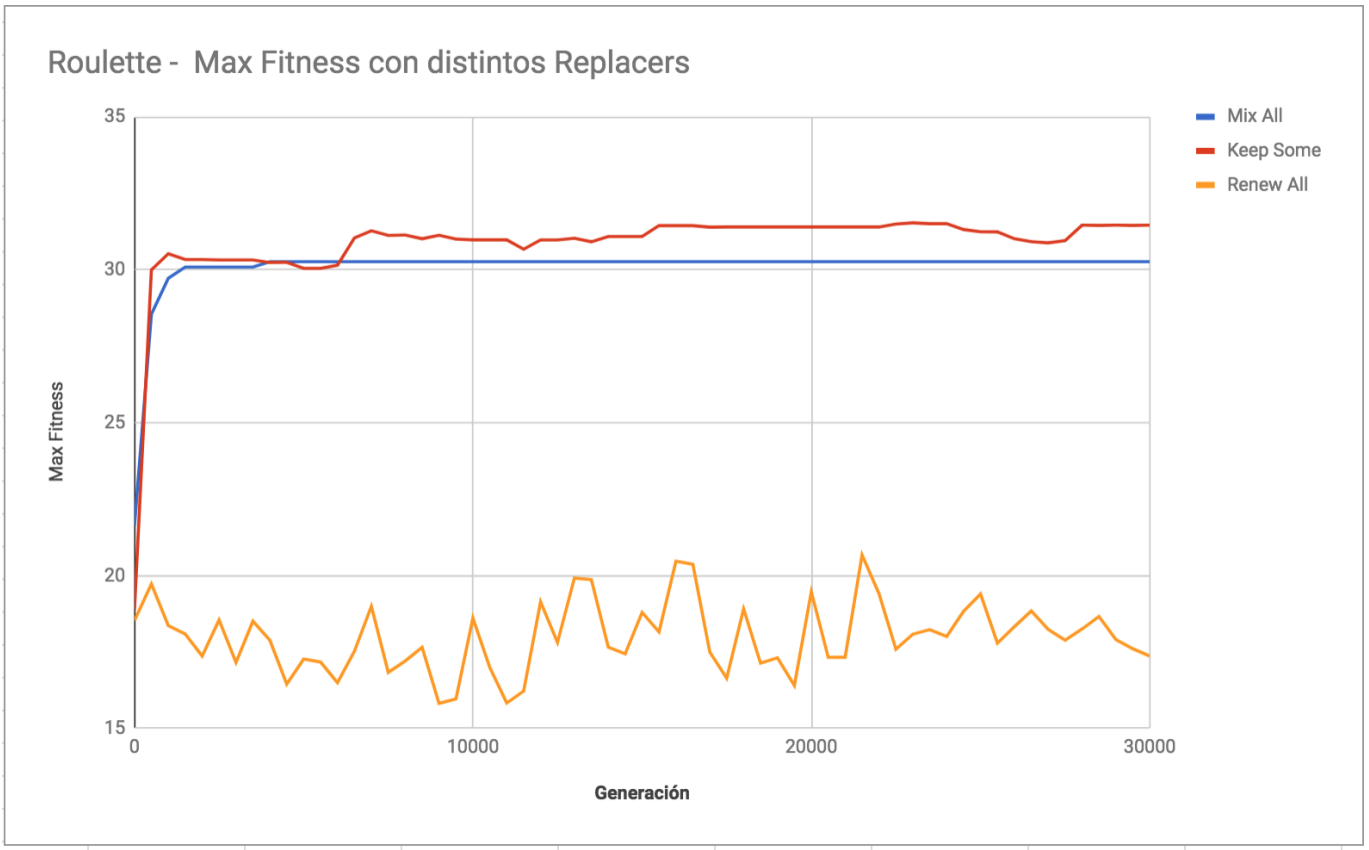

Varianza por cada método de reemplazo
Probabilistic Tournament
Roulettte
Elite
Variance
4
3
2
1
0
0
10000
20000
30000
Generación

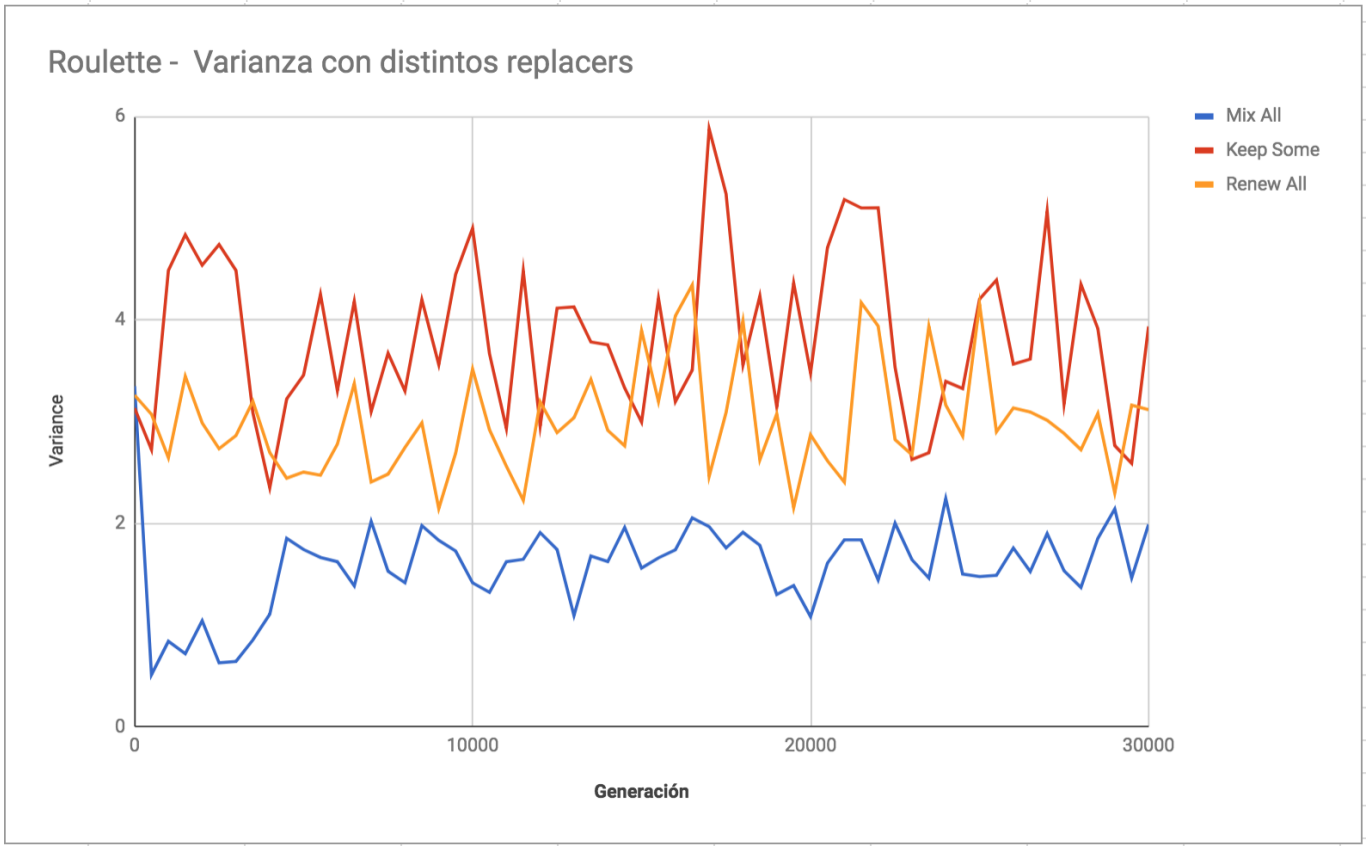

Roulette - Mutación uniforme y no uniforme
Avg Fitness
35
30
25
20
15
10
0
10000
20000
30000
Generación
Uniforme
No uniforme - Lambda 0.0001
No Uniforme - Lambda 0.00001

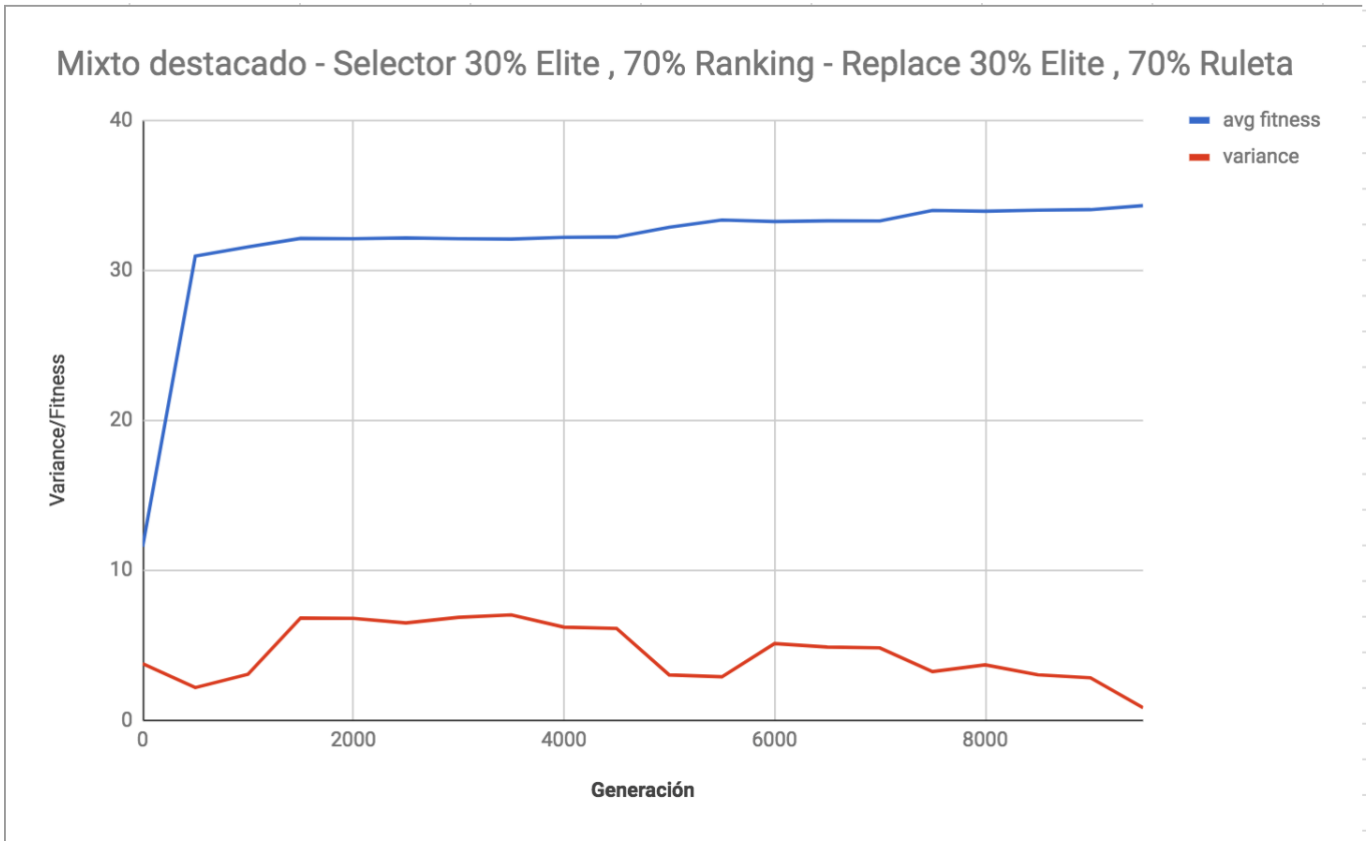

Roulette - Mutación uniforme y no uniforme
Variance
4
3
2
1
0
0
10000
20000
30000
Generación
Uniforme
No uniforme - Lambda 0.0001
No Uniforme - Lambda 0.00001


Roulette - Max Fitness con distintos Combinators
Anullar
Single Point
Double Point
Uniform
Max Fitness
35
30
25
20
15
0
10000
20000
30000
Generación


Roulette - Varianza con distintos Combinators
Anullar
Single Point
Double Point
Uniform
Variance
4
3
2
1
0
0
10000
20000
30000
Generación


Roulette - Max Fitness con distintos Replacers
Mix All
Keep Some
Renew All
Max Fitness
35
30
25
20
15
0
10000
20000
30000
Generación


Roulette - Varianza con distintos replacers
Mix All
Keep Some
Renew All
Variance
6
4
2
0
0
10000
20000
30000
Generación


Mixto destacado - Selector 30% Elite , 70% Ranking - Replace 30% Elite , 70% Ruleta
avg fitness
variance
Variance/Fitness
40
30
20
10
0
0
2000
4000
6000
8000
Generación