EvoFighter: aplicación de algoritmos evolutivos a juegos de lucha 1 vs 1*

Noelia Escalera[1,*], Antonio Miguel Mora[1], Pablo García-Sánchez[2]

[1]Depto. Teoría de la Señal, Telemática y Comunicaciones. ETSIIT-CITIC, Universidad de Granada.
[2]Depto. Arquitectura y Tecnología de Computadores. ETSIIT-CITIC, Universidad de Granada.

Abstract

This paper presents a preliminary study on the application of evolutionary algorithms for the optimisation of the behaviour of autonomous agents for 1 vs 1 fighting games. Different evolutionary schemes have been proposed and quite promising results have been obtained, which leave room for obtaining very competitive agents following this methodology.

Keywords

Juego de lucha 1 vs 1, Non-Player Character, Bot, Algoritmos Genéticos, Optimización,

1. Introducción

Un juego de lucha es un enfrentamiento 1 vs 1 en el que los jugadores (personajes) intentan reducir la barra de salud del adversario a base de puñetazos, patadas e incluso armas blancas en algunos juegos. Los jugadores normalmente también pueden hacer movimientos especiales, que son más complicados de ejecutar, pero que pueden reducir la salud del rival en una cantidad mayor. La inteligencia artificial (IA) en este tipo de juegos tiene un gran hándicap en comparación con otros géneros, ya que deben tomar decisiones muy rápidas, dado el gran dinamismo de los movimientos de los rivales. En este trabajo, hemos implementado un *non-player character* (NPC o bot) con el objetivo de vencer a cualquier rival, sea un jugador humano u otro NPC, en el contexto de una IA. Para ello, un experto humano ha diseñado un bot competitivo que tiene como motor de IA un conjunto de reglas dependiendo de algunas condiciones, umbrales y pesos. Luego se han optimizado estos valores mediante diferentes esquemas de Algoritmos Genéticos (AG) [1].

Los juegos de lucha han sido un área de investigación muy prolífica, en la que se han propuesto muchos diferentes enfoques: bots basados en scripts [2, 3], *Monte-Carlo Tree Search* [4] y, recientemente, los de aprendizaje profundo por refuerzo (*Deep Reinforcement-Learning*) [5, 6]. También se han aplicado algoritmos evolutivos en este ámbito [7, 8], aunque, hasta donde

I Congreso Español de Videojuegos, December 1–2, 2022, Madrid, Spain

*Este trabajo ha sido financiado en parte por los proyectos PID2020-113462RB-I00 (ANIMALICOS) y PID2020-115570GB-C22 (DemocratAI::UGR) del Ministerio español de Economía y Competitividad, así como los proyectos P18-RT-4830 y A-TIC-608-UGR20, financiados por la Junta de Andalucía.

*Corresponding author.

escaleranm@gmail.com (N. Escalera); amorag@ugr.es (A.M. Mora); pablogarcia@ugr.es (P. García-Sánchez)

0000-0003-1603-9105 (A.M. Mora); 0000-0003-4644-2894 (P. García-Sánchez)

sabemos sólo un número muy reducido de estudios, como [9], se han centrado en enfoques evolutivos para la IA.

De modo que este trabajo presenta un estudio preliminar sobre las mejoras en el comportamiento y rendimiento obtenidas mediante la aplicación de AGs sobre un motor de IA de un agente/bot para videojuegos de lucha. Puede considerarse como un primer paso en nuestra investigación en este dominio, dadas nuestras experiencias previas y exitosas en otros juegos [10, 11, 12].

Los agentes obtenidos, uno por cada variante del AG, han sido probados contra un conjunto de rivales, algunos ya pre-implementados y otros creados por otros autores de anteriores ediciones de la competición. Hemos considerado un simulador llamado FightingICE[1], que es el utilizado en la competición internacional de IA de juegos de lucha (FGAIC). FightingICE también ofrece un marco donde los investigadores pueden desarrollar y probar sus propios agentes. Además incluye algunos bots prefabricados que pueden utilizarse para luchar contra (y probar) nuestros propios agentes (véase la Figura 1).

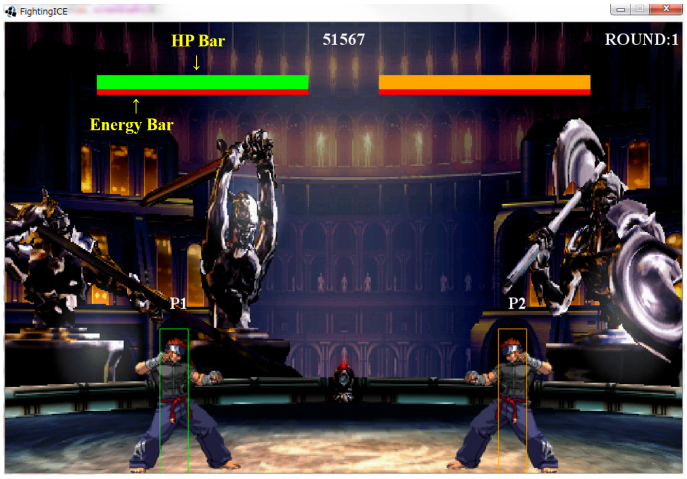

Figura 1: Captura de ejemplo de FightingICE

2. Agente Implementado

Para este estudio hemos considerado el modo estándar de salud/tiempo en el que tenemos que reducir la barra de salud del oponente a 0 o reducirla más de lo que se reduzca la de nuestro bot, en un tiempo limitado por combate. Las entradas para los bots pueden ser tres botones (Golpe, Patada y Especial) o una de las 9 direcciones en un pad de control o joystick. Hemos considerado el mismo personaje para todos los agentes, llamado Zen. Se ha implementado una versión modificada del bot de Mizuno [13], llamado *MizunoAI*. Éste fue presentado en la FGAIC de 2014, y se basa en la combinación de dos técnicas: El bot alterna entre las decisiones

[1]https://www.ice.ci.ritsumei.ac.jp/ ftgaic/index-2.html

basadas en kNN y el comportamiento basado en un sistema difuso, que considera valores aleatorios para elegir su funcionamiento. La optimización considera 10 parámetros: distancias al enemigo asociadas a diferentes ataques ($dist_{THROW}$, $dist_{AIR}$, $dist_{STAND}$, $dist_{FA}$), distancia para aproximarse al rival ($dist_{AVAN}$), niveles de salud del agente asociadas a ataques ($salud_{FB}$, $salud_{BB}$, $salud_{DFB}$, $salud_{DFA}$, $salud_{AIR}$).

Este bot genético ha sido implementado en Java e interactúa con el framework FightingICE a través de diferentes archivos externos. El AG optimiza el conjunto de parámetros de los que depende el motor de IA. Así, cada individuo es un vector de 10 valores que "modela" un comportamiento para el agente, dependiendo de los valores que tenga. Cada individuo se evalúa en un conjunto de 6 combates de juego contra dos posibles rivales. Tras estos combates, se calcula un valor de fitness como la media de $tiempo\_restante * (salud_{agente} - salud_{rival})$ de los enfrentamientos realizados.

Se han aplicado dos enfoques diferentes de AG:

- *Estacionario*: se seleccionan y enfrentan cuatro individuos, y sólo los ganadores serán los padres. Se crean dos descendientes que reemplazan a otros en cada generación, si son mejores que sus padres. Se considera cruce uniforme y mutación aleatoria en un gen. Con este enfoque se busca una alta presión selectiva.
- *Generacional*: en el que toda la población puede ser reemplazada. Se realiza un torneo binario que considera todos los individuos de la población (1 vs 1), de modo que los mejores en sus respectivos combates sobrevivirán. Al igual que antes, se considera cruce uniforme y mutación aleatoria de un gen. La población será la descendencia generada + la mitad de los individuos al azar. Así, buscamos un alto factor de exploración con este enfoque. Hemos añadido también elitismo, por lo que, los mejores individuos siempre sobrevivirán.

3. Experimentos y Resultados

La configuración considerada en los experimentos es: 20 generaciones, 16 individuos en la población, probabilidad de cruce = 60 %, probabilidad de mutación 10 %, número de combates para calcular el fitness = 6, salud de los jugadores = 300. Se han utilizado dos rivales diferentes para calcular el fitness: Dora (un agente avanzado de la literatura) y BCP (un bot muy agresivo también del estado del arte). Hemos realizado varias ejecuciones con los diferentes esquemas de AG y utilizando diferentes rivales en el cálculo del fitness. Tras esta fase de optimización, hemos elegido al mejor individuo de cada experimento enfrentando al mejor de cada ejecución en un torneo contra el resto, siendo finalmente seleccionado el ganador de un mayor número de combates.

A continuación, los agentes elegidos (los mejores) han luchado contra otros bots. En la Tabla 1 se puede ver un resumen del porcentaje de victorias de cada bot obtenido tras una serie de 9 combates. El porcentaje de victorias representado es el obtenido por el agente de la fila contra el de la columna.

Aunque no hemos conseguido obtener tasas de victoria notables, la Tabla 2 ilustra que hay una cierta mejora de los resultados con respecto al agente base. Los resultados muestran que, aunque la optimización no ha sido suficiente para ganar un gran número de combates, nos ha

	MizunoAIOrig	BCP	Dora	Thunder
MizunoAIOrig	-	0 %	0 %	0 %
MizunoAIEV-DoraBCP-Elitism	0 %	22.22 %	0 %	0 %
MizunoAIEV-DoraBCP-NoElitism	0 %	11.11 %	0 %	0 %
MizunoAIEV-BCP-Elitism	0 %	0 %	0 %	0 %
MizunoAIEV-BCP-NoElitism	33.33 %	0 %	0 %	0 %
MizunoAIEV-Dora-Elitism	33.33 %	0 %	0 %	0 %
MizunoAIEV-Dora-NoElitism	33.33 %	0 %	0 %	0 %

Tabla 1
Porcentaje de victorias de cada agente contra bots del estado del arte. Los nombres de las filas pertenecen a nuestros agentes optimizados (EV) indican el rival en el cálculo del fitness y el esquema de AG utilizado.

permitido mejorar prácticamente todos los resultados. *MizunoAIEV-DoraBCP-Elitism* ha sido la configuración con mejores resultados. Es interesante señalar que el hecho de que BCP obtenga diferencias de salud tan negativas está relacionado con su comportamiento, es decir, es un agente que una vez que consigue su objetivo (encerrar al oponente), vence con facilidad, por lo que en sus victorias gana por una gran diferencia. También destacamos el hecho de que normalmente hemos obtenido mejores resultados con las configuraciones elitistas que con las no elitistas. Esto se debe a que en las configuraciones no elitistas hay una gran variabilidad y dada la naturaleza ruidosa del problema, es probable que se obtengan 'falsas' soluciones buenas.

	BCP	Dora	Thunder
MizunoAIOrig	-134.44	-285.67	-279.67
MizunoAIEV-DoraBCP-Elitism	-103.77	-269.22	-210.56
MizunoAIEV-DoraBCP-NoElitism	-167.67	-232.56	-218.89
MizunoAIEV-BCP-Elitism	-146.33	-212.33	-230.33
MizunoAIEV-BCP-NoElitism	-173.33	-282.33	-238.56
MizunoAIEV-Dora-Elitism	-142.11	-255.33	-260.55
MizunoAIEV-Dora-NoElitism	-161.33	-286.44	196.33

Tabla 2
Diferencias de salud entre los agentes optimizados y los bots del estado del arte. Diferencias entre el bot de cada fila frente al de la columna.

4. Conclusiones y Trabajo Futuro

Este trabajo presenta un estudio preliminar sobre la aplicación de diferentes esquemas de Algoritmos Genéticos (AGs) para optimizar agentes diseñados para el combate en un juego de lucha 1 vs 1. Los resultados, aunque no son extraordinarios, muestran algunas mejoras sobre un agente de partida muy bueno. De modo que continuaremos esta línea de investigación intentando mejorar los resultados de optimización de los AG mediante esquemas más avanzados, como una configuración con operadores específicos y adaptados, por ejemplo una mejor función de fitness. También realizaremos pruebas más completas contra agentes de alto nivel de la Fighting AI Competition.

Referencias

[1] D. E. Goldberg, Genetic Algorithms in search, optimization and machine learning, Addison Wesley, 1989.

[2] N. Sato, S. Temsiririrkkul, S. Sone, K. Ikeda, Adaptive fighting game computer player by switching multiple rule-based controllers, in: 2015 3rd international conference on applied computing and information technology/2nd international conference on computational science and intelligence, IEEE, 2015, pp. 52–59.

[3] K. Majchrzak, J. Quadflieg, G. Rudolph, Advanced dynamic scripting for fighting game AI, in: International Conference on Entertainment Computing, Springer, 2015, pp. 86–99.

[4] S. Yoshida, M. Ishihara, T. Miyazaki, Y. Nakagawa, T. Harada, R. Thawonmas, Application of monte-carlo tree search in a fighting game AI, in: 2016 IEEE 5th Global Conference on Consumer Electronics, IEEE, 2016, pp. 1–2.

[5] S. Yoon, K.-J. Kim, Deep q networks for visual fighting game AI, in: 2017 IEEE conference on computational intelligence and games (CIG), IEEE, 2017, pp. 306–308.

[6] Y. Takano, W. Ouyang, S. Ito, T. Harada, R. Thawonmas, Applying hybrid reward architecture to a fighting game AI, in: 2018 IEEE Conference on Computational Intelligence and Games (CIG), IEEE, 2018, pp. 1–4.

[7] G. L. Zuin, Y. P. Macedo, L. Chaimowicz, G. L. Pappa, Discovering combos in fighting games with evolutionary algorithms, in: Proceedings of the Genetic and Evolutionary Computation Conference 2016, 2016, pp. 277–284.

[8] E. H. Skjærseth, H. Vinje, Evolutionary algorithms for generating interesting fighting game character mechanics, Master's thesis, NTNU, 2020.

[9] Z. Tang, Y. Zhu, D. Zhao, S. M. Lucas, Enhanced rolling horizon evolution algorithm with opponent model learning, IEEE Transactions on Games (2020).

[10] A. Fernández-Ares, A. M. Mora, P. García-Sánchez, P. A. Castillo, J. J. Merelo, Analysing the influence of the fitness function on genetically programmed bots for a real-time strategy game, Entertainment Computing 18 (2017) 15–29.

[11] P. García-Snchez, A. Tonda, A. M. Mora, G. Squillero, J. J. Merelo, Automated playtesting in collectible card games using evolutionary algorithms, Knowledge-Based Systems 153 (2018) 133–146.

[12] M. Salem, A. M. Mora, J. J. Merelo, Overtaking uncertainty with evolutionary TORCS controllers: Combining BLX with decreasing operator and grand prix selection, IEEE Transactions on Games (2021).

[13] C. Yin Chu, R. Thawonmas, Applying fuzzy control in fighting game AI, in: 77th National Convention of IPSJ, volume 2, 2015, pp. 101–102. Http://www.ice.ci.ritsumei.ac.jp/ ruck/PAP/ipsj4P-03.pdf.