Desarrollo de un Bot Evolutivo Interactivo para Unreal Tournament 2004

José L. Jiménez, Antonio J. Fernández-Leiva, Antonio M. Mora

Resumen— **En este trabajo se ha implementado un algoritmo genético interactivo en un bot para el juego Unreal Tournament 2004, utilizando como base un bot que se definió anteriormente modelando el conocimiento de un jugador experto. El algoritmo ofrece dos tipos de interacción: por parte de un experto en el juego, o por parte de un experto en el algoritmo. Cada uno influirá en distintos aspectos del algoritmo, para dirigirlo hacia unos mejores resultados con respecto a la humanidad que presente el bot (objetivo de este artículo). Se ha hecho un análisis de la influencia del experto en la ejecución y los resultados muestran cierta mejoría con la versión sin interactividad.**

El mejor bot obtenido como resultado ha sido presentado a la BotPrize competition de 2014 (buscan el bot más humano posible), quedando en segundo lugar.

Palabras clave— **Inteligencia Computacional, Inteligencia Artificial, Computación Evolutiva, Algoritmos Evolutivos, Algoritmo Evolutivo Interactivo, Unreal Tournament, Human-like, Humanidad, Test de Turing**

I. Introducción

Unreal Tournament 2004[1], también conocido como UT2004 o UT2K4, es un videojuego de acción en primera persona, principalmente creado para la experiencia multijugador, en el que el objetivo varía según el modo de juego elegido. El juego incluye un modo de un solo jugador que emula el juego multijugador a través del uso de entidades controladas por ordenador que simulan a los jugadores humanos, llamadas bots. El juego fue desarrollado por Epic Games y Digital Extremes, y publicado por Atari. La mayor adición que UT2004 hace al universo Unreal es la introducción de vehículos a la fórmula de los juegos FPS (*First person Shooter*), siguiendo los pasos de otros juegos como Tribes 2, Halo: Combat Evolved, y Battlefield 1942.

Una de las características centrales y definitorias de los Unreal Tournaments, y uno de los mayores componentes de su éxito, es la facilidad con la cual muchos jugadores pueden crear y compartir sus propias modificaciones y contenido propio tales como mapas, armas, y modos de juego innovadores, llamados también mods. Varios de los mapas creados por jugadores suelen ser más populares entre los jugadores que aquellos que vienen con el juego. En consecuencia, muchos, sino la mayoría, de los servidores de UT alrededor del mundo incluyen mapas de terceros. Los mapas personalizados pueden ser creados con el editor (UnrealED) incluido con el juego.

Departamento de Arquitectura y Tecnología de Computadores. CITIC, ETSIIT, Universidad de Granada E-mail: amorag@geneura.ugr.es, Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga. e-mail afdez@lcc.uma.es, josejl1987@gmail.com

Los elementos de inteligencia artificial **(IA** en adelante) que se usan en los juegos de ordenador han recorrido un largo camino. Al principio los sistemas desarrollados se basaban en un conjunto de reglas programadas directamente en el código del juego o en secuencias de comandos de comportamiento interpretadas por el código, dónde el factor aleatorio era muy influyente.

Otro paso en el proceso de desarrollo fue la introducción de métodos sencillos de Ciencias de la Computación, tales como las máquinas de estados finitos [2] . Esta técnica sigue siendo popular y se utiliza frecuentemente. Una máquina de estados finitos describe el comportamiento de los enemigos controlados por el ordenador (denominados popularmente NPCs, *Non-Player Characters* por sus siglas en Inglés). Sin embargo, como la demanda de los jugadores crecía día a día, los juegos se hicieron cada vez más complejos, gracias al uso de algoritmos de computación cada vez más avanzados.

Una arquitectura de IA para juegos FPS suele estar organizada en una jerarquía con cuatro componentes principales integrados: *comportamiento*, *movimiento*, *animación* y *combate*.

Este artículo propone una modificación del Bot Experto[3], con la inclusión de un algoritmo genético interactivo. [4] El objetivo es conseguir un bot que se comporte de la manera más humana posible, en lugar de la idea inicial de los autores de crear un bot muy competitivo en el combate. Además se ha conseguido paralelizar el algoritmo, consiguiendo reducir el tiempo de ejecución a la décima parte.

En el algoritmo implementado se han tenido en cuenta dos tipos de interactividad, dependiendo del perfil del experto humano que evaluará el comportamiento del bot. Estos perfiles son *experto en Unreal Tournament 2004* y *experto en algoritmos genéticos*.

II. Trabajos Previos

A. Test de Turing para bots

El test de Turing para bots [5] es una variante del Test de Turing, donde un juez humano que interactúa con un mundo virtual debe distinguir entre otros humanos y bots que también interactúan con ese mundo virtual.

El test de Turing para bots se propuso para avanzar en los campos de la Inteligencia Artificial e Inteligencia Computacional en relación a los videojuegos. Se consideró que un bot implementado deficiente-

mente implicaba un juego mediocre, por lo que un bot que sea capaz de pasar esta prueba, y por lo tanto que podría ser indistinguible de un jugador humano, podría mejorar directamente la calidad de un juego, además de que también sirvió para desacreditar la noción errónea de que 'la IA para juegos es un problema resuelto.'

Se hace hincapié en un bot que interactúa con otros jugadores en un entorno multijugador. A diferencia de un bot que simplemente tiene que tomar decisiones óptimas y similares a las humanas para jugar o vencer a un juego, esThe 2K BotPrizete bot tiene que tomar las mismas decisiones y al mismo tiempo dar la impresión de tener un comportamiento similar a un humano.

Este test fue diseñado para probar la capacidad de un bot para interactuar con un entorno de juego en comparación con un jugador humano. Esto se convirtió en un torneo con unos cuantos objetivos en mente:

- Hay tres participantes: un jugador humano, un bot y un juez.
- El bot necesita parecer más humano que el jugador humano. Las calificaciones del juez no son bipolares, ambos jugadores pueden ser calificados de 1 a 5 (1=No es humano, 5=Humano).
- Los tres participantes deben ser indistinguibles, salvo por el nombre, que deberá ser generado aleatoriamente. Estas medidas ayudan a que el juez no se vea influido por el nombre o la apariencia de los jugadores.
- El chat está desactivado durante la partida.
- Los bots no tienen poderes omniscientes como en otros juegos. Estos solo pueden reaccionar ante estímulos que puedan estar disponibles a otro jugador humano. Los participantes humanos tenían un nivel medio, ninguno era inexperto o capaz de jugar a nivel profesional.

En 2008 tuvo lugar el primer torneo 2K BotPrize (en adelante BotPrize), donde utilizó el juego Unreal Tournament 2004. Los participantes crearon sus bots con antelación usando el interfaz GameBots, que fue modificado para ajustarse a las condiciones previamente explicadas.

B. Botprize

El primer torneo BotPrize se celebró en Perth, Australia, el 17 de diciembre de 2008, como parte del simposio IEEE 2008 sobre Inteligencia Computacional y Juegos.

A cada equipo participante se le dio tiempo para configurar y ajustar sus bots al cliente de juego modificado, aunque no se permitieron otros cambios de programación. El torneo se llevó a cabo en las rondas, utilizando partidas en modo Deathmatch de 10 minutos de duración. Los jueces fueron los últimos en unirse al servidor y todos los jueces observaron cada jugador y cada bot exactamente una vez, aunque la pareja formada por jugadores y bots cambiara. Cuando terminó el torneo, ningún bot fue calificado como más humano que cualquier otro jugador.

En los siguientes torneos que se disputaron entre 2009 y 2011, las puntuaciones de los bots fueron mejorando, aunque siguieron sin superar a ningún humano, por lo que el premio quedó desierto. No obstante, en la edición de 2012, dos bots fueron capaces de obtener puntuaciones superiores a los humanos.

Esta afirmación (que un bot que aparente ser más humano que un propio humano) es una exageración, ya que en el torneo en el que los bots tuvieron éxito, la calificación media de "humanidad"de los jugadores humanos era sólo del 41, 4 %.

Esto demuestra algunas limitaciones de esta prueba de Turing, ya que los resultados demuestran que el comportamiento humano es más complicado y cuantitativo de lo que se pensaba.

También se cree que los métodos y las técnicas desarrolladas para el test de Turing para bots serán útiles en otros campos distintos a los videojuegos, como los entornos virtuales de formación y en la mejora de la interacción humano-robot.

C. Otros bots

Los bots ganadores de la edición 2012 fueron *UT*'2Bot [6] y MirrorBot [7]. Más específicamente, *UT*'2Bot fue creado por *UT*'2 , un equipo de la universidad de Texas; este bot se basa en dos ideas principales: la neuroevolución multiobjetivo, para aprender habilidades de combate filtrando aquellas que no den una sensación de humanidad, y el uso de una base de datos con registros de partidos jugadas por humanos, lo cual ayuda a mejorar la navegación por el mapa de juego. Por otra parte, Mihai Polceanu, un estudiante de doctorado de Rumanía, creó un bot llamado *MirrorBot*, que imitaría las reacciones del oponente, basándose en técnicas de interacción humanas. Dado que los jueces son humanos, el bot imitaría las reacciones propias de un humano, consiguiendo engañar al juez. No obstante, esta técnica solo funciona cuando imita a un oponente humano.

En todo caso, la mayor referencia del bot presentado en este artículo es el Bot Experto [3], ya que este bot (esto es, NizorBot) es una versión evolucionada del mismo.

La idea principal que reside en el desarrollo del bot experto [3], es la de tener un agente que pueda ser controlado por una serie de estados primarios, cuyo comportamiento a su vez, pueda ser adulterado por una serie de estados secundarios. El paso de unos estados a otros dependerá de las decisiones tomadas por el sistema experto.

El controlador está constituido por dos capas distintas, la capa cognitiva y la capa reactiva. La capa cognitiva se encarga del control de la máquina de estados finita. En base a lo que ocurra en el entorno (información percibida a través de los sensores), esta capa será la encargada de decidir cuando se pasa de un estado a otro y con qué estado secundario,

apoyándose en el uso de la base de datos y el sistema experto. La capa reactiva, al contrario que la capa cognitiva, no realiza ningún tipo de deliberación, simplemente, responde con acciones ciegas ante eventos (a modo de sistema nervioso). Es decir, las reacciones disparadas por esta capa representan acciones o respuestas inconscientes del agente y por lo tanto, no deben ser controlables. Es por este motivo, que las acciones disparadas por la capa reactiva no son controlables ni tan siquiera por la capa cognitiva.

III. Unreal Tournament 2004

Unreal Tournament 2004 dispone de una gran cantidad de modos de juego, pero en nuestro caso, el bot se ha diseñado específicamente para la modalidad Deathmatch (Figura 1). Más concretamente, *Deathmatch* es un tipo de juego, en el que el objetivo es o bien llegar a un cierto número de frags (o muertes), o bien con el número más alto de frags en el límite de tiempo para la partida. Deathmatch es un tipo de juego difícil, que requiere mucho entrenamiento, si es que uno está jugando contra otros jugadores . El modo Deathmatch se puede jugar contra una mezcla de oponentes humanos o manejados por el ordenador.

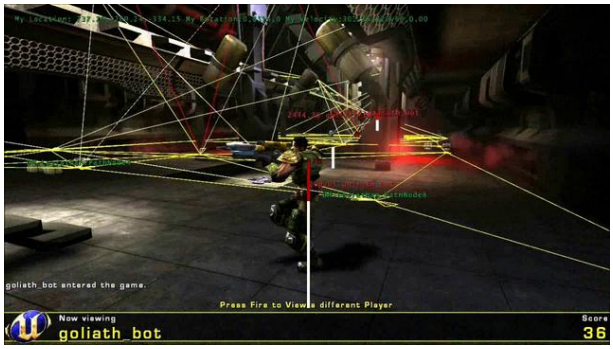

Fig. 1. Deathmatch en UT2004 con grafo usado para A*

Unreal Tournament 2004 goza de una amplia variedad de armas, combinando armas clásicas actualizadas de Unreal Tournament, nuevas armas de UT2003, y armas totalmente nuevas. Varias armas fueron diseñadas específicamente para su uso en modos de juego basados en vehículos, y suelen aparecer únicamente en es modos de juego. Las superarmas pueden desactivarse como una opción, por lo que puede o no aparecer en el juego dependiendo de la configuración del servidor.

IV. Metodología

La intervención humana en el guiado del proceso de optimización de problemas ha resultado ser muy provechosa en un buen número de casos mejorando significativamente diversos aspectos de la búsqueda como puede ser la calidad de la solución obtenida o bien el rendimiento del algoritmo empleado para obtenerla [8]. Esta mejora es especialmente significativa en problemas que demandan un componente subjetivo o que se formulan en un espacio de soluciones que son difíciles de evaluar matemáticamente porque inherentemente contienen algún tipo de factor que se mueve en el terreno de la psicología o la misma naturaleza humana; este es el caso de problemas que se basan en la creatividad o que necesitan encontrar soluciones cuyo fin máximo es el de incrementar la satisfacción humana; precisamente muchos de estos problemas se encuentran en áreas que se pueden asociar claramente a la sensibilidad humana tales como la música, el arte o los videojuegos.

Sin embargo, la interacción de un algoritmo de optimización con el ser humano puede realizarse de formas diversas y puede establecerse desde puntos de vista bastante diferentes: por una parte, la presencia de un experto en el problema a resolver es importante para evaluar correctamente (quizás en un plano subjetivo) la calidad de los candidatos, mientras que la presencia de un humano experto en el algoritmo es necesaria para modificar o controlar el proceso algorítmico de búsqueda de las soluciones [9]. A pesar de las ventajas de un algoritmo interactivo, la realidad es que estos producen fatiga al usuario (tanto al experto en el problema como al experto en el algoritmo de optimización) debido normalmente al alto número de intervenciones que deben realizar. Una solución a este problema consiste en "proactivizar" el algoritmo de forma que el método de optimización subyacente pueda predecir a priori las decisiones del experto antes de la intervención de este (por ejemplo basándose en sus decisiones anteriores) y de esta forma reducir el número de intervenciones. En lo que respecta a técnicas metaheurísticas pues ya se han realizado investigaciones eficaces en este sentido [10].

Relacionado con lo anteriormente declarado, la mayor novedad del trabajo presentado aquí con respecto al Bot Experto ha sido la inclusión de la evaluación interactiva en el algoritmo genético, lo cual parece ser algo natural para la generación de bots que se *comporten humanamente* ya que no es fácil de definir exactamente las caracterícas de un comportamiento de esta clase.

El algoritmo genético que se ha implementado es el mismo que se utilizó en el bot experto original. No obstante, se le ha añadido un componente interactivo a la hora de evaluar los individuos.

A continuación se describirá con detalle las características del algoritmo empleado.

A. Estructura del cromosoma

El cromosoma de cada individuo se divide en 6 bloques de información, utilizando un total de 26 genes, tal como se puede ver en la Figura 2

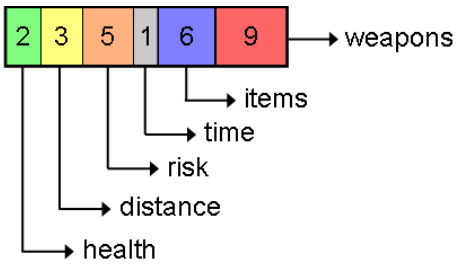

Fig. 2. Representación gráfica del cromosoma empleado

A continuación se describen la función de cada uno de estos bloques:

Bloque de distancias

Conjunto de parámetros que regulan las distancias a partir de las cuales el agente ataca con cierto tipo de armas y mantiene las distancias de seguridad con respecto al enemigo.

- *Gen 0 :* Regula el valor de la distancia que el agente considera como corta. Varía entre 0 y 1200.
- *Gen 1 :* Regula el valor de la distancia que el agente considera como media. Varía entre el valor del Gen 0 y 2000.
- *Gen 2 :* Regula el valor de la distancia que el agente considera como larga. Varía entre el valor del Gen 1 y 2800.

Bloque de selección de armas

Conjunto de parámetros que regulan la prioridad que se le asigna a cada arma en función de la distancia, la altura y la situación en la que se encuentra el agente con respecto al enemigo.

- *Gen 3-11:* Controlan la prioridad que se le asigna a cada arma. Varían entre 0 y 100.

Bloque de control de salud

Conjunto de parámetros que controlan los niveles de salud a partir de los cuales el agente decide ser ofensivo, neutro o defensivo.

- *Gen 12:* Si el agente posee menos salud que la indicada por este gen, el perfil sera generalmente defensivo. Varía entre 0 y 100.
- *Gen 13:* Si el agente posee menos salud que la indicada por este gen, el perfil sera generalmente neutro. Varía entre el valor del gen 12 y 160. y 100.

Bloque de riesgo

Conjunto de parámetros que controlan la cantidad de puntos de salud que el agente esta dispuesto a arriesgar.

- *Gen 14:* Varía entre 5 y 30.
- *Gen 15:* Varía entre 15 y 80.
- *Gen 16:* Varía entre 15 y 60.
- *Gen 17:* Varía entre 10 y 120.
- *Gen 18:* Varía entre 20 y 100.

Bloque de tiempo

Este bloque esta compuesto por un solo gen que regula la cantidad de tiempo a partir de la cual el agente considera que a perdido de vista al enemigo.

- *Gen 19:* Varía entre 3 y 9 segundos.

Bloque de control de items

Conjunto de parámetros que regulan la prioridad que se le asigna a determinados items. Cuanto más alto sea el valor asignado, mayor prioridad se le asignará al item a la hora de ser 'timeado'. El valor de cada gen varía entre 0 y 100.

- *Gen 20:* Regula la prioridad asignada al Super Shield Pack.
- *Gen 21:* Regula la prioridad asignada al Shield Pack.
- *Gen 22:* Regula la prioridad asignada a la Lightning Gun.
- *Gen 23:* Regula la prioridad asignada al Shock Rifle.
- *Gen 24:* Regula la prioridad asignada al Flak Cannon y el Rocket Launcher.
- *Gen 25:* Regula la prioridad asignada a la Minigun.

B. Medición del fitness

Para poder medir la habilidad del individuo es necesario jugar una partida contra otro bot, en este caso hemos usado el bot nativo de UT2004 en su máximo nivel, ya que hemos decidido que tiene un comportamiento más estable que otras alternativas.

Estas partidas tienen una duración de 5 minutos y se ha utilizado el mapa DM-Ironic. Una vez que terminan los 5 minutos de juego, se recopilan los datos necesarios para la función de fitness.

La función utilizada para medir el fitness de los individuos es la siguiente:

$$f(fr,d,dmgG,dmgT,s1,s2,tS,tL) = \begin{cases} (fr-d)+s2+\frac{s1}{2} \\ +log((dmgG-dmgT)+1) & sifr \geqslant d \\ \frac{fr}{d}+s2+\frac{s1}{2} \\ +log((dmgG-dmgT)+1) & sifr < d \end{cases}$$

Donde:

- fr: Número de frags.
- d: Número de bajas.
- $dmgG$: Daño total causado.
- $dmgT$: Daño total recibido.
- $s1$: Número de escudos pequeños recogidos.
- $s2$: Número de escudos grandes recogidos.

Por norma general, las funciones de fitness para UT2004 suelen ser más sencillas, utilizando solo el número de victorias (fr) y derrotas (d), sin embargo, debido al comportamiento no determinista del juego, se ha optado por usar otros factores para estabilizar la función de fitness. Además estos factores sirven para evaluar otros aspectos del bot, sobre todo su comportamiento a la hora de recoger objetos. Los objetos juegan un papel muy importante en UT2004, por lo que un bot eficiente debe tener un buen comportamiento en este aspecto.

Estos factores son los siguientes:

Número de escudos recogidos (s1 y s2): El número máximo de escudos pequeños que pueden obtenerse en una partida de cinco minutos es de doce, mientras que por el contrario el número máximo de escudos grandes que pueden obtenerse es de quince. Por este motivo, en la ecuación se divide el número de escudos pequeños por dos.

Daño total hecho (dmgG) y recibido (dmgT): Debido a que la diferencia de daños total obtenida suele

oscilar entre los miles, decidimos utilizar el logaritmo en base de diez de dicha operación para obtener un valor que favoreciese al candidato más agresivo. Al resultado de la diferencia se le suma una constante para asegurarnos de que no se obtienen valores negativos.

C. Método de selección

Se ha optado por utilizar el algoritmo de selección por ruleta, un algoritmo de selección probabilista. Esto quiere decir que un individuo tiene una probabilidad de ser seleccionado proporcional a su fitness. Sin embargo, al entrar en juego la probabilidad, es posible que el mejor candidato no sea elegido, lo que puede ser útil al impedir que un individuo monopolice la solución, aunque esto implique una velocidad de convergencia menor.

Además se ha optado por el uso del elitismo. Esto implica que en cada generación del algoritmo se han incluido los 5 mejores individuos de la generación anterior, sin modificación alguna.

D. Operadores genéticos

Se ha optado por la opción del cruce uniforme. Este tipo de cruce implica una mayor diversidad del individuo, donde cada gen del individuo descendiente tiene la misma probabilidad de ser heredado de uno de los dos progenitores.

Para la batería de pruebas se han empleado los siguientes parámetros en el algoritmo:

- *Tamaño de población*:30
- *Generaciones*:50
- *Variación en la mutación*:10 %
- *Probabilidad de la mutación*:33 %
- *Elitismo*:5
- *Cruce*:Uniforme

E. Evaluaciones interactivas

Las evaluaciones interactivas se realizan de la siguiente forma:

Al llegar a un punto de parada del algoritmo, se muestra en el formulario de la Figura 3.

Este formulario muestra los datos del mejor individuo seleccionado de la generación en la que se ha producido la parada. La persona encargada de evaluar el individuo debe ver la partida grabada del mismo. Para ello tiene que buscar la partida guardada a partir de la generación actual y el ID del bot actual (la fila de posición). Una vez vista la partida, el evaluador debe rellenar el formulario según las impresiones que ha obtenido de la partida que acaba de ver. Cada casilla que se marque hará que una parte del cromosoma se quede bloqueada, consiguiendo disminuir el espacio de búsqueda. El algoritmo seguirá ejecutándose normalmente hasta que finalice o llegue otro punto de parada definido por el usuario

A la hora de seleccionar las generaciones donde se debe realizar las evaluaciones interactivas se ha decidido por mantener la equidistancia entre las mismas, es decir:

- *1 evaluación* : Generación 25
- *2 evaluaciones* : Generaciones 16 y 33
- *4 evaluaciones* : Generaciones 10, 20, 30 y 40

Uno de los mayores problemas que se encontraron en la versión anterior del bot fue la cantidad de tiempo empleado para realizar una batería de pruebas.

Unreal Tournament 2004 no permite acelerar la velocidad de juego sin provocar efectos secundarios en el comportamiento de los bots, por lo que es necesario jugar las partidas a velocidad real. Por lo tanto, teniendo en cuenta que usamos generaciones de 30 individuos y 5 minutos por cada partida, pues es necesario algo más 150 minutos para evaluar una generación, ya que es necesario cargar y descargar el nivel cada vez que se evalúa un individuo.

Por lo tanto, una solución viable para acelerar el tiempo de evaluación es ejecutar varias partidas simultáneamente. Esto es posible ya que cada individuo de una generación es independiente del resto, por lo que pueden ser evaluados a la vez. No obstante, la implementación de esta solución ha sido un tanto costosa, ya que el Unreal Tournament 2004 no fue diseñado para ser ejecutado concurrentemente.

Para la implementación de la paralelización ha resultado muy útil la extensión de Pogamut ut2004-tournament. Esta extensión permite lanzar partidas desde Java de una forma sencilla, encargándose de lanzar ejecutar el servidor de UT2004 con las opciones que se deseen, además de ejecutar los bots. De hecho, todo el proceso de paralelización se ha construido alrededor de la funcionalidad de esta extensión.

F. Arquitectura

Si bien la parte referente al comportamiento del bot ha permanecido prácticamente intacta (salvo arreglar algunos bugs en el algoritmo de navegación), el algoritmo genético se ha reimplementado utilizando otras herramientas:

El original usaba su propia implementación de los operadores genéticos. En esta versión se ha reemplazado por la librería Watchmaker [11], para permitir una mayor flexibilidad.

El original usaba una base de datos SQLite para almacenar los datos. En esta versión hemos optado por guardar los datos en RAM durante la ejecución, ahorrando tiempo en lectura/escritura de disco. Para ello se ha creado una aplicación servidor que se encarga de gestionar los datos durante la ejecución. Para almacenar los datos en disco se han utilizado archivos XML generados automáticamente con la librería XStream [12] para Java (esta librería permite serializar cualquier clase Java). Estos archivos XML son más pesados que una base de datos SQLite, pero no se utilizan durante la ejecución de las pruebas.

Por lo tanto, se ha programado una aplicación utilizando una estructura cliente/servidor. El servidor se ejecuta dentro un hilo de la interfaz gráfica, y los clientes son los bots ejecutados por la interfaz gráfica, comunicándose mediante sockets.

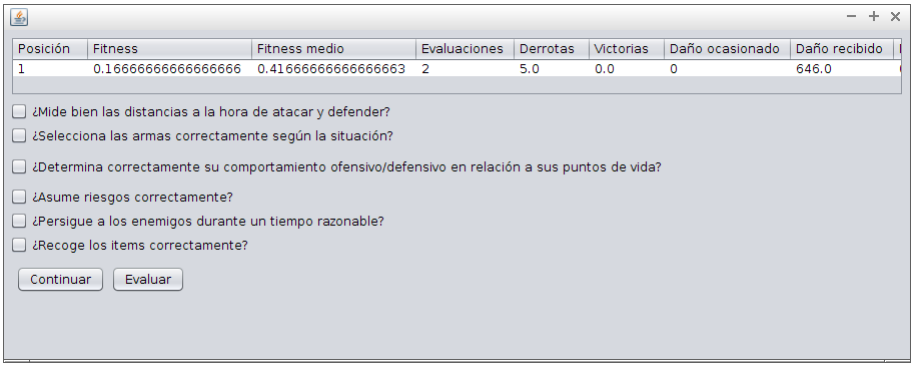

Fig. 3. Formulario para evaluación interactiva

La aplicación principal se encarga de lanzar los bots y de asignarles el cromosoma correspondiente. Este cromosoma se lo envía utilizando los sockets, y una vez que la partida finaliza, el bot le envía al servidor las estadísticas de la partida que se ha jugado para evaluar el fitness del bot.

Una vez que se han evaluado todos los bots de una generación, se ejecuta el algoritmo genético implementado para calcular los nuevos individuos, y se vuelve a repetir el mismo proceso hasta alcanzar el número de generaciones deseado.

Por supuesto, durante este proceso pueden producirse una gran cantidad de errores por diversos motivos, por lo que el servidor es capaz de detectar estos errores y actuar en consecuencia, por ejemplo, reiniciando la partida de un bot que ha fallado. Para ello es necesario que el servidor esté al corriente del estado de cada partida.

V. Análisis de Resultados

En esta sección se describirán los resultados obtenidos en las distintas pruebas tanto propias como las evaluaciones por terceros.

En las evaluaciones propias se han realizado varias ejecuciones del algoritmo genético previamente descrito. Para comprobar el impacto de la evaluación interactiva se han realizado varias baterías de pruebas usando una, dos y cuatro evaluaciones interactivas. interactivas, además de una batería de pruebas sin evaluaciones interactivas, mientras que para evaluar la humanidad del bot, este fue inscrito en el Botprize 2014

En la Figura 4 se muestra una comparativa entre las 4 variaciones del algoritmo, dependiendo del grado de interactividad que se ha utilizado:

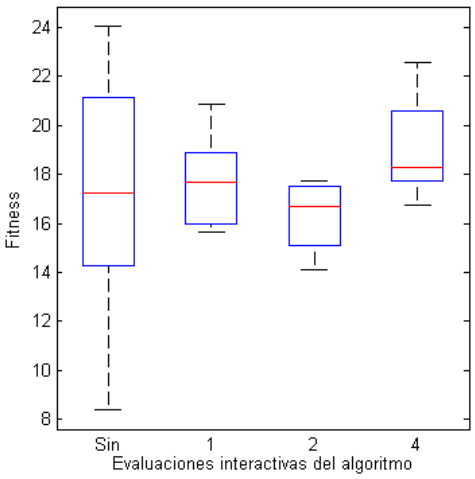

Fig. 4. Comparativa de fitness

Se puede observar que la mediana (representada por la línea roja) ha aumentado a medida que las evaluaciones interactivas han ido aumentando. También se puede ver que la diferencia entre el fitness más alto y el más bajo de cada tipo disminuye considerablemente al usar la interactividad.Al usar la evaluación interactiva se congelan ciertas partes del cromosoma, por lo que la diversidad de la población disminuye, lo que provoca que los fitness de los individuos sean más similares.

A. Evaluación de humanidad

Para evaluar la humanidad del bot, se decidió que inscribirlo en la edición 2014 de Botprize sería la forma más adecuada de obtener una evaluación exhaustiva. A continuación se describirá con mayor detalle las novedades de esta edición y los resultados obtenidos.

La edición 2014 de Botprize se celebró en Dortmund, Alemania en Agosto del 2014 como parte del *CIG (IEEE Conference on Computational Intelligence and Games)*. El objetivo de la competición era determinar si un bot que fuera capaz de jugar a un videojuego podría convencer a un panel de jue-

ces expertos de que se comportaba como un jugador humano.

En el protocolo de evaluación de *Botprize original*, la evaluación recaía exclusivamente de estos jueces humanos que participaban en las partidas directamente. Es decir, que los jueces evaluaban a partir de su perspectiva en primera persona. Pero en 2014 se modificó el protocolo añadiendo evaluaciones de terceros, usando un plataforma de crowdsourcing. Por lo tanto, la humanidad del bot se calcula a partir de las *evaluaciones en primera persona* (FPA, First Person Assessment) y en *tercera persona* (TPA, Third Person Assessment).

En ediciones anteriores los jueces eran personas independientes, pero en esta edición los participantes y otros miembros de organización del Botprize actuaron como jueces en primera persona.

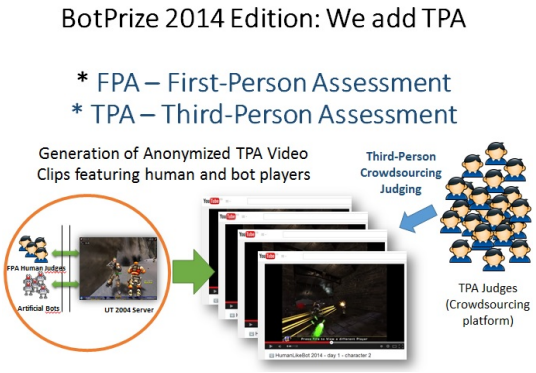

Fig. 5. Nuevo sistema de evaluación de Botprize 2014 (I)

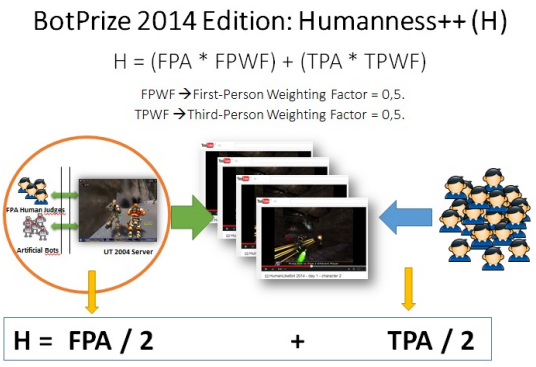

Fig. 6. Nuevo sistema de evaluación de Botprize 2014 (II)

Además, existe un objetivo secundario en el concurso, donde se evalúa la capacidad de los bots para poder juzgar la humanidad de los otros rivales.Este objetivo no es una prioridad de nuestro bot, tal como se puede ver en las siguientes figuras 7 y 8.

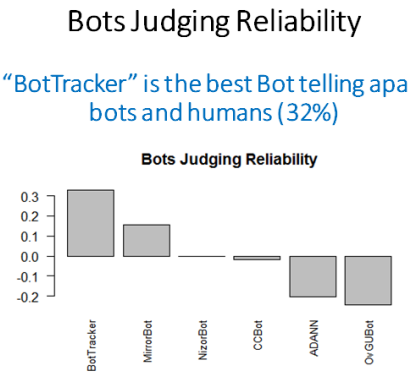

Fig. 7. Resultados de detección de bots de Botprize 2014(I)

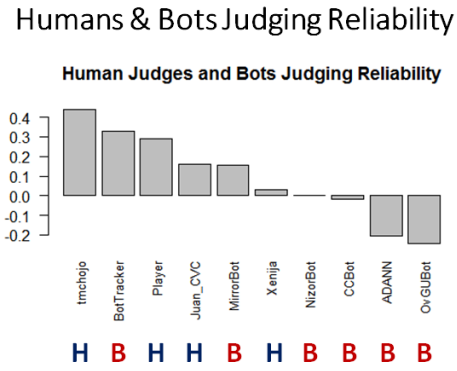

Fig. 8. Resultados de detección de bots de Botprize 2014(II)

El ganador de esta edición (ver figuras 9 y 10) vuelve a ser MirrorBot, programado por Mihai Polceanu (NIB CERV Centre de Réalité Virtuelle, France).

Valoramos el segundo lugar obtenido muy positivamente, ya que solo ha sido superado por MirrorBot, uno de los campeones de la edición de 2012.

En la figura 10 se recogen los resultados de todos los participantes.Las filas resaltadas de azul son los jugadores humanos,las amarillas son los bots que han competido oficialmente y las grises aquellos bots que han participado simplemente por motivos de investigación.

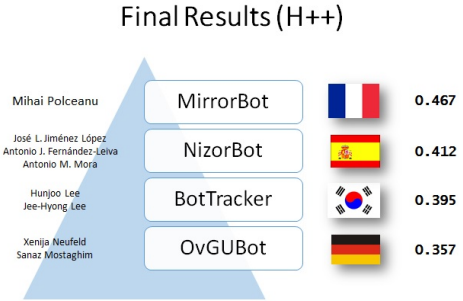

Fig. 9. Resultados de humanidad de Botprize 2014(I)

NizorBot ha sido programado teniendo en cuenta

los patrones de comportamiento de un jugador humano experto en Unreal Tournament 2004, lo que explica que haya obtenido un buen índice de humanidad, sobre todo a la hora de combatir, que es el comportamiento que más destaca a la hora de juzgar a un bot. No obstante, existe cierto margen de mejora a la hora de navegar por el mapa, ya que depende mucho del grafo de navegación del mapa.

MirrorBot tiene un mejor comportamiento en este área.

BotName	FPA	TPA	H++
Xenija	0.17139763	0.8235294	0.4974635
MirrorBot	0.20164771	0.7333333	0.4674905
Player	0.19328127	0.6315789	0.4124301
tmchojo	0.17757519	0.6470588	0.4123170
NizorBot	0.11821633	0.7058824	0.4120493
BotTracker	0.20070203	0.5909091	0.3958056
CCBot	0.06214746	0.7058824	0.3840149
Juan_CVC	0.12372294	0.6190476	0.3713853
OvGUBot	0.10545765	0.6086957	0.3570767
ADANN	0.08351664	0.4761905	0.2798536

Fig. 10. Resultados de humanidad de Botprize 2014(II)

VI. Conclusiones y trabajo futuro

En este trabajo se ha implementado un algoritmo genético interactivo en un bot para el juego Unreal Tournament 2014. Viendo los resultados se puede apreciar que se han mejorado los resultados de la versión anterior sin interactividad, además de que se ha conseguido mejorar los tiempos de evaluación, consiguiendo una mejora de un 1000 % en ese aspecto.

Como líneas de trabajo futuro se podrían destacar la creación de una forma de evaluación de los individuos y la implementación de un nuevo algoritmo de navegación para el bot. La función de fitness actual tiene mucho ruido debido a la complejidad del Unreal Tournament 2004, siendo muy dependiente del azar, por lo que resulta interesante buscar formulas nuevas; otra línea de trabajo consiste en reducir la fatiga de los usuarios humanos, motivadas por las intervenciones a las demandas del algoritmo subyacente y por la propia naturaleza estocástica de mismo (lo cual requiere múltiples ejecuciones), proactivizando nuestra propuesta.

El algoritmo de navegación actual usa un grafo que está incluido en el mapa del juego, siendo imposible que el bot navegue sin esa información. Además ese tipo de navegación resulta poco humana, ya que el bot se va moviendo por los distintos puntos del grafo en línea recta (salvo cuando entra en modo de combate). Es otra vía de posible mejora en el futuro.

Agradecimientos

Este trabajo ha sido financiado en parte por el proyecto PYR-2014-17 del proyecto GENIL - CEI BIOTIC (Granada); también ha sido parcialmente subvencionado por la Junta de Andalucía dentro del proyecto P10-TIC-6083 (DNEMESIS[1]), por el MICINN dentro del proyecto ANYSELF[2], y la Universidad de Málaga, Campus de Excelencia Internacional Andalucía Tech.

Referencias

[1] http://www.unrealtournament.com/, "Unreal tournament," 2014.

[2] Michael A Arbib, *Theories of abstract automata*, Prentice-Hall, 1969.

[3] R. Caballero P. García-Sánchez J.J. Merelo P.A. Castillo R. Lara-Cabrera A.M. Mora, F. Aisa, "Designing and evolving an unreal tournamenttm 2004 expert bot," *Springer*, pp. 312–323, 2013.

[4] Karl Sims, "Artificial evolution for computer graphics," *ACM SIGGRAPH Computer Graphics*, vol. 25, no. 4, pp. 319–328, 1991.

[5] http://botprize.org/, "The 2k botprize," 2014.

[6] Jacob Schrum, Igor V Karpov, and Risto Miikkulainen, "Ut? 2: Human-like behavior via neuroevolution of combat behavior and replay of human traces," pp. 329–336, 2011.

[7] Mihai Polceanu, "Mirrorbot: Using human-inspired mirroring behavior to pass a turing test," pp. 1–8, 2013.

[8] Gunnar Klau, Neal Lesh, Joe Marks, and Michael Mitzenmacher, "Human-guided search," *Journal of Heuristics*, vol. 16, pp. 289–310, 2010.

[9] Carlos Cotta and Antonio José Fernández Leiva, "Bio-inspired combinatorial optimization: Notes on reactive and proactive interaction," in *Advances in Computational Intelligence - 11th International Work-Conference on Artificial Neural Networks, IWANN 2011, Part II*, Joan Cabestany, Ignacio Rojas, and Gonzalo Joya Caparrós, Eds. 2011, vol. 6692 of *Lecture Notes in Computer Science*, pp. 348–355, Springer.

[10] Ana Reyes Badillo, Juan Jesús Ruiz, Carlos Cotta, and Antonio José Fernández Leiva, "On user-centric memetic algorithms," *Soft Comput.*, vol. 17, no. 2, pp. 285–300, 2013.

[11] Daniel Dyer, "The watchmaker framework for evolutionary computation (evolutionary/genetic algorithms for java)," 2014.

[12] http://xstream.codehaus.org/, "Xstream - about xstream," 2014.

[13] http://pogamut.cuni.cz/main/, "Pogamut - virtual characters made easy | about," 2014.

[14] http://human machine.unizar.es/, "Human-like bots competition 2014," 2014.

[1] http://dnemesis.lcc.uma.es/wordpress/

[2] http://anyself.wordpress.com/