Sistemas de inteligencia artificial

(1er cuatrimestre 2018)

Trabajo Práctico 3

Algoritmos Genéticos

Julián Benítez 56283

Nicolás Marcantonio 56288

Eliseo Parodi Almaraz 56399

Índice

Introducción

Para este trabajo práctico desarrollamos un programa para calcular, mediante diversos algoritmos genéticos, la mejor configuración de items para un personaje de un videojuego de juego de rol. Para esto, tuvimos en cuenta su clase ya que la mejor conformación de los objetos depende de eso. En nuestro caso, analizamos el comportamiento del algoritmo para un personaje, Guerrero 2.

Implementación

Arquitectura

Se implementó en Java, utilizando la librería de Google GSON para parsear el archivo de configuración.
En cuanto a la estructura del código en sí, optamos por usar interfaces para todo a lo largo del codigo para tener una buena compatibilidad entre todos los diferentes componentes. Esto terminó sirviendo porque podíamos cambiar grandes partes del código sin que este afecte al funcionamiento del resto.

Se utilizó la dependencia JFreeChart para graficar en tiempo real la evolución del fitness generación a generación, para eso cada vez que pasa una generación se grafica el fitness más alto , más bajo y promedio de dicha generación.

En cuanto a la generación de individuos, nuestro primer approach fue leer los archivos .tsv y cargar por completo todos los ítems en memoria, para luego elegir de manera aleatoria los ítems para los n individuos a generar. Al estar usando los ítems de prueba, la eficiencia no se vió perjudicada por esta implementación.

Sin embargo, al momento de probar con los ítems reales, la memoria no daba a basto, y un experimento de tan sólo 400 generaciones tardaba más de 30 minutos. Es por eso que decidimos cambiar esto por “Reservoir Sampling”, que lo que hace es leer el archivo y con una probabilidad que depende de la cantidad de ítems que se quiere cargar en el momento y se retorna una cola de ítems seleccionados al azar con una cantidad más o menos similar a la previamente determinada. Cuando la cola se queda sin elementos, se vuelve a llamar a la función que rehace los pasos previos para conseguir más ítems al azar.

Métricas

Para medir la performance de las generaciones, usamos distintas métricas:

- Máximo fitness en la generación
- Fitness promedio de la generación
- Clones: cantidad de individuos que son iguales entre sí
- Diversidad: Cantidad de Fenotipos distintos en la población

- Fitness promedio de los padres

Condiciones de corte
Usamos la métrica de clones para cortar si hay una cantidad mayor a x clones como corte.

Cromosoma

El cromosoma está compuesto por seis elementos, armadura, botas, guantes, casco, arma y altura. Los primeros cinco elementos son ítems ya dados en un archivo, y la altura en el rango 1.3m y 2.0m. Este parámetro es diferente al resto ya que modifica los stats de ataque y defensa del personaje. Realizando los cálculos analíticos adecuados, que se pueden observar en la sección de demostraciones, en la sección 1 (fitness máximo), pudimos llegar a la conclusión de que hay un fitness máximo que se puede alcanzar analíticamente de **214,88.** Debido a que la única variable que está a nuestro alcance es la altura y que dependemos de los ítems que se nos otorgan en una base de datos, es probable que no se llegue a un resultado parecido, ya que es un resultado matemático y no contempla la realidad de los ítems que estamos manejando.

Análisis de resultados

Mutadores

El mutador no uniforme le llamamos “evolving” y toma un ratio de mutación inicial, uno final y una duración e interpola el ratio de mutación entre el ratio inicial y el final según qué generación está siendo evaluada. Esperamos que un alto índice de mutación nos evite converger en un mínimo, pero como vemos en el gráfico 8 que el promedio de fitness de las generaciones va convergiendo a un valor que no es el máximo a pesar de haber usado un valor relativamente alto (termina con 0.8) de mutación. En el caso del híbrido Elite-Ruleta, se ve que el promedio de fitness de las generaciones mejora lentamente a medida que transcurren. Esto se debe a que Elite siempre toma los mejores individuos dentro de cada generación y Ruleta altera un poco los genes que traspasan para que no queden estancados en una convergencia prematura.

Cruce

No vimos que el tipo de cruce influya mucho en los resultados que obtuvimos, ya que en los gráficos los 4 tipos de cruzas se comportan de la misma manera, por lo tanto optamos por usar cruza simple en nuestras pruebas, por cuestiones de eficiencia temporal.

Selectores y Reemplazos

Como tienen una función muy similar, los selectores y reemplazos están modelados de la misma manera en el código, y probando nos dimos cuenta que la combinación que mejores

resultados daba es un selector híbrido entre Elite y Ruleta. También probamos Squared, que es el mismo principio que Ruleta, solo que el fitness se eleva al cuadrado, así los de mejor fitness tienen más chances de pasar sus genes.
En el gráfico 7 se muestra como el modo simple y normal disminuyen lentamente su diversidad, al contrario que Complex, que inmediatamente la baja. Esto es por la cantidad de copias

Histogramas y selectores

Se corrieron 300 experimentos en simultáneo y se puso como target fitness 47. Luego se hizo un histograma de cuantas generaciones tardaron cada experimento en llegar a ese target. Estos gráficos se encuentran en el anexo (gráficos 3, 4, 5 y 6). Tournament funcionó muy bien según este parámetro pero eso fue porque el tamaño de población era muy chico relativo al tamaño del torneo y había muchos repetidos. Usamos un tamaño de población de 20 y elegimos 20 padres. Usamos mutación no uniforme de 1 a 0.8 a lo largo de 400 generaciones. Usamos un pequeño número de individuos para evitar finalizar demasiado rápido.

Anexo

Gráficos

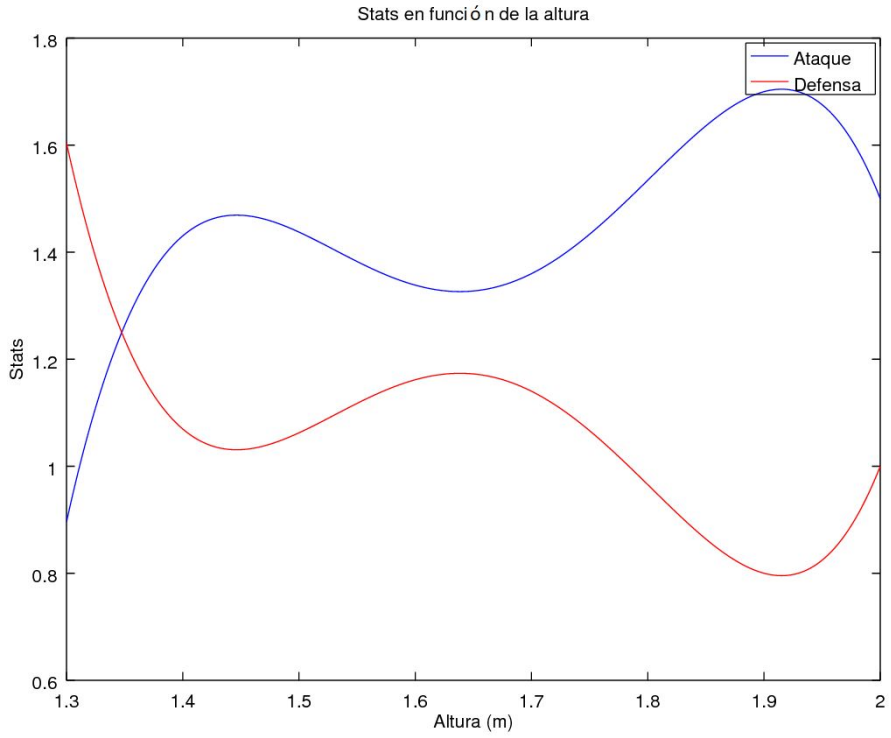

Gráfico 1: Stats en función de la altura.

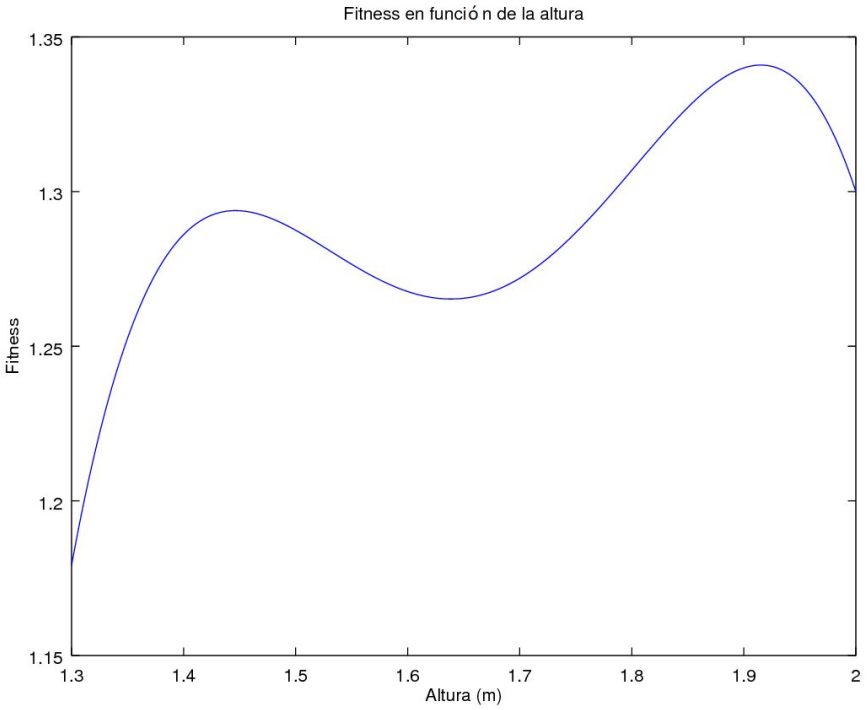

*Gráfico 2: Fitness en función de la altura, con (Agilidad + Pericia) * Fuerza = 1 y (Resistencia + Pericia) * Vida = 1. Básicamente, se estaría graficando: f(x) = 0,6 * ATM + 0,4 * DEM*

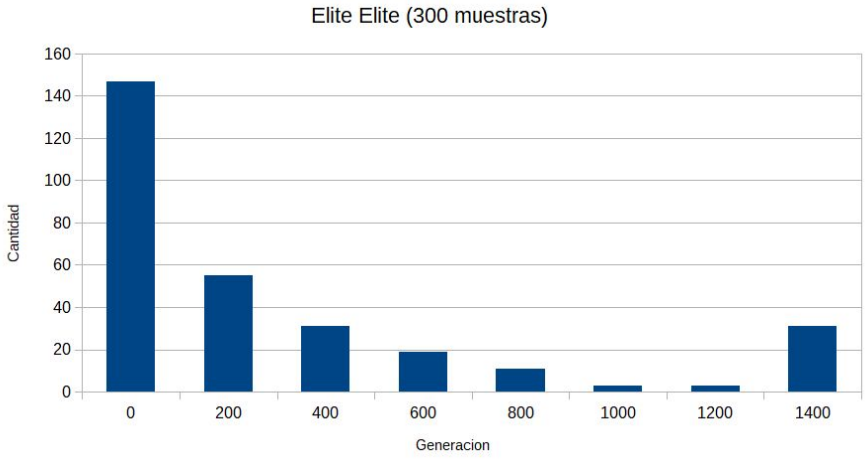

Gráfico 3: Histograma de cantidad de generaciones en llegar a un target fitness de 47 para elite/elite en selector/reemplazo

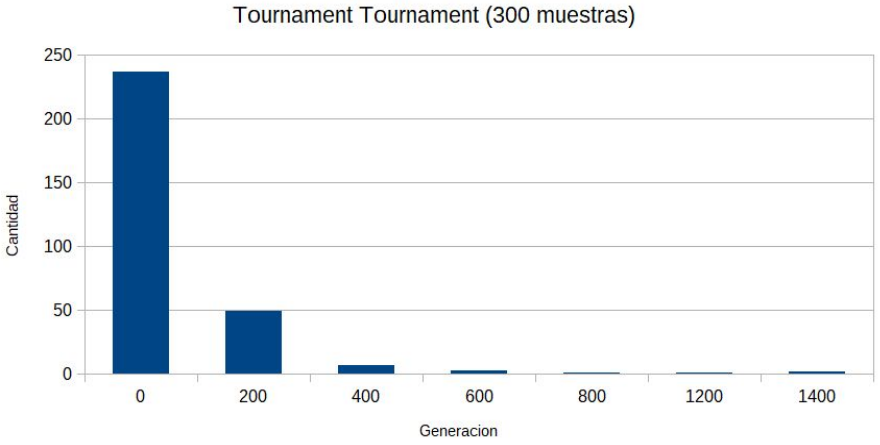

Gráfico 4: Histograma de cantidad de generaciones en llegar a un target fitness de 47 para Tournament/Tournament en selector/reemplazo

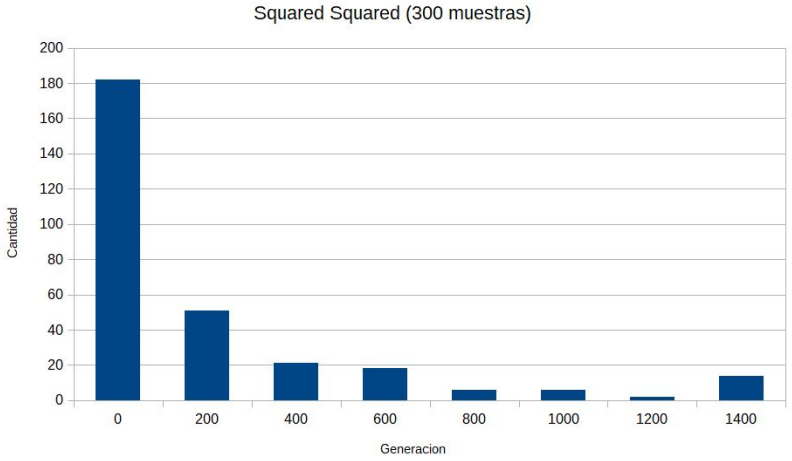

Gráfico 5: Histograma de cantidad de generaciones en llegar a un target fitness de 47 para el selector/reemplazo Squared

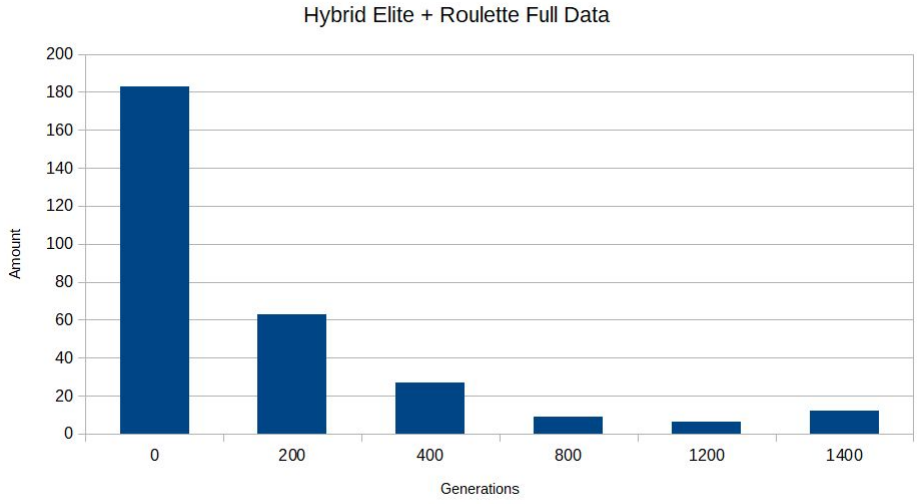

Gráfico 6: Histograma de cantidad de generaciones en llegar a un target fitness de 47 para hybrid/hybrid en selector/reemplazo

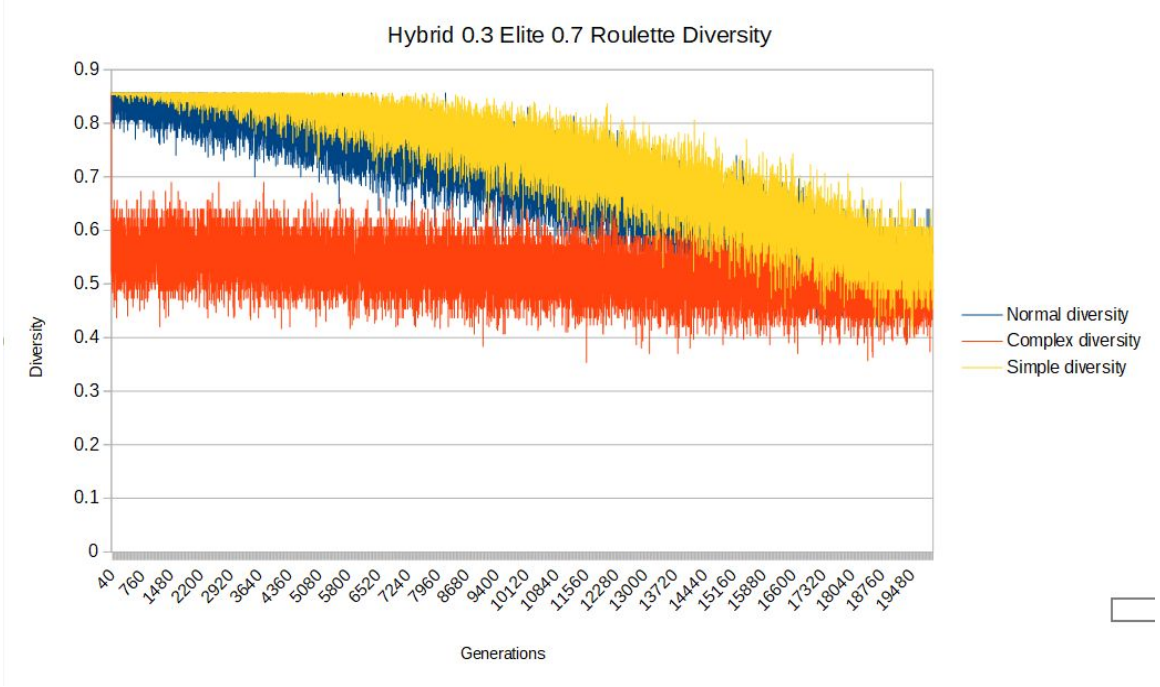

Gráfico 7: Gráfico de diversidad, cantidad de genes distintos en una población, para un experimento usando como selector elite y ruleta, para los métodos de reemplazo normal, complex y simple.

potenciales en Complex.

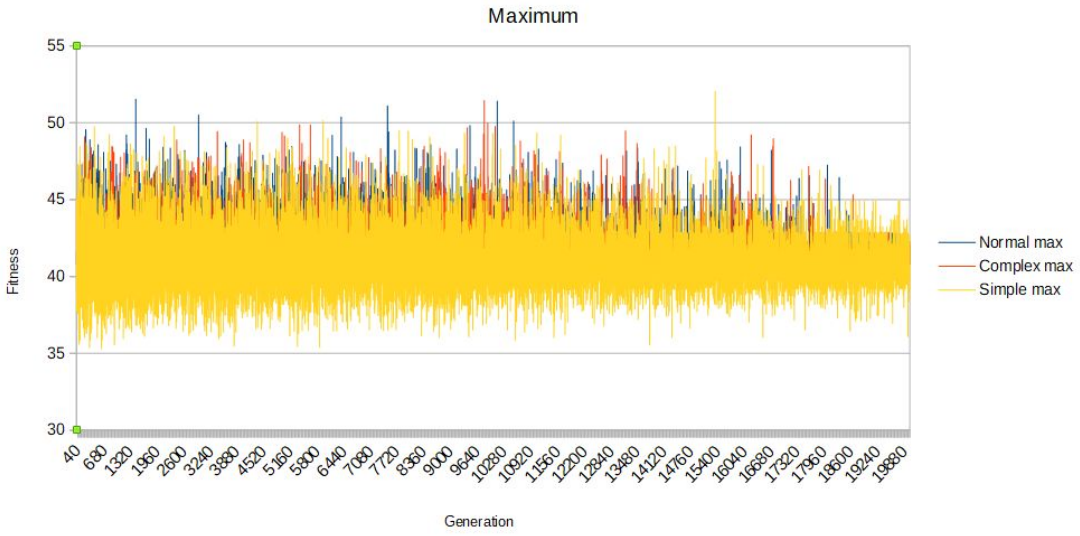

Gráfico 8: Gráfico de fitness máximo en una población para un experimento usando como selector elite y ruleta, para los métodos de reemplazo normal, complex y simple.

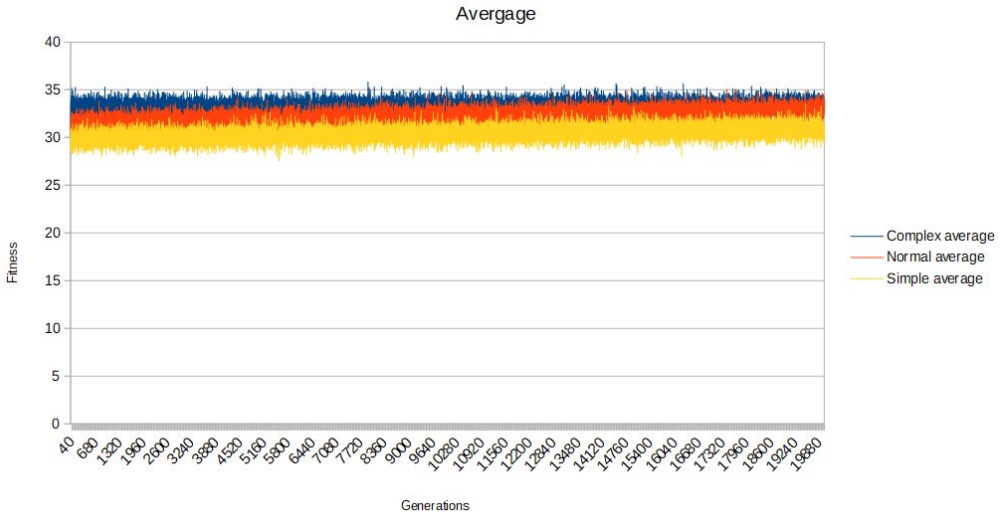

Gráfico 9: Gráfico de promedio de fitness para un experimento usando como selector elite y ruleta, para los métodos de reemplazo normal, complex y simple.

Demostraciones

1. Fitness máximo

Los stats están definidos con las siguientes fórmulas:

$$Fuerza_p = 100 * tanh(0{,}01 * \sum Fuerza_{item})$$

$$Agilidad_p = tanh(0{,}01 * \sum Agilidad_{item})$$

$$Pericia_p = 0{,}6 * tanh(0{,}01 * \sum Pericia_{item})$$

$$Resistencia_p = tanh(0{,}01 * \sum Resistencia_{item})$$

$$Vida_p = 100 * tanh(0{,}01 * \sum Vida_{item})$$

La función tangente hiperbólica tiene una imagen definida entre -1 y 1, esto quiere decir que está acotada superiormente por 1. Por lo tanto, los stats están acotados de la siguiente manera:

Fuerza ≤ 100
Agilidad ≤ 1
Pericia ≤ 0.6
Resistencia ≤ 1
Vida ≤ 100

Las funciones de ataque y defensa están definidas como:

$$Ataque = (Agilidad_p + Pericia_p) * Fuerza_p * ATM$$

$$Defensa = (Resistencia_p + Pericia_p) * Vida_p * DEM$$

Por lo tanto, reemplazando los valores máximos, podemos obtener que están acotadas de la siguiente forma:

Ataque ≤ 160 * ATM
Defensa ≤ 160 * DEM

Siendo nuestra función de fitness:

*f(x) = 0,6 * Ataque + 0,4 * Defensa*

Podemos reemplazar:

*f(x) ≤ 0,6 * 160 * ATM + 0.4 * 160 * DEM*
*f(x) ≤ 160 * (0,6 * ATM + 0.4 * DEM)*

Como se puede observar en el gráfico 2, el máximo de *(0,6 * ATM + 0.4 * DEM)* es **1.343**.
Por lo tanto, el máximo fitness es:

f(x) ≤ 160 * 1.343 = **214,88**