Sistemas de Inteligencia Artificial

Trabajo Especial:

Algoritmos genéticos

Instituto Tecnológico de Buenos Aires

Buenos Aires, Argentina

2018

Integrantes:

- Alan Arturo Hernández Gutiérrez alhernandez@itba.edu.ar 60438
- Kevin Chidiac kchidiac@itba.edu.ar 60431
- Romain Thibaut Marie Latron rlatron@itba.edu.ar 60440
- Ramiro Hernán Olivera Fedi rolivera@itba.edu.ar 56498

Profesores:

- María Cristina Parpaglione
- Alan Pierri

Introducción

En el presente trabajo, se describe el desarrollo de un motor de algoritmos genéticos con el fin de obtener las mejores configuraciones de personajes de un juego de rol. El juego consiste en personajes que tienen cierta clase, ciertas propiedades, y cierto equipamiento. El objetivo es lograr la mejor configuración de ellas para optimizar el desempeño del personaje en el juego.

Con el fin de simplificar el análisis de los resultados, nuestro trabajo buscará optimizar el desempeño de un personaje particular del juego: El **Defensor 2**. No obstante, se desarrollarán las clases necesarias para que el motor funcione con cualquier tipo de personajes.

Implementación del proyecto

El motor se divide en distintos paquetes de Java. Cada paquete incluye una interfaz y un conjunto de clases con implementaciones particulares de cada método. En primer lugar, aquellos paquetes relacionados con el funcionamiento de los algoritmos.

- Selection
- Replacement
- Mutation
- Crossover
- Stop condition

Por otro lado, también se implementaron paquetes del modelado del problema

- Item
- Character

Además se implementaron distintas clases de utilidades para graficar, leer los archivos de items y el de configuración. El archivo de configuración permite la parametrización de las clases de los métodos y sus argumentos.

Finalmente, la clase `AlgoritmosGenéticos` representa el *main* de nuestro motor y contiene la estructura básica del algoritmo. Decidimos utilizar un approach funcional para nuestro motor, por lo que el algoritmo se reduce básicamente a la línea:

```
chromosomes =
reemplazar.apply(mutation.apply(recombinar.apply(seleccionar.apply
(chromosomes))), chromosomes);
```

Esta línea se ejecutará una vez tras otra generando nuevas generaciones, hasta que se cumpla la condición de corte escogida. La idea es que cada uno de los métodos reciba una lista de cromosomas, y devuelva una lista nueva con las transformaciones aplicadas. A

excepción del método de reemplazo, que recibe la lista de cromosomas transformados (A través de la mutación, cruza y selección) y la lista de cromosomas de la generación anterior para generar una nueva generación de acuerdo al método de reemplazo seleccionado.

Implementación del modelo

Se implementó una clase `Chromosome` que referencia al `Character` que se está analizando. Cada cromosoma consiste entonces del personaje, la lista de `Items` y la altura del personaje. De esta forma, cada generación consiste de una lista de cromosomas cada uno con sus respectivas configuraciones.

Desarrollo y análisis de resultados

En primer lugar, es necesario definir la forma en la que se calculará el desempeño de cada personaje (cromosoma). Nótese que los valores que se expresan a continuación representan aquellos válidos para el cálculo del fitness para el **Defensor 2**.

$$Fitness = 0.1 * Ataque + 0.9 * Defensa$$

$$Ataque = (Agilidad + Pericia) * Fuerza * ATM$$
$$Defensa = (Resistencia + Pericia) * Vida * DEM$$

$$ATM = 0.5 - (3h-5)^4 + (3h-5)^2 + h/2$$
$$DEM = 2 + (3h-5)^4 - (3h-5)^2 - h/2$$

$$Fuerza = 100 * \tanh(1.1 * 0.01 * \sum Fuerza_{item})$$
$$Agilidad = \tanh(0.8 * 0.01 * \sum Agilidad_{item})$$
$$Pericia = 0.6 * \tanh(0.8 * 0.01 * \sum Pericia_{item})$$
$$Resistencia = \tanh(1.1 * 0.01 * \sum Resistencia_{item})$$
$$Vida = 100 * \tanh(1.1 * 0.01 * \sum Vida_{item})$$

Donde `h = [1.3, 2]` es la posible altura del personaje, y las variables con `item` como subíndice representan las características brindadas por los items que usa el personaje.

Una vez definida la forma en la que se *calificará* el desempeño de cada personaje, pensamos comenzar analizando algunos detalles que saltan a la vista. Por ejemplo, podemos graficar el `ATM` y el `DEM` en función de la altura dado a que son funciones de una sola variable, y por lo tanto fáciles de graficar, y al hecho de que son factores directos del fitness.

Dado que la defensa representa el 90% del desempeño de nuestro personaje, esperamos que la altura del personaje esté bastante cercana al máximo del `DEM`. De esta forma, la *Figura 1* nos indica que la altura óptima de nuestro personaje rondará 1.3 metros.

Comenzamos entonces a probar distintos algoritmos genéticos con diferentes parámetros en busca de la mejor configuración de items y altura para nuestro personaje.

Partimos de las recomendaciones de la cátedra para el establecimiento de parámetros en algoritmos genéticos:

$$N = [20, 200]$$
$$Mutacion_{prob} = [0.001, 0.01]$$
$$Cruza_{prob} = [0.6, 0.95]$$
$$K = [N * 0.6, N]$$

Siendo `N` la población, y `K` la cantidad de cromosomas seleccionados en cada iteración. Con la finalidad de simplificar el análisis decidimos fijar el método de reemplazo en el `K-mutated`, o método de reemplazo 2; la población en 200 cromosomas; la cantidad de seleccionados en 140, es decir, un generation gap del 70%; y la condición de corte en un corte por contenido, es decir, detener el algoritmo luego de que el mejor desempeño de la generación no mejore por 10.000 generaciones.

Comenzamos con una cruza en un punto con una probabilidad del 80%, una mutación uniforme con una probabilidad del 10%, y un método de selección de Ruleta. El método de reemplazo cuenta con la misma selección.

La evolución del mejor fitness por cada generación puede verse en la *Figura 3*. Con estos parámetros logramos un fitness de `51.63` luego de 39,000 iteraciones.

Para seguir realizando pruebas decidimos mantener los mismos parámetros y cambiar únicamente el método de selección por uno de Elite. Como puede verse en la *Figura 4* Se alcanzó un fitness de `55.42` luego de 52,000 iteraciones.

Si bien el fitness mejoró, podríamos estar en un caso de convergencia prematura. Es por esto que decidimos utilizar un método de selección mezclada entre Boltzmann (Con una temperatura inicial de 400 grados) y Elite. Manteniendo los mismos parámetros, le otorgamos un peso de 80% a Boltzmann y uno de 20% a Elite para evitar la convergencia prematura. Tras haber realizado los cambios podemos ver que el fitness alcanza `54.70` luego de 50,000 iteraciones (*Figura 5*).

Aunque el mejor fitness fue algo menor que cuando se utilizó el método de Elite por sí solo, al incluir una parte de Boltzmann nos aseguramos que no se caiga en un caso de convergencia prematura.

Dado que los resultados con este método de selección fueron positivos, decidimos fijar ese método y cambiar los parámetros. Tomamos entonces una cruza uniforme con una probabilidad del 50% y, una mutación uniforme con una probabilidad del 10%.

Aplicando el algoritmo con los parámetros descritos se obtienen los resultados de la *Figura 6* y un fitness de `56.48` luego de 54,000 iteraciones.

Vale mencionar que en todos los casos la altura converge a 1.3 metros como se había supuesto en el apartado anterior.

Conclusiones

Habiendo probado con distintos parámetros y métodos, encontramos que los mejores resultados se obtienen utilizando una combinación de métodos de selección, en el que el método de selección Elite juega un rol importante. Mezclarlo con otro método como Boltzmann (Implementado con Ruleta), no solo mejora el rendimiento cuando los parámetros son los adecuados, sino que evita el problema de convergencia prematura.

Finalmente, la mejor configuración encontrada para el Defensor 2 es la siguiente:

Item	Id
Botas	801166
Casco	516677
Armadura	96822
Guantes	382596
Arma	37856
Altura	1.300 metros

Anexo

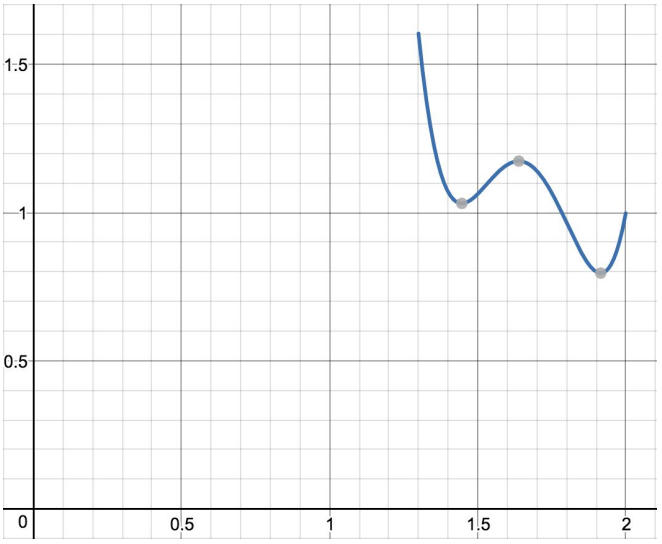

Figura 1 - DEM en función de la altura

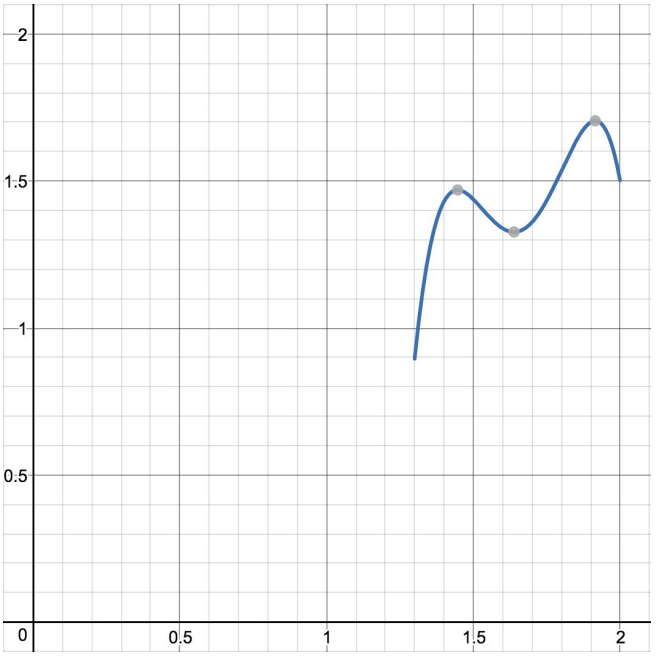

Figura 2 - ATM en función de la altura

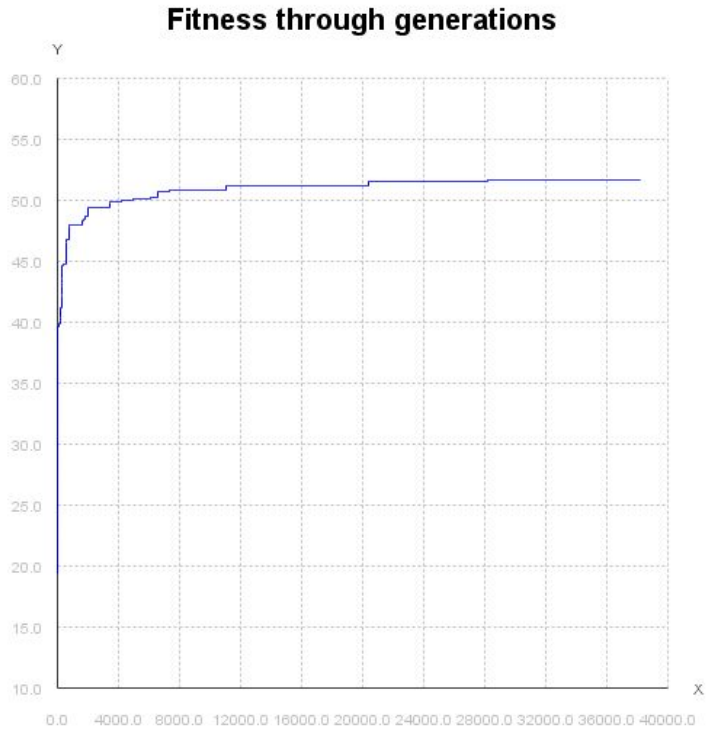

Figura 3 - Fitness a través de las generaciones con el método Ruleta.

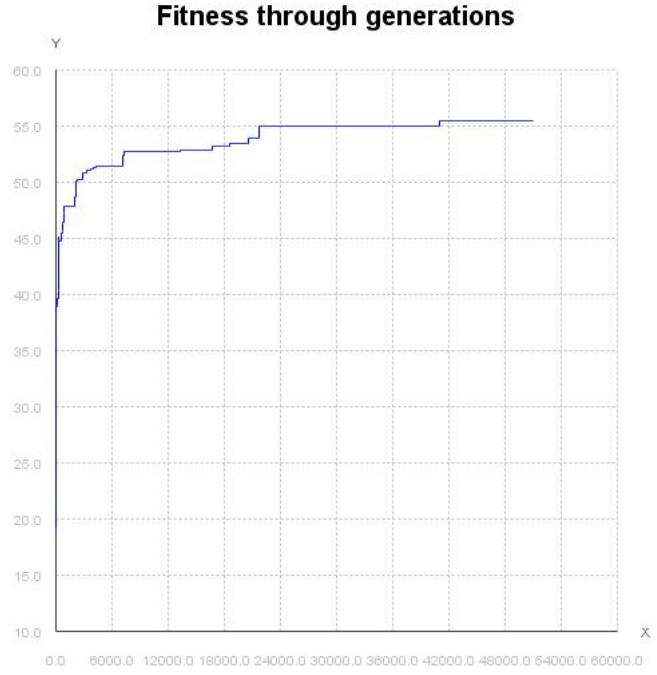

Figura 4 - Fitness a través de las generaciones con el método Elite.

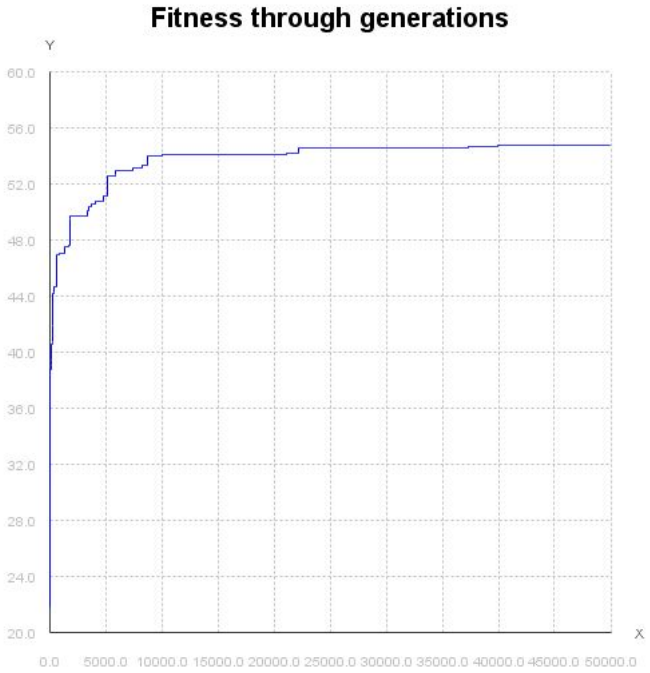

Figura 5 - Fitness a través de las generaciones con el método 0.8 Boltzmann + 0.2 Elite.

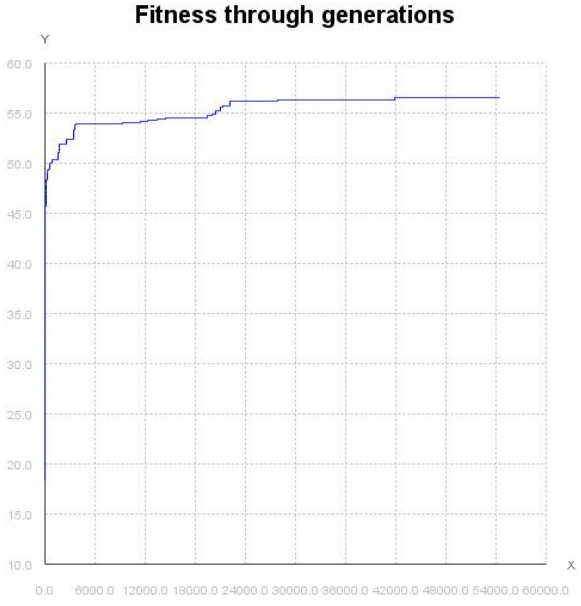

Figura 6 - Fitness a través de las generaciones con el método 0.8 Boltzmann + 0.2 Elite con nuevos parámetros.