



UNIVERSIDAD
DE MÁLAGA

DEPARTAMENTO DE LENGUAJES Y CIENCIAS DE LA
COMPUTACIÓN

ALGORITMOS COEVOLUTIVOS COMPETITIVOS PARA LA
PROGRAMACIÓN DE INTELIGENCIA ARTIFICIAL EN
VIDEOJUEGOS

Memoria de la tesis de doctorado

Autora: Mariela Nogueira Collazo

Directores:

Dr. Antonio J. Fernández Leiva

Dr. Carlos Cotta Porras

Málaga, España


Junio de 2017





UNIVERSIDAD
DE MÁLAGA

AUTOR: Mariela Nogueira Collazo

 <http://orcid.org/0000-0003-3456-5512>

EDITA: Publicaciones y Divulgación Científica. Universidad de Málaga



Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-SinObraDerivada 4.0 Internacional:

<http://creativecommons.org/licenses/by-nc-nd/4.0/legalcode>

Cualquier parte de esta obra se puede reproducir sin autorización pero con el reconocimiento y atribución de los autores.

No se puede hacer uso comercial de la obra y no se puede alterar, transformar o hacer obras derivadas.

Esta Tesis Doctoral está depositada en el Repositorio Institucional de la Universidad de Málaga (RIUMA): riuma.uma.es



Dº. Antonio José Fernández Leiva y Dº. Carlos Cotta Porras, Profesores Titulares del Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga

Certifican:

Que Dña. Mariela Nogueira Collazo, Ingeniera en Informática por la Universidad de las Ciencias Informáticas de la Habana y Máster en Ingeniería del Software e Inteligencia Artificial por la Universidad de Málaga, ha realizado en el Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga el trabajo correspondiente a su Tesis Doctoral titulada:

Algoritmos Coevolutivos Competitivos para la Programación de Inteligencia Artificial en Videojuegos.

Revisado el presente trabajo, estimamos que puede ser presentado al tribunal que ha de juzgarlo, y autorizamos la presentación de esta Tesis Doctoral en la Universidad de Málaga.

En Málaga, a 13 de junio de 2017



Fdo.: Dr. Antonio J. Fernández Leiva
Director de la tesis
Profesor Titular
Dpto. de Lenguajes y
Ciencias de la Computación
Universidad de Málaga

Fdo.: Dr. Carlos Cotta Porras
Director de la tesis
Profesor Titular
Dpto. de Lenguajes y
Ciencias de la Computación
Universidad de Málaga



UNIVERSIDAD
DE MÁLAGA

Resumen

Esta investigación trata sobre la aplicación del enfoque competitivo de la Coevolución en la programación de soluciones de Inteligencia Artificial (IA) para videojuegos y propone diferentes modelos para abordarlo. La motivación principal de esta tesis es que la Coevolución resulta un enfoque muy adecuado para explotarlo como método de búsqueda y optimización en videojuegos ya que estos son de por sí escenarios intrínsecamente competitivos. Está demostrado que en los contextos donde resulta complejo definir una medida de calidad de las soluciones, aplicar la Coevolución es muy conveniente porque se podría calcular el *fitness* de los individuos a partir de sus resultados en las competiciones contra las poblaciones adversarias. Este mecanismo de evaluación es tradicional en los videojuegos basados en competición, donde la calidad de un jugador está determinada por su desempeño frente a los oponentes.

Por otra parte, además de las muchas muestras que existen de que la Coevolución es un método de optimización muy adecuado en videojuegos, también se han dedicado varios trabajos en la comunidad científica a explicar las “patologías” que afectan a los modelos coevolutivos. Incluso se han propuesto “remedios” para combatirlos, pero a pesar de eso, la complejidad que genera la dinámica de estos modelos hace que sea difícil evadir esos fallos. Este problema es también una de las motivaciones que han guiado esta investigación pues los algoritmos que hemos diseñado pretenden disminuir el riesgo de padecer algunos de los fallos típicos de un proceso coevolutivo. Nuestra apuesta en este sentido ha sido el empleo de los conocidos métodos de archivo y el



diseño de tres enfoques que mejoran los resultados del modo de uso tradicional de dichos métodos.

Los juegos que han sido objeto de experimentación en esta investigación son del género estrategia en tiempo real (RTS, por sus siglas en inglés: Real Time Strategy Games). Se escogió este género porque ofrece una amplia variedad de problemas muy interesantes en el ámbito de la investigación de la IA aplicada, tales como, la planificación de tareas y recursos en tiempo real teniendo en cuenta al adversario y la toma de decisiones con incertidumbre. Es por esto que es uno de los géneros de juego que más se ha empleado en las investigaciones de IA publicadas en la bibliografía científica.

Son dos los juegos RTS sobre los que se ha experimentado en esta tesis. En ambos se han probado los dos enfoques diseñados en esta investigación. Uno de ellos tiene como objetivo la creación de estrategias de IA para manejar los *bot* y/o los *NPCs* (Non-Player Character) de los juegos RTS antes mencionados. Y el otro, tiene un objetivo mucho más complejo, el de generar de forma simultánea, estrategias de IA y contenido para videojuegos. Este segundo modelo representa el aporte más novedoso de esta investigación, pues hasta donde hemos podido investigar en la bibliografía relacionada, es la primera vez que se logra la coevolución de lo que vendrían siendo (teniendo en cuenta la teoría de la evolución) dos especies tan distintas. Además, constituye un paso importante hacia la creación procedural de juegos completos a nivel de Diseño.

Nuestros algoritmos pueden verse como métodos procedurales de generación de contenidos que permiten crear inteligencias artificiales (o sea, comportamientos) de los *NPCs* y también contenidos (en forma de mapas) del juego. Esta tesis presenta el primer algoritmo conocido en la literatura científica (hasta donde nuestros conocimientos alcanzan) que es capaz de crear proceduralmente ambos contenidos a la vez; y propone un esquema genérico que puede aplicarse a otros videojuegos (no sólo a RTS).

A lo largo de la memoria son explicadas detalladamente las propuestas de soluciones que han sido definidas en esta tesis y se aporta el correspondiente análisis de sus

resultados para poder valorar seriamente su idoneidad.

Palabras Claves: Coevolución, Algoritmos Bioinspirados, Computación Evolutiva, Inteligencia Artificial, Videojuegos, Generación de Contenidos por Procedimientos.



UNIVERSIDAD
DE MÁLAGA

Abstract

The research presented here concerns with the application of the Coevolution competitive approach for programming AI solutions in a videogames context and proposes different models to apply. The main motivation of this thesis is that Coevolution is a highly suitable approach to use as the optimization method in videogame domains because these are, in themselves, intrinsically competitive scenarios. It is demonstrated that in those contexts in which defining a measure of the quality of the solutions is a little complex, to apply a Competitive Coevolutionary approach is very useful because the fitness of the individuals could be derived from its results in the competitions with the adversary populations. This mechanism is a tradition in videogames, where if you have to evaluate the behavior of a player it has to compete against its opponents and is evaluated according to the result.

On the other hand, besides the many examples that show the suitability of Coevolution as an optimization mechanism in videogames, several publications in the scientific community have also explained the “pathologies” that affect coevolutionary models. In fact, some “remedies” have already been proposed to combat them, but even so, the complexity generated by the dynamics of these models makes it difficult to evade these failures. This is also another of the motivating factors of our research because the algorithms designed by us are intended to reduce the risk of the coevolutionary process suffering from these typical failures. Our aim in this regard has been to use well-known archival methods and the design of three approaches that improve the results of the



traditional way of using such methods.

The games used in the experimental phases of this research are RTS games. This genre was chosen because it offers a great variety of very interesting problems in the field of AI research, such as real-time task planning and resources, taking into account the adversary and decision making with uncertainty. This is why it is the most-used game genre in AI research that one can find in the scientific literature. With these games we have tested the algorithms that have been designed in the course of this thesis. These are two base algorithms with three instances of each one. The first one aims to create AI strategies to manage the bots of the aforementioned RTS games. The other has a much more complex goal i.e. the simultaneous generation of AI strategies and content for videogames. This second model represents the most innovative contribution of this research, because, to the best of our knowledge, this is the first time that the coevolution of two, such different, species (taking into account the theory of evolution) has been achieved. All the algorithms that have been designed in the course of this work are explained in detail in this report and the corresponding analysis of the results reached is provided.

Keywords: Coevolution, Bio-inspired Algorithms, Evolutionary Computation, Artificial Intelligence, Videogames, Procedural Contents Generation.

Agradecimientos

A la Universidad de Málaga por la oportunidad de hacer esta tesis.

A mis directores por guiarme y apoyarme en todo el camino.

A mis padres porque les debo todo lo que soy y son un ejemplo de perseverancia.

A mi hermana por ser mi soporte emocional en la distancia.

A mi abuelo que no pudo ver el final de este proyecto.

A Marilis y a Yaima por cuidar de mi en todo momento.

A Antonio, por motivarme a continuar y estar pendiente siempre.

A Lisa y a Lola por haber sido las primeras que me arrojaron en este país.

A mi familia y amigos de Cuba que se han empeñado en acortar las distancias.

A los profesores, compañeros y amigos que de una forma u otra

me han acompañado en el trayecto.

¡A todos ustedes gracias!



UNIVERSIDAD
DE MÁLAGA

Índice general

Acrónimos	XVII
Lista de Algoritmos	XIX
Lista de Figuras	XXI
Lista de Tablas	XXV
1. Introducción	1
1.1. Objetivos	6
1.2. Contribuciones	8
1.3. Publicaciones que avalan esta investigación	11
1.3.1. Revistas indexadas en el <i>Journal Citation Report</i> (JCR)	11
1.3.2. Revista indexada en el <i>Emerging Sources Citation Index</i>	12
1.3.3. Congresos internacionales	12
1.3.4. Congresos nacionales	13
1.4. Estructura de la memoria	14
2. Computación Evolutiva y Coevolución	15
2.1. Bases teóricas	16
2.1.1. Conceptos y bases biológicas de los Algoritmos Evolutivos	16
2.1.2. Ramas de la Computación Evolutiva	25



2.2.	Coevolución	28
2.2.1.	Patologías más frecuentes de los modelos coevolutivos.	29
2.2.2.	Remedios para combatir las patologías	32
2.3.	Conclusiones del capítulo	36
3.	Inteligencia Artificial aplicada a videojuegos	39
3.1.	Problemas de Optimización frecuentes en videojuegos	41
3.2.	Tendencias actuales y retos	43
3.2.1.	Computación afectiva y satisfacción del jugador	44
3.2.2.	La generación procedural de contenidos	48
3.3.	Marcos de trabajo	51
3.3.1.	Competiciones	53
3.4.	Trabajos relacionados con la aplicación de la Coevolución en videojuegos	54
3.5.	Videojuegos que han sido objeto de experimentación en esta tesis	58
3.5.1.	El juego <i>Robot Wars</i>	59
3.5.2.	El juego <i>Planet Wars</i>	67
3.6.	Conclusiones del capítulo	73
4.	Generación de estrategias de juegos mediante Coevolución	75
4.1.	El enfoque completo del Salón de la Fama (HoFC) y su aplicación al <i>Robot Wars</i>	77
4.1.1.	Algoritmo reducido del Salón de la Fama (HoFR)	80
4.1.2.	Algoritmo que optimiza el Salón de la Fama (HoFU)	90
4.2.	Otros enfoques de optimización del HoF y su aplicación al <i>Planet Wars</i>	104
4.2.1.	Esquema general del enfoque HoFCC	104
4.2.2.	Optimización del HoF basado en la diversidad (HoFCC-Diversity)	107
4.2.3.	Optimización del HoF basado en la calidad de las soluciones (HoFCC-Quality)	108



4.2.4.	Optimización del HoF con enfoque multiobjetivo (HoFCC-U) . . .	109
4.2.5.	Experimentos y resultados	109
4.2.6.	Configuración de los experimentos	110
4.2.7.	Análisis de los resultados	112
4.3.	Conclusiones del capítulo	117
5.	Generación simultánea de contenidos y estrategias para juegos usando	
	Coevolución	119
5.1.	Modelo propuesto: MuCCCo	121
5.1.1.	Descripción del esquema básico de MuCCCo	123
5.1.2.	Variante del modelo MuCCCo que potencia la diversidad de las soluciones (MuCCCo Div)	132
5.1.3.	Variante del modelo MuCCCo que se enfoca en la calidad de las soluciones (MuCCCo-Quality)	134
5.1.4.	Variante del modelo MuCCCo que combina las heurísticas de diversidad y calidad (MuCCCo-U)	134
5.2.	Experimentos y resultados	135
5.2.1.	Configuración de los experimentos	136
5.2.2.	Análisis de los resultados	137
5.3.	Conclusiones del capítulo	148
6.	Conclusiones	151
	Bibliografía	155
	Apéndices	173
	Apéndice A. El juego <i>Eryna</i>	175





UNIVERSIDAD
DE MÁLAGA

Acrónimos

Algunas siglas se han respetado tal y como suelen usarse en la lengua inglesa para ser consecuentes con la literatura científica.

<i>Siglas</i>	<i>Significado</i>
AE	Algoritmo Evolutivo
AG	Algoritmo Genético
CC	Coevolutivo Competitivo
CE	Computación Evolutiva
EE	Estrategia Evolutiva
FRB	Fixed Rival Bot
HoF	Hall of Fame approach
HoFR	Hall of Fame Reduced approach
HoFU	Hall of Fame Updating approach
HoFCC-Diversity	Competitive Coevolutive Hall of Fame Diversity algorithm
HoFCC-Quality	Competitive Coevolutive Hall of Fame Quality algorithm
HoFCC-Updating	Competitive Coevolutive Hall of Fame Updating algorithm
IA	Inteligencia Artificial
IC	Inteligencia Computacional
MuCCCo	Multi Content Competitive Coevolutive Hall of Fame approach
MuCCCo-Diversity	Multi Content Competitive Coevolutive Hall of Fame Diversity algorithm
MuCCCo-Quality	Multi Content Competitive Coevolutive Hall of Fame Quality algorithm
MuCCCo-Updating	Multi Content Competitive Coevolutive Hall of Fame Quality algorithm
NPC	Non-Player Character
PE	Programación Evolutiva
PCG	Procedural Content Generation
PG	Programación Genética
RTS	Real Time Strategy
3D	3-Dimensional





UNIVERSIDAD
DE MÁLAGA

Lista de Algoritmos

1.	Esquema básico de un Algoritmo Evolutivo	19
2.	Algoritmo HoFR	81
3.	Actualización del Salón de la Fama	93
4.	Cálculo del <i>fitness</i> de un individuo en el método HoFU	94
5.	Algoritmo HoFCC(α, λ)	105
6.	Algoritmo MuCCCo(α, λ)	129





UNIVERSIDAD
DE MÁLAGA

Lista de Figuras

2.1. Estructura genética de un individuo en un AE	20
3.1. Evolución del número de autores y publicaciones en la red de coautoría del área de IA en videojuegos.	40
3.2. Captura de pantalla del juego NERO	42
3.3. Personajes de los juegos <i>Fable</i> (izquierda) y <i>Beyond: Two Souls</i> (derecha) mostrando emociones	45
3.4. Open Real-Time Strategy	52
3.5. Un mapa sencillo sin obstáculos del juego <i>Robot Wars</i>	61
3.6. Un mapa con obstáculos del juego <i>Robot Wars</i>	61
3.7. Captura de pantalla de la simulación de una batalla en el juego <i>Robot Wars</i>	67
3.8. Captura de pantalla del <i>Planet Wars</i> : los planetas tienen un número que indica la cantidad de naves que hay en ese mundo. Los colores identifican la pertenencia de los planetas y las naves a uno u otro jugador.	69
3.9. Una matriz de acciones en el juego <i>Planet Wars</i> que fue generada a partir de uno de los algoritmos que presentaremos más adelante en esta memoria.	72
4.1. Cantidad de coevoluciones logradas en una ejecución del HoFR	84
4.2. Cantidad de coevoluciones logradas en una ejecución del HoFC	85



4.3. Resultados de los enfrentamientos contra un bando (el B) del HoFC (o sea, $(A \text{ vs } B)$ y $(A' \text{ vs } B)$)	86
4.4. Resultados de los enfrentamientos contra un bando (el B') del HoFR (o sea, $(A \text{ vs } B')$ y $(A' \text{ vs } B')$)	87
4.5. Prueba de transitividad en los enfoques HoFC y HoFR	88
4.6. Tiempo medio (en segundos) que tardan las ejecuciones de cada algoritmo	90
4.7. Duración del ciclo coevolutivo del HoFC y el HoFU	96
4.8. Fitness medio del enfrentamiento de los bandos A (del HoFC y el HoFU) contra el bando B del HoFC	98
4.9. Fitness medio del enfrentamiento de los bandos A (del HoFC y el HoFU) contra el bando B del HoFU	99
4.10. Tiempo medio (en segundos) que tardaron las coevoluciones del Experimento 1 en el juego <i>Robot Wars</i>	100
4.11. Fitness medio del enfrentamiento del conjunto A del HoFC y HoFU contra el conjunto B del HoFC	101
4.12. Fitness medio del enfrentamiento del conjunto A del HoFC y HoFU contra el conjunto B del HoFU	102
4.13. Tiempo medio de las coevoluciones en el Experimento 2	102
4.14. Distribución de los mejores valores de <i>fitness</i> logrados por cada algoritmo. Como es usual, cada caja representa el segundo y el tercer cuartil de la distribución, la mediana está marcada con una línea vertical y los valores atípicos se indican con un signo de '+'.	113
4.15. Distribución del <i>fitness</i> medio obtenido en cada algoritmo.	113
4.16. Distribución del número de evaluaciones empleadas por cada algoritmo.	114
4.17. Distribución del número de victorias obtenido por cada algoritmo en el torneo de <i>Todos vs. Todos</i>	115



5.1. Esquema del modelo coevolutivo multi-contenido para un juego de n -jugadores	125
5.2. Configuración del escenario de batallas para ambas poblaciones (el individuo que está siendo evaluado se resalta con líneas discontinuas)	127
5.3. Ejemplos de mapas generados con el algoritmo $\text{MuCCCo}(\alpha, \lambda)$ para el juego <i>Planet Wars</i> : tanto los planetas como las flotas son identificados por un número que muestra la cantidad de naves que poseen. Sus colores identifican el jugador al que pertenecen, el <i>FRB</i> se denota por el color verde.	130
5.4. Análisis estadístico del mejor <i>fitness</i> de la población de <i>bots</i> encontrado por cada versión del algoritmo $\text{MuCCCo}(\alpha, \lambda)$	138
5.5. Análisis estadístico del mejor <i>fitness</i> de la población de mapas encontrado por cada versión del algoritmo $\text{MuCCCo}(\alpha, \lambda)$	138
5.6. Representación en un frente de Pareto de la distribución de los mejores <i>fitness</i> de <i>bots</i> (eje x), los mejores <i>fitness</i> de los mapas (eje y), y el número de evaluaciones (eje z) logrado por cada algoritmo	140
5.7. Los mapas utilizados en el torneo de <i>Todos vs. Todos</i> , los dos primeros son mapas balanceados del conjunto de mapas originales del <i>Planet Wars</i> y los otros dos son mapas generados por los algoritmos aquí presentados.	142
5.8. Análisis estadístico de la distancia promedio de los planetas al planeta inicial del Jugador 2 para cada uno de los mapas del HoF.	143
5.9. Análisis estadístico de la Correlación de Pearson entre el “número de naves” y el “ratio de crecimiento”.	144
5.10. Análisis estadístico de la diversidad en el conjunto de mapas.	146
A.1. <i>Eryna</i> : Vista lógica de la plataforma interactiva de prueba y optimización	178
A.2. <i>Eryna</i> : Vista lógica del entorno de competición online	179





UNIVERSIDAD
DE MÁLAGA

Lista de Tablas

4.1. Parámetros usados en los experimentos	83
4.2. Parámetros del ciclo coevolutivo para el HoFCC	110
4.3. Resultados del test de Kruskal-Wallis para todos los indicadores.	114
5.1. Parámetros del ciclo coevolutivo	136
5.2. Resultados del test de Kruskal-Wallis para los indicadores.	137
5.3. Resultados del <i>Todos vs. Todos</i> en los mapas ‘30’ y ‘70’ (del conjunto original de mapas del <i>Planet Wars</i>).	141
5.4. Resultados del <i>Todos vs. Todos</i> en dos mapas elegidos de forma aleatoria de los mapas victoriosos generados por nuestros algoritmos	141
5.5. Resultados del torneo de <i>Todos vs. Todos</i> distinguiendo entre los <i>bots</i> evolucionados con un FRB (<i>GeneBot</i>) y los de tres FRBs.	147





UNIVERSIDAD
DE MÁLAGA

Capítulo 1

Introducción

It isn't that they can't see the solution.

It is that they can't see the problem.

G.K. Chesterton

La Inteligencia Artificial (IA) es uno de los campos de investigación más interesantes e importantes en el desarrollo de videojuegos. Dentro del campo de la IA aplicada a videojuegos hay muchas ramas y cada una de ellas representa un campo de investigación abierto. La más tradicional de las aplicaciones, y podríamos decir que la pionera, ha sido la de los mecanismos de IA para controlar el proceso de toma de decisiones de los *Non-Player Characters* (NPCs, por sus siglas en inglés).

En la actualidad no es usual encontrar un videojuego donde no se empleen mecanismos de IA para controlar la toma de decisiones de los *bots* (en un sentido amplio, un *bot* se refiere a un personaje del juego que no es controlado por el jugador humano). La alta demanda que experimenta la industria del videojuego ha provocado que las empresas dediquen mucho esfuerzo y presupuesto para dotar a sus juegos de mecanismos

de IA que compitan con los ya muy avanzados que tienen las producciones más conocidas en el mundo *gamer*. Conseguir *bots* que jueguen de manera similar a un usuario humano (es decir, no necesariamente ganando siempre, sino exhibiendo personalidad, creatividad, y capacidad de aprender de los errores) puede considerarse una variante del célebre Test de Turing (Turing, 1950), y plantea enormes desafíos desde el punto de vista de la IA clásica.

A medida que la industria del videojuego ha avanzado han ido apareciendo nuevas tendencias que han abierto otros campos de investigación muy prometedores. Dentro de las más destacadas se encuentran por ejemplo, los *Videojuegos Afectivos* (Sykes y Brown, 2003) que apuestan por incluir la emociones al proceso de toma de decisiones de los NPCs (Non-Player Character) que imitan el comportamiento humano (es decir, los jugadores virtuales) y generar una interacción afectiva bidireccional entre los *bots* y el jugador humano. Los resultados en este campo aún son discretos, excepto por casos aislados, la mayoría de las producciones comerciales se han conformado por simular emociones en los *bots* mediante animaciones que tienen mucho mérito artístico pero muy poco de IA. Otra de las tendencias es la de los *Videojuegos Auto-Adaptativos* (Spronck, 2005) que está muy relacionado con la anterior. Se basa también en mantener un flujo de interacción bidireccional entre el humano y el juego, pero en este caso no está centrado en estados emocionales sino que, de manera más general, se pretende que la partida se auto-adapte al jugador humano para hacer que su experiencia de juego sea única y placentera. Los avances en esta rama (y esto se aplica también para los videojuegos afectivos) han estado acotados por la limitación que existe a la hora de capturar información sobre el *estado* del jugador para poder obtener retroalimentación de lo que está generando la partida en él y saber hacia dónde debe ir encaminado el proceso de adaptación. No obstante, existen resultados muy interesantes que aunque no han explotado esta idea muy ampliamente y se han limitado a juegos con una lógica sencilla, sí que han logrado adaptar las partidas a los jugadores de forma exitosa. Más



adelante en esta memoria se explicarán con más detalles estos avances en cada campo.

Otra de las tendencias *modernas* de la IA es la Generación de Contenidos por Procedimientos (PCG), cuya esencia está inspirada en la premisa de “hacer más con menos” y su objetivo es generar de forma procedural contenido para videojuegos mediante algoritmos especializados que permitan esa reducción de esfuerzo. De ahí se desprende un reclamo para nada despreciable que es el de la reducción de costes. Hablamos de una industria que sólo en el año 2015 facturó en España 510,7 millones de euros, cifra que representa un 24% más que en el ejercicio anterior y se estima una previsión de crecimiento anual del 22,4 (hasta el 2019), lo que supondría alcanzar en 2019 los 1.140 millones de euros de facturación (DEV, 2017); donde hay una feroz competencia por sacar al mercado nuevas entregas en menos tiempo y con una buena aceptación de los usuarios.

La PCG plantea un cambio de perspectiva en lo que había sido hasta el momento el proceso de desarrollo de un videojuego, sobre todo de la fase de diseño del juego, la cual tradicionalmente dependía del trabajo manual de los diseñadores humanos e integraba a todo un equipo multidisciplinario compuesto por artistas, músicos, guionistas, animadores, modeladores, entre otros, que eran los encargados de materializar las demandas de los diseñadores generando el contenido (dígase, mapas, modelos, personajes, texturas, animaciones, sonidos, guiones, ect.) que va a componer el juego. La PCG no plantea una independencia total con ese equipo multidisciplinario sino una flexibilidad de esa dependencia, hablamos de la posibilidad de enriquecer, extender y transformar ese diseño inicial del juego mediante la creación de contenido procedural a base de algoritmos y técnicas computacionales.

Se deja entrever la cantidad de matices que genera la PCG y por ello, es de las ramas más abordadas dentro de la comunidad científica de la IA aplicada a videojuegos. Existen ya resultados tangibles de esta tendencia, incluso en juegos comerciales conocidos como es el caso del *MineCraft* y el *No Man's Sky*. En esta memoria se abordará



con detalles la PCG pues es una de las líneas en las que hemos trabajado en nuestra investigación.

Por otra parte, hay un denominador común que se mantiene constante desde los inicios del videojuego hasta la actualidad y es el empeño por incrementar la satisfacción del jugador humano. Incluso en los juegos serios, el factor “diversión del jugador humano” es una clave para el éxito. De ahí que todas las tendencias antiguas, modernas y futuras van encaminadas hacia ese objetivo primordial. Y es entonces cuando surge la pregunta de ¿cómo se alcanza ese resultado? Dicha cuestión es precisamente una de las motivaciones de esta investigación, nos hemos centrado en la creación de técnicas de IA que contribuyan a mejorar la calidad de los juegos y que vayan de la mano de conceptos muy importantes como son la optimización y la eficiencia.

La relación con la eficiencia y la optimización está presente en muchas de las técnicas que han sido creadas para implementar mecanismos de IA en videojuegos, más adelante se irán mencionando algunas a medida que se vayan explicando los trabajos relacionados con esta investigación. En particular en esta tesis se ha apostado por la Computación Evolutiva (CE) (Bäck et al., 1997) porque resulta un enfoque muy apropiado en este sentido, ya que propone métodos que son capaces de producir soluciones de gran complejidad como resultado emergente de un proceso de optimización, y su capacidad adaptativa les permite incorporar información proporcionada por el usuario y explotarla de manera no anticipada por el mismo. La CE abarca varias líneas de investigación (que se explicarán en el siguiente capítulo), nosotros nos hemos centrado en los Algoritmos Bioinspirados y más concretamente en la Coevolución (Thompson, 2005).

La Coevolución viene a formar parte de la CE como otra de las técnicas basadas en la aplicación de métodos heurísticos, inspirada en la Teoría de la Evolución Natural. Los modelos coevolutivos son muy similares a los algoritmos evolutivos tradicionales, en ellos también los individuos representan soluciones potenciales que son evolucionaria-

das mediante los operadores de transformación genética, y se someten a un proceso de búsqueda dirigido por la selección basada en el *fitness*. Pero sí que hay varios aspectos que tienen un particular tratamiento en la coevolución, por ejemplo, la evaluación. El proceso evaluativo requiere de múltiples interacciones entre los individuos. También la exploración de nuevos enfoques de las nociones de cooperación y competición, es una característica especial de los modelos coevolutivos, incluso algunos aspectos de la representación de las soluciones pueden ser únicos de la Coevolución. Existen escenarios en los que puede ser útil aplicar la Coevolución, por ejemplo, cuando existen espacios de búsqueda grandes y complejos, y es difícil definir una medida de calidad objetiva; cuando no existe una medida de calidad definida, es decir, se puede comparar pero no cuantificar; o cuando el espacio de búsqueda tiene cierta estructura, cuya descomposición se puede explotar.

En particular, se puede considerar que los videojuegos tienen una predisposición favorable para aplicarles un enfoque coevolutivo debido a su naturaleza competitiva. De hecho, como veremos en capítulos posteriores, han sido escenario de la aplicación de varios modelos coevolutivos con resultados exitosos aunque también han puesto al descubierto varios inconvenientes causados por algunas patologías intrínsecas de los enfoques coevolutivos que son difíciles de resolver y plantean un reto a la comunidad científica. Motivado por esto, parte del trabajo de esta tesis ha estado enfocado en la creación de nuevos algoritmos que permitan obtener buenos resultados en modelos coevolutivos competitivos y reduzcan el impacto de esas patologías coevolutivas.

En esta investigación nos hemos centrado en un género particular: los videojuegos de Estrategia en Tiempo Real (RTS, por sus siglas en inglés). En este tipo de juego, los jugadores gestionan un equipo (o ejército) el cual lo conforman un número de (no necesariamente iguales) unidades (por ejemplo, soldados, trabajadores, ...) y cada jugador tiene normalmente dos objetivos principales: (a) destruir las fuerzas del oponente y (b) crear estructuras adicionales con un fin específico (por ejemplo, la construcción



de edificios para proteger sus unidades o defender una posición determinada en el mapa). Al inicio del juego se cuenta con un número limitado de recursos y a medida que avanza la partida el jugador debe ir optimizándolos. Así que un juego RTS puede ser visto como un juego de recolección u optimización de recursos. En este contexto, una IA normalmente se dirige a gestionar el comportamiento (es decir, el mecanismo de toma de decisiones) de un jugador virtual (o sea, un jugador artificial que simula un oponente humano). Este género en particular ofrece una amplia variedad de problemas muy interesantes en el ámbito de la investigación de la IA, tales como la planificación de tareas y recursos en tiempo real teniendo en cuenta al adversario y la toma de decisiones con incertidumbre, entre otros. Precisamente por estas razones, los juegos RTS han sido de los más empleados como entornos de experimentación en la investigación de nuevas técnicas de IA (Lara-Cabrera et al., 2013c).

A continuación se resumen los objetivos principales que han guiado esta tesis.

1.1. Objetivos

Esta investigación ha estado dirigida hacia el cumplimiento de cinco objetivos que se describen seguidamente.

- Estudiar las principales tendencias en la aplicación de la IA en videojuegos, haciendo un resumen de los trabajos más relevantes en esos temas y valorando cada una de ellas según nuestra propia percepción. Este estudio servirá para centrar la investigación en apoyar estas líneas que son las que prometen guiar el desarrollo de los videojuegos en el futuro próximo.
- Estudiar y analizar el estado del arte de la aplicación de la Computación Evolutiva en la programación de IA para videojuegos, haciendo énfasis en el uso de la Coevolución. Se analizaron muchos trabajos relacionados y esto permitió identificar aquellas líneas de investigación que estaban abiertas a nuevas propuestas.

Se constató que la Coevolución ha sido ampliamente aplicada en la solución de problemas de optimización en videojuegos y ha mostrado resultados exitosos. Sin embargo, también es cierto que, la dinámica que genera un proceso coevolutivo ha motivado varias investigaciones para determinar los factores que a veces desatan el caos en estos modelos. Estos factores también son parte de nuestro estudio de campo.

- Analizar el comportamiento de los métodos de archivos en los modelos coevolutivos competitivos. Posteriormente, y partiendo de este análisis, nuestro objetivo es mejorar las propuestas existentes. Para ello nos planteamos diseñar e implementar algoritmos novedosos que logren optimizar su funcionamiento en ciclos coevolutivos cuyo fin es la generación de estrategias de IA para videojuegos RTS.
- En este punto de la tesis, y una vez alcanzados los objetivos anteriores, nos planteamos el abrir nuevas líneas de investigación y romper barreras existentes en la generación procedural/automática de contenidos en videojuegos. Una de estas barreras establece que la PCG debe ser usada para generar un tipo de contenido específico en los videojuegos (normalmente las estrategias de comportamiento de los *bots* o el diseño de otro tipo de recursos en el juego tales como mapas o armas). Nosotros hemos ido un paso más allá y proponemos el diseño de un modelo coevolutivo competitivo totalmente novedoso que permite la generación automática y simultánea de estrategias de IA y contenido para videojuegos. El objetivo que nos establecimos consistía en diseñar esta propuesta e implementarla para validar su viabilidad.
- Desarrollar experimentos sobre juegos RTS con el objetivo de analizar y evaluar los resultados de los algoritmos diseñados, apoyándonos en métodos estadísticos que nos permitan extraer conclusiones serias y generales sobre esta investigación.

Queremos subrayar que todos los algoritmos diseñados en esta tesis son completa-

mente novedosos y suponen pues una contribución a la comunidad científica de la aplicación de técnicas de inteligencia computacional/artificial en los videojuegos. Algunos de ellos, como ya hemos mencionado anteriormente, abren nuevas líneas de investigación dirigidas a la generación automática (o procedural) y paralela de contenido de distintas naturaleza para los videojuegos. En este sentido, no pretendemos reemplazar al artista gráfico, al programador o al diseñador, sino que nuestro enfoque consiste en proporcionarles una herramienta que les facilite su labor o pueda inspirarles.

1.2. Contribuciones

Las primeras contribuciones de esta investigación se exponen en los Capítulos 2 y 3. Ambos proporcionan un análisis con cierto detalle de la Coevolución como técnica de optimización y su aplicación en el universo del desarrollo de videojuegos para la generación automática de contenido.

El Capítulo 4 inicia la descripción de nuestras propuestas de algoritmos que usan la Coevolución Competitiva (CC) como mecanismo de auto-aprendizaje. El aporte principal de esta parte de la memoria es el análisis que se hace del funcionamiento de los algoritmos CC para la generación de estrategias de IA en dos juegos de tipo RTS; y de forma más particular el estudio de la eficiencia y la eficacia que puede aportar un mecanismo de manejo de memoria a largo plazo dentro del ciclo coevolutivo.

Por otra parte, en el Capítulo 5 se definirá un nuevo modelo CC el cual es un intento de implementar la coevolución interespecífica, es decir, entre especies distintas (salvando las distancia con los procesos coevolutivos que ocurren en la naturaleza). En él se manejarán (dos) especies que son intrínsecamente diferentes (respecto a la naturaleza de sus dominios). Una de ellas agrupa a los individuos de tipo *bot* cuya codificación se corresponde con la de una estrategia de comportamiento para guiar a un jugador virtual y el otro conjunto representa el contenido del juego (que podrían ser mapas o niveles, por ejemplo). Nótese que la relación entre las poblaciones mencionadas (es decir,



bots y contenido) no se asemeja con el contexto típico de dos especies competidoras como *depredador-presa* o *parásito-huésped*, en la que el *fitness* de cada individuo está directamente relacionado con el de su competidor y en el cual, de forma natural, una mejora en un individuo supone empeorar a su competidor y viceversa (Floreano et al., 1998). Por lo tanto, la manera en que se va a establecer la competición entre el contenido del juego y los jugadores virtuales supone un reto importante en esta propuesta y hace que sea novedosa en el campo de la IA para juegos.

El principal aporte de este modelo es la capacidad de permitir la generación de IA para juegos (es decir, estrategias de juego que rigen el comportamiento de los jugadores virtuales), y a la misma vez, crear contenido que se adapte dinámicamente para alcanzar objetivos específicos de diseño. Imaginemos, por ejemplo, un mapa/nivel que se adapta para favorecer a los jugadores novatos frente a los más experimentados, de manera que todos ellos puedan disfrutar de la partida (es decir, que la experiencia de juego no resulte ni demasiado fácil para el jugador habilidoso, ni muy difícil para los menos experimentados). Desde la perspectiva del diseñador, nuestra propuesta puede ser vista como un mecanismo para la adaptación dinámica del juego de acuerdo con el perfil de un jugador específico a través de PCG. Hasta lo que nuestro conocimiento alcanza, esta es la primera vez que un algoritmo permite generar de forma automática el contenido de un juego y sus estrategias de IA simultáneamente; y consideramos que esto constituye un paso importante hacia la creación de juegos completos a nivel de diseño mediante procedimientos algorítmicos.

Es importante destacar que nuestro sistema coevolutivo ha sido diseñado con un enfoque genérico que lo hace aplicable a cualquier género de juego que promueva la competición entre los jugadores. En el Capítulo 5 se explicará una prueba de concepto que fue realizada para demostrar la viabilidad de la propuesta, usando un juego RTS como entorno experimental. Adicionalmente se podrá comprobar que es posible generar contenido del juego que se adapte a objetivos específicos de diseño, además de estra-



tegrías de IA capaces de vencer a algunos de los mejores *bots* descritos en la literatura científica para el juego de prueba empleado.

Otra de las contribuciones adicionales de esta investigación se explica en el Apéndice A. Se trata de un videojuego que ha sido creado en el marco de esta investigación. Su nombre es *Eryna* (Nogueira et al., 2014b), y consiste en una herramienta de apoyo a la investigación, similar a las que existen en la literatura científica (y que serán discutidas en el Capítulo 3). Su principal distinción es que en ella se han integrado módulos que facilitan la aplicación de algoritmos que están en sintonía con las nuevas tendencias de la IA aplicada a videojuegos.

Esta herramienta se ha descrito en un apéndice para diferenciarla claramente de nuestras propuestas algorítmicas desarrolladas a lo largo de la tesis. *Eryna* no ha sido publicitada (a nivel científico) de forma internacional aunque lo será con posterioridad a la defensa de esta tesis. Esta razón también contribuye a que la presentemos en un apéndice y no en el texto principal. No obstante, queremos remarcar que ha supuesto un duro trabajo de diseño e implementación.

A continuación se resumen las principales contribuciones de la tesis.

- Definición de un modelo Coevolutivo Competitivo (CC) que optimiza el uso de los métodos de archivo tradicionales de la Coevolución.
- Diseño de dos métricas basadas en el control de la diversidad y la calidad que pueden ser aplicadas a cualquier algoritmo CC que utilice métodos de archivos.
- Diseño e implementación de un algoritmo coevolutivo novedoso que aplica las métricas antes mencionadas desde tres enfoques distintos y que permite generar estrategias de IA en videojuegos de tipo RTS.
- Diseño e implementación de un modelo coevolutivo nuevo que permite la generación automática y simultánea de estrategias de IA y contenidos para videojuegos.

Este es la contribución más novedosa de esta tesis pues, según nuestros conocimientos, es la primera vez que se logra la generación simultánea de contenidos de distinta naturaleza.

- Diseño e implementación de un videojuego que servirá de apoyo a la investigación de las nuevas tendencias de la IA aplicada a videojuegos; y en especial de las técnicas de PCG que permitirán en un futuro la generación completa de un videojuego a nivel de diseño.

Como resultado del proceso de divulgación científica de esta investigación, se han publicado varios artículos en revistas y congresos especializados, tanto en España como a nivel internacional. Los detalles se muestran a continuación.

1.3. Publicaciones que avalan esta investigación

1.3.1. Revistas indexadas en el *Journal Citation Report (JCR)*

1. Mariela Nogueira Collazo, Carlos Cotta, Antonio J. Fernández: Virtual player design using self-learning via competitive coevolutionary algorithms. *Natural Computing*. Springer Berlin/Heidelberg. Volumen 13(2) (2014), p. 131-144. Factor Impacto 0.757, Ranking 96/123 (Computer Science, Artificial Intelligence), 65/102 (Computer Science, Theory & Methods), Tercil T2, Cuartil Q3.
2. Mariela Nogueira Collazo, Carlos Cotta, Antonio J. Fernández: Competitive Algorithms for Coevolving Both Game Content and AI. A Case Study: Planet Wars. *IEEE Transactions on Computational Intelligence and AI in Games*, Volumen 8(4) (2016), p. 325-337. Factor Impacto 1,00, Ranking 53/106 (Computer Science, Software Engineering), Tercil T2, Cuartil Q2.



1.3.2. Revista indexada en el *Emerging Sources Citation Index*

1. Raúl Lara-Cabrera, Mariela Nogueira Collazo, Carlos Cotta, Antonio J. Fernández Leiva: Procedural Content Generation for Real-Time Strategy Games. *International Journal of Interactive Multimedia and Artificial Intelligence (IJIMAI)* Volumen 3(2) (2015), p. 40-48.

1.3.3. Congresos internacionales

1. Mariela Nogueira Collazo, Carlos Cotta: Aplicación de algoritmos genéticos coevolutivos en la programación de videojuegos de estrategia. *V Congreso Internacional de Tecnologías, Contenidos Multimedia y Realidad Virtual* (Informática 2011), ISBN: 978-959-7213-01-7, La Habana, Cuba, Febrero 2011.
2. Mariela Nogueira Collazo, Carlos Cotta, Antonio J. Fernández Leiva: On Modeling, Evaluating and Increasing Players' Satisfaction Quantitatively: Steps towards a Taxonomy. *Applications of Evolutionary Computation - EvoApplications 2012 (EvoGames)*. Springer, Lecture Notes in Computer Science, Vol. 7248, p. 245-254, Málaga, España, Abril 2012.
3. Mariela Nogueira, Juan M. Gálvez, Carlos Cotta, Antonio J. Fernández-Leiva: Hall of Fame based competitive coevolutionary algorithms for optimizing opponent strategies in a new RTS game. *13th annual European Conference on Simulation and AI in Computer Games (GAMEON'2012)*, Eurosis, p. 71-78, Málaga, España, Noviembre 2012.
4. Mariela Nogueira Collazo, Carlos Cotta, Antonio J. Fernández Leiva: An Analysis of Hall-of-Fame Strategies in Competitive Coevolutionary Algorithms for Self-Learning in RTS Games. *Learning and Intelligent Optimization - 7th International Conference (LION)*. Springer, Lecture Notes in Computer Science, Vol. 7997, p. 174-188, Catania, Italia, Enero 2013.



1.3.4. Congresos nacionales

1. Mariela Nogueira Collazo, Carlos Cotta, y Antonio J. Fernández-Leiva: Apuntes del estudio sobre modelado, evaluación e incremento de la satisfacción del jugador. *VIII Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB 2012)*, Albacete, España, p. 699-705, Febrero 2012.
2. Mariela Nogueira Collazo, Antonio J. Fernández Leiva, Carlos Cotta Porras: Eryna: una herramienta de apoyo a la revolución de los videojuegos. *Primer Congreso de la Sociedad Española para el Videojuego (CoSECivi 2014)*, CEUR Workshop Proceedings, Vol. 1196 p. 173-184, Barcelona, España, Junio 2014.
3. R. Lara-Cabrera, M. Nogueira, C. Cotta, y A. J. Fernández-Leiva, Optimización en videojuegos: retos para la comunidad científica. *X Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB 2015)*, p. 463-470, Mérida, España, Febrero 2015.
4. Raúl Lara-Cabrera, Mariela Nogueira Collazo, Carlos Cotta, Antonio J. Fernández Leiva: Game Artificial Intelligence: Challenges for the Scientific Community. *Segundo Congreso de la Sociedad Española para el Videojuego (CoSECivi 2015)*, CEUR Workshop Proceedings, Vol. 1394p. 1-12, Barcelona, España, Junio 2015.
5. Alejandro Ruiz-Moyano, Mariela Nogueira Collazo, Antonio J. Fernández Leiva: Hacia la Generación Automática de Mecánicas de Juego: un Editor de Reglas para Eryna. *Tercer Congreso de la Sociedad Española para el Videojuego (CoSECivi 2016)*, CEUR Workshop Proceedings, Vol. 1682, p. 125-134, Barcelona, España, Junio 2016.



1.4. Estructura de la memoria

Esta memoria se ha organizado de la siguiente manera. Después de este capítulo introductorio le sigue el Capítulo 2 que está enfocado a explicar el marco teórico de los principales conceptos que están relacionados con la Computación Evolutiva y la Coevolución. El Capítulo 3 describe brevemente los trabajos relacionados con nuestra investigación mediante un análisis del estado del arte de la aplicación de la IA en videojuegos. Posteriormente, se da paso a los dos capítulos que muestran las propuestas de algoritmos realizadas en esta investigación y más tarde, se resumen las conclusiones del trabajo en el Capítulo 6. Luego se incluye un apéndice en el cual se describe el juego *Eryna* que ha sido creado en el marco de esta investigación.

Capítulo 2

Computación Evolutiva y Coevolución

*Look deep into nature, and then
you will understand everything better.*

Albert Einstein

La Computación Evolutiva (CE) puede considerarse más como una estrategia de resolución de problemas que como una herramienta de talla única (Bäck et al., 1997; Michalewicz y Fogel, 2004). Del mismo modo que la naturaleza selecciona las especies mejor adaptadas para perdurar y reproducirse, la CE intenta imitar el proceso biológico de la evolución con el fin de resolver un problema dado (Fogel y Michalewicz, 2001). Enfoca sus mecanismos resolutivos en la simulación de la selección natural y la evolución de las especies como un proceso de aprendizaje mediante el cual los individuos adquieren distintas características y se vuelven más aptos para sobrevivir.

Ha pasado más de medio siglo desde los primeros trabajos científicos publicados



en este campo. Uno de los pioneros en abordar el tema fue Alan M. Turing quien hacía referencia a una conexión entre la evolución y el aprendizaje automático en un artículo publicado a mediados del siglo XX (Turing, 1950). A finales de la década del 50 y principios de los años 60 se comenzaron a publicar los primeros resultados prometedores (Friedberg, 1958; Barricelli, 1962). El auge de la investigación científica en este campo no se ha hecho esperar, numerosos trabajos han sido publicados y se han ido distinguiendo distintos campos de aplicación. En este capítulo se explicarán los Algoritmos Evolutivos, sus bases biológicas y el esquema básico de su funcionamiento. También se abordarán las principales ramas de investigación que han surgido en el campo de la CE. Y la última sección estará dedicada a la Coevolución como un enfoque muy interesante dentro la CE y en el cual se centra todo el trabajo investigativo que se expone en esta memoria.

2.1. Bases teóricas

2.1.1. Conceptos y bases biológicas de los Algoritmos Evolutivos

La teoría evolutiva propuesta por Charles Darwin ha generado desde sus inicios una amplia polémica entre los estudiosos del tema, y de ahí que haya continuado siendo objeto de muchas investigaciones posteriores. La combinación de dicha teoría con otros planteamientos como el del *Seleccionismo* de August Weismann, la *Herencia de caracteres* de Gregor Mendel entre otros, ha devenido en lo que se conoce como el paradigma *Neo-Darwiniano*. Esta teoría afirma que la historia de la gran mayoría de la vida en nuestro planeta puede ser explicada a través de una serie de procesos estadísticos que actúan sobre y dentro de las poblaciones y especies (Rose, 1990): la reproducción, mutación, competencia y selección.

En general, se les conoce como Algoritmos Evolutivos (AEs) (Bäck y Hoffmeister, 1991) a los métodos de búsqueda y optimización que se basan en los paradigmas de la



evolución natural. Si se analizan desde un punto de vista matemático, los AEs son métodos numéricos que permiten explorar un espacio de búsqueda y obtener soluciones cuyo desempeño estará lo más cerca posible de un valor óptimo predefinido. Computacionalmente hablando, estos algoritmos son programas informáticos cuya implementación se basa en un modelo iterativo que va evolucionando un conjunto de posibles soluciones y seleccionando las más adecuadas según criterios de calidad definidos previamente. De la misma forma que en la naturaleza la evolución y la selección natural van fomentando la aparición de individuos mejor adaptados, los AEs encaminan su proceso de búsqueda hacia el hallazgo de soluciones evolucionadas que estén lo más cerca posible a un prototipo óptimo.

Se puede resumir que un AE está compuesto por:

- Un conjunto de individuos que representan la solución que se pretende encontrar, y que es la imitación directa de una población real de la naturaleza.
- Un mecanismo de selección que permita escoger quiénes son los individuos más aptos que van a reproducirse y sobrevivir de una generación a otra.
- Mecanismos de cambios que son los que permitirán simular el proceso de transformación constante que ocurre en la evolución biológica y que da lugar a nuevos individuos mejor adaptados.

Una explicación más detallada de cómo los AEs simulan la evolución biológica será suministrada en los que resta de esta sección. Comencemos por ver a continuación el esquema básico de funcionamiento de estos algoritmos.

Esquema básico

El esquema básico de funcionamiento de un AE se describe en pseudolenguaje en el Algoritmo 1. Nótese que como se mencionaba anteriormente en esta sección, el punto de partida es un conjunto de individuos que forman la población. Más adelante



se explicarán las pautas fundamentales de la definición y representación de dichos individuos. En la primera línea del Algoritmo 1 se inicializa un contador que permitirá controlar el número de generaciones que se crearán a partir de la población inicial. La definición de la población es el segundo paso expuesto en el esquema, se puede hacer de forma aleatoria, o emplear un mecanismo particular para la generación de los individuos, no existe una fórmula preestablecida. En la Línea 3 se da inicio al ciclo evolutivo en el cual se irá transformando la población mientras no se cumpla la condición de parada, que puede estar determinada por un valor máximo de generaciones o un umbral de calidad de los individuos evolucionados.

Para llevar a cabo el proceso de evaluación, es necesario definir una función matemática que permita calcular para cada individuo la proximidad de dicha solución al óptimo, este valor se conoce como *fitness*. Luego de la evaluación se procede a elegir los individuos que van a reproducirse para generar la siguiente generación, esta reproducción se lleva a cabo simulando los mecanismos de recombinación genética que ocurren en la vida real como el *cruce* y la *mutación*. Existen muchos métodos de cruce predefinidos que se pueden aplicar o también se puede definir alguno particular dependiendo de las características del problema simulado. También se suelen utilizar métodos de mutación para provocar aún más cambios en los descendientes que se van creando. Por último, los nuevos individuos se van insertando en la población original, bien sustituyéndola por completo o mezclando individuos de una y otra generación, a este proceso se le conoce como *reemplazo*.

Codificación de la solución

En un AE los individuos de la población representan las posibles soluciones o los candidatos potenciales a formar parte del conjunto solución del problema de optimización planteado. Es por esto que la forma de codificar dichas soluciones o candidatos depende en gran medida del problema en cuestión. Dicha codificación es un aspecto



Algoritmo 1: Esquema básico de un Algoritmo Evolutivo

```

1  $i \leftarrow 0$ ; // Contador de generaciones
2  $pop[i] \leftarrow Inicializar()$ ; // Se inicializa la población en la iteración  $i$ 
3 while NOT CriterioParada do
4     EVALUAR ( $pop[i]$ ); // Evaluación de la población en la generación  $i$ 
5      $pop'[i] \leftarrow SeleccionarProgenitores (pop[i])$ ;
6      $pop''[i] \leftarrow AplicarVariaciones (pop'[i])$ ; // Aplicar operadores
        genéticos
7     Evaluar ( $pop''[i]$ );
8      $pop[i + 1] \leftarrow AplicarReemplazo (pop[i], pop''[i])$ ;
9      $i \leftarrow i + 1$ ;
10 end while

```

muy importante para el éxito del proceso, pues de él depende que las soluciones obtenidas contemplen todos los requisitos del problema real que se pretende resolver. Algunos de los conceptos principales que hay que tener en cuenta para definir la codificación son (Bäck, 1996):

- Cromosoma o genotipo: se trata de una cadena codificada (números binarios, valores enteros, flotantes, etc.) que representa los parámetros del problema planteado.
- Individuo: uno o más cromosomas que tienen asociados un valor de *fitness*, representa un posible candidato a ser una solución del problema.
- Gen: la versión codificada de cada parámetro del problema a resolver, en su conjunto forman la cadena del cromosoma.
- Alelo: es cada uno de los valores del gen.
- Fenotipo: suele asociarse al valor de *fitness* del genotipo, representa la expresión del genotipo en su entorno.

En la Figura 2.1 se ha empleado una codificación binaria. Cada gen representará una característica o parámetro de la posible solución y el número binario que le sea



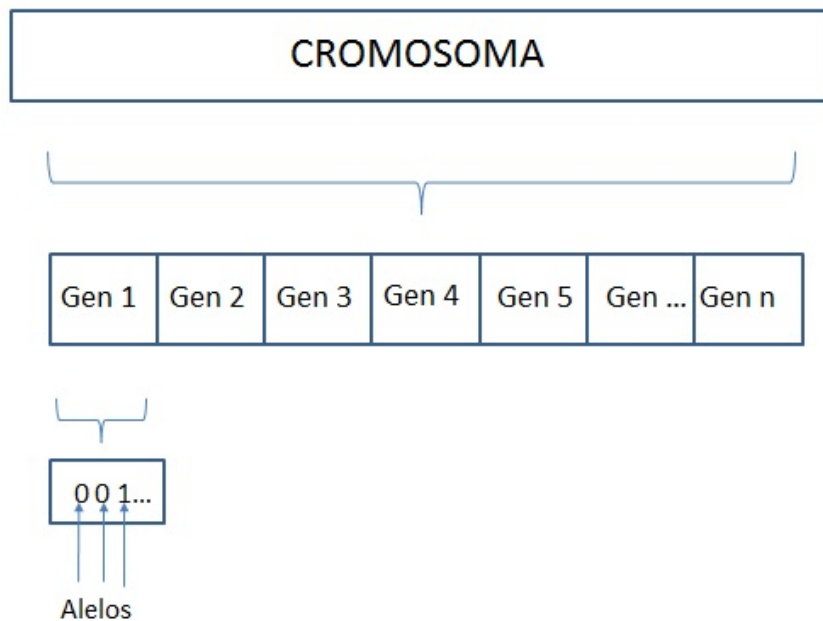


Figura 2.1: Estructura genética de un individuo en un AE

asignado indicaría su valor a la hora de evaluar mediante la función del *fitness* la calidad de dicha solución. La codificación, al igual que la función de evaluación, son dos elementos muy importantes del algoritmo y deben ser definidas con mucha precisión para que puedan representar correctamente todo el alcance del problema planteado y permitan al algoritmo explorar sobre un espacio de búsqueda que tenga una alta probabilidad de contener una solución que al menos esté cercana al óptimo deseado.

Tamaño de la población

No existe una regla general para definir el tamaño de la población, el valor que se escoja debe de ser suficiente para garantizar la diversidad de las soluciones y permitir la explotación del espacio de búsqueda. Hay investigadores que afirman que un tamaño “pequeño” de la población podría guiar la búsqueda del algoritmo hacia soluciones pobres (Pelikan et al., 2000) y que un tamaño de población “grande” podría hacer que el algoritmo aumentara sustancialmente el tiempo de cálculo en la búsqueda de un

solución (Lobo y Goldberg, 2004). En la práctica este valor puede llegar a ser difícil de estimar y depende mucho de cada problema en particular, es por eso que quizás el método más común para definir el tamaño la población inicial es el método empírico (Eiben et al., 1999) en el que el algoritmo es probado con varios tamaños de poblaciones y el que da mejores resultados es el que se escoge.

De forma general, la configuración de los parámetros de un AE ha sido objeto de una amplia investigación por la comunidad científica. Se ha prestado especial atención al auto-ajuste de estos valores dentro del funcionamiento del propio algoritmo (Eiben et al., 2004; Eiben et al., 1999) como una alternativa a la elección de valores estáticos. Para el caso particular de la cantidad de individuos, no hay duda de que existe una relación muy clara con la eficiencia del algoritmo, pues este parámetro es directamente proporcional al consumo computacional que demanda la fase de evaluación. Así, si el número de generaciones es constantes y tenemos un AE generacional, cuantos más individuos hayan, más evaluaciones se tendrán que hacer. Y este es un punto importante que también se tiene en cuenta a la hora de definir el tamaño de la población.

Población inicial

La generación de la población inicial de un AE suele plantearse a partir de un mecanismo aleatorio que crea los individuos con la codificación que haya sido previamente diseñada. La idea es que todos los individuos de la población compitan en igualdad de condiciones y que se inicie la búsqueda con un conjunto de individuos diversos. Una vez generada la población inicial, en algunos problemas, se puede realizar una criba de individuos no factibles. Este tratamiento implica la regeneración de aquellos individuos cuyas características los conviertan en no válidos para resolver el problema.

Hay estudios que plantean que si la población inicial es buena el algoritmo tiene mayores posibilidades de encontrar una buena solución (Burke et al., 2004), y por otra parte si el conjunto inicial de individuos no es lo suficientemente bueno, entonces



será difícil obtener un proceso de búsqueda exitoso (Lobo y Lima, 2005). Algunos autores sugieren que la diversidad es fundamental para garantizar el buen desempeño del algoritmo y evita la convergencia prematura hacia óptimos locales (Leung et al., 1997).

Función de evaluación

No hay duda de que uno de los elementos claves del algoritmo es la función que permite calcular el *fitness* de un individuo. Ese valor indica el grado de adaptación al medio, que es a su vez la calidad de la solución representada. La evaluación se podría calificar como el eje principal del proceso evolutivo pues representa la heurística que va guiando la búsqueda hacia la solución del problema. Su definición puede ser un trabajo complejo pues se tienen que incluir en ella todos los factores que determinan el nivel de adecuación de un individuo en su entorno, y podrían no ser pocos; incluso, algunos son complicados de cuantificar con un valor objetivo y hay estudios que proponen incluir criterios subjetivos en los procedimientos de evaluación (Watson et al., 2001).

La función de evaluación debe ser capaz de penalizar a aquellos individuos que representen soluciones no factibles. También es conveniente que para dos individuos que se encuentren cercanos en el espacio de búsqueda, sus respectivos valores en las funciones objetivo estén próximos. A la hora de simular un problema de optimización, es muy usual encontrarse con casos en los que existen varios criterios para determinar la calidad de las soluciones e incluso, algunos de ellos son divergentes entre sí. En estos casos es conveniente utilizar el enfoque multiobjetivo que nos aportan los conocidos Algoritmos Evolutivos Multiobjetivos (Zitzler y Thiele, 1999).

Selección

Después de realizado el proceso de evaluación, viene la etapa de seleccionar a los individuos que se van a reproducir para generar la población descendiente. El valor

de *fitness* es el que rige el proceso selectivo, el objetivo es que aquellos individuos que obtuvieron un mejor resultado tengan mayor probabilidad de ser elegidos, para que sus características sean heredadas y mejoradas por la descendencia. Existen varios métodos ya definidos para llevar a cabo la selección, se podría afirmar que dos de los más utilizados (Sarma y Jong, 1997) son:

- Selección por ruleta: es un método en el que la probabilidad de que un individuo sea seleccionado es proporcional a su ventaja de *fitness* respecto al resto de los miembros de la población.
- Selección por torneo: se eligen subgrupos de individuos, los miembros de cada subgrupo compiten entre ellos y sólo se selecciona un individuo de cada uno.

Métodos de cambio

Los AEs utilizan mecanismos de transformación para generar nuevos individuos a partir de los anteriores. Según el tipo de algoritmo (se especificarán en la Sección 2.1.2), se aplicarán los métodos de cambios más adecuados. Dichos métodos están inspirados en dos procesos que ocurren en la naturaleza: la recombinación genética o cruce y la mutación. El operador de recombinación crea n descendientes a partir de p padres, aunque normalmente se cruza una pareja de progenitores para producir uno o dos descendientes. El cruce se basa en combinar la información genética de los padres, intercambiando genes entre ellos, en este proceso interviene un parámetro conocido como la tasa de recombinación p_c que indica la probabilidad de cruce en el algoritmo.

Por otra parte, la mutación, también tiene como objetivo variar la información genética de los cromosomas, se basa en alterar el valor de uno o varios de los genes. Del mismo modo que sucede en la naturaleza, esas alteraciones genéticas pueden dar lugar a mejoras muy interesantes o a consecuencias especialmente graves en los individuos mutados. Es por eso que este método de variación se debe usar de forma cautelosa. No obstante, en el contexto de los AEs, es muy usual emplearlo para fomentar la



diversidad en la población de individuos y ampliar el alcance de la exploración del espacio de búsqueda.

Tanto para la recombinación genética como para la mutación, existen varios métodos ya definidos en la literatura que se pueden utilizar según convenga, o también se puede innovar creando un método propio que se ajuste mejor a la codificación diseñada. La mayoría de las librerías evolutivas (por ejemplo, *JGAP*¹, *ECJ*², *Mallba*³) que están disponibles para facilitar la programación de AEs en diferentes lenguajes incluyen los operadores de cambios más utilizados.

Mecanismo de reemplazo

Luego de aplicar los métodos de cambio sobre la población seleccionada se emplea algún mecanismo de reemplazo que permita obtener una única población (que será la que sobreviva a la nueva generación) a partir de la población inicial (o población de padres) y la nueva (población de hijos). Para llevar a cabo el reemplazo existen varios criterios definidos en la literatura científica, y se aplican uno u otro según las necesidades del algoritmo que estemos implementando. Los nuevos individuos se pueden ir insertando en la población original, bien sustituyéndola por completo o mezclando individuos de una y otra generación.

En los modelos generacionales, la nueva población reemplaza a la anterior, de esta manera, en cada iteración se trabaja con una población nueva. Por su parte, los modelos estacionarios conservan individuos de la población padre. Para esto es usual aplicar *Elitismo*, lo cual consiste en mantener a los mejores individuos en esa población original. Como ejemplos de los métodos que existen para los modelos estacionarios tenemos el enfoque de “Reemplazar al peor de la población” (RW) y el “Torneo Restringido” (RTS) (Lozano et al., 2008).

¹<http://jgap.sourceforge.net/>

²<http://cs.gmu.edu/~eclab/projects/ecj/>

³<http://neo.lcc.uma.es/mallba/easy-mallba/>



2.1.2. Ramas de la Computación Evolutiva

Anteriormente se explicaron de forma breve los principios de funcionamiento de los AEs, los cuales proponen una metáfora biológica de la evolución que es común a todas las técnicas que se agrupan dentro de la Computación Evolutiva (CE). Son cuatro los paradigmas clásicos que se distinguen dentro del campo de la CE: Programación Evolutiva (PE), Algoritmos Genéticos (AGs), Estrategias Evolutivas (EEs) y Programación Genética (PE). En esta sección se explican cada uno de ellos.

Programación Evolutiva

Lawrence J. Fogel propuso en el año 1966 la técnica denominada Programación Evolutiva (Fogel et al., 1966). Este paradigma enfatiza los nexos de comportamiento entre padres e hijos, en vez de simular operadores genéticos específicos (como hacen los AGs). La PE es una abstracción de la evolución al nivel de las especies, por lo que no se requiere el uso de un operador de recombinación (diferentes especies no se pueden cruzar entre sí). La codificación de la solución se realiza usando números reales. El esquema básico de funcionamiento sigue las pautas principales de los AEs: generar aleatoriamente una población inicial; aplicar métodos de transformación en los individuos (sólo mutación); se calcula el *fitness* de cada hijo y se aplican técnicas de selección probabilística para determinar qué individuos permanecerán en la población.

Algoritmos Genéticos

Estos algoritmos fueron definidos por Holland en los años 60 (Holland, 1975) como una “abstracción de la evolución biológica”. Su estrategia se basa en simular el proceso de evolución como una sucesión de frecuentes cambios en los genes, con soluciones análogas a cromosomas. Las posibles soluciones del problema son representadas mediante cadenas de caracteres y el espacio de soluciones posibles es explorado aplicando transformaciones a estas soluciones candidatas mediante cruce y mutación. La

recombinación permite mezclar la información genética de los padres y traspasarla a los descendientes. Mientras que la mutación introduce un cierto factor de innovación en la información de los individuos de la población. Es usual que la población inicial se genere a partir de un mecanismo aleatorio y el proceso de búsqueda se ejecuta mientras no se cumpla una condición de parada que suele ir enfocada a no superar un número máximo de evaluaciones o a alcanzar el umbral de calidad deseado.

En general son métodos muy flexibles pues pueden adoptar con facilidad nuevas ideas que surjan dentro del campo de la CE. Además, se pueden combinar fácilmente con otros paradigmas y enfoques, incluso con alguno que no tenga relación con la CE.

Estrategias de Evolución

Las EEs (Rechenberg, 1984) emplean vectores de números reales con desviaciones estándares para codificar las posibles soluciones. Utilizan recombinación, mutación y selección que puede ser determinista o probabilística. Los operadores de recombinación pueden ser: *sexuales*, si el operador actúa sobre 2 individuos elegidos aleatoriamente de la población de padres; o *panmíticos*, cuando se elige un solo padre al azar y se mantiene fijo mientras se escoge al azar un segundo padre para cada componente de sus vectores. En el proceso de búsqueda se eliminan las peores soluciones de la población y no genera copia de aquellos individuos con un *fitness* por debajo del valor promedio.

Programación Genética

La base biológica de la PG (Koza, 1993) es la misma que la de los AGs. La diferencia entre una técnica y otra consiste en la forma de codificación de los problemas, lo que hace que su dominio de aplicación sea mayor que el de los AGs. En PG los individuos de la población son programas de ordenador, la codificación de la solución se realiza en forma de árbol, estableciendo una relación de jerarquía entre los nodos de la estructura. Para explorar el espacio de búsqueda se emplean operadores de variación como el cruce,



la mutación y la reproducción sin cambios, además de mecanismos de selección. Para calcular el *fitness* de un individuo, se ejecuta ese programa en distintos contextos y se comprueba cómo de correcta es la salida en cada uno.

Otros tipos de Algoritmos Bioinspirados

Como se ha visto hasta el momento, la naturaleza ha inspirado a muchos investigadores de disímiles maneras. Además de los AEs existen otras técnicas que están también inspiradas en procesos de la vida real y se usan como mecanismos de búsqueda y optimización en diferentes contextos. Dentro de los más conocidos están los inspirados en la inteligencia de enjambres (p.e. bandadas de aves, bancos de peces, colonias de hormigas, etc.) como son los Algoritmos de Colonias de Hormigas (Dorigo, 1992), los de Enjambres de Partículas (Kennedy y Eberhart, 1995), los que simulan la Búsqueda de Alimento de las Bacterias (Das et al., 2009) y los inspirados en la Búsqueda del Cuckoo (Yang y Deb, 2009), por citar solo algunos.

Otros tipos de algoritmos se basan en simular el comportamiento conocido del cerebro humano (principalmente el referido a las neuronas y sus conexiones), a estos métodos se les conoce como Redes Neuronales Artificiales (RNAs) (Dayhoff, 1990). Existen varios tipos de RNAs, distinguidas principalmente por la topología que emplean. En general, son métodos complejos y el proceso de aprendizaje suele demandar un coste computacional elevado. Entre sus ventajas principales resaltan la capacidad de adaptación y su tolerancia a fallos.

Nos hemos limitado a mencionar sólo algunos casos de los métodos bioinspirados más conocidos. Actualmente hay publicados alrededor de 40 algoritmos diferentes (Fister Jr. et al., 2013). Otras investigaciones exitosas han propuesto algoritmos *híbridos* que combinan dos o más técnicas, las cuales pueden ser bioinspiradas o muy distintas entre sí. Siguiendo la pista de la naturaleza como fuente de inspiración infinita, en la siguiente sección se explicará otro de los métodos inspirados en la teoría de la evolución



y que ha sido objeto de estudio en esta tesis.

2.2. Coevolución

John Thompson en su libro *Mosaico Geográfico de la Coevolución* (Thompson, 2005) planteó que: *la coevolución es un cambio evolutivo recíproco entre especies interactuantes conducido por selección natural*. Algunos investigadores aseguran que este proceso de adaptación mutua ha moldeado la red de la vida, dando como resultado un mundo donde no sólo existen millones de especies, sino también decenas de millones de interacciones entre ellas, que hacen posible que se conserve la diversidad genética (Thompson, 2010). En la naturaleza un organismo no tiene un valor de *fitness* o aptitud *per se* sino que está determinado por su nicho ambiental y la presencia de otros individuos, ya sean de su misma especie o de otra. Entre los individuos pueden producirse interacciones positivas (p.e. simbiosis y mutualismo) o negativas (p.e. predación y parasitismo). Varios son los ejemplos estudiados por científicos que demuestran el desarrollo coevolutivo que ha tenido y aún tiene la vida en el planeta (Oyama, 1986; Torres, 2009; Marquez et al., 2007; Ueda et al., 2008; Moran et al., 2008; Joussein et al., 2009).

Gran parte del trabajo en algoritmos coevolutivos se ha centrado en dos tipos de interacción: los sistemas de coevolución cooperativa y los sistemas de Coevolución Competitiva (CC). En la primera los individuos interactúan de forma simbiótica. En el modelo cooperativo se evoluciona un objeto complejo que ha sido descompuesto en subcomponentes que coevolucionan, y el *fitness* está determinado por el buen funcionamiento con los otros subcomponentes en la producción de una solución completa. Varios métodos de colaboración han sido diseñados para aplicarse en algoritmos coevolutivos-cooperativos (Wiegand et al., 2001; Adamu y Phelps, 2010; Au y Leung, 2008) y aún se sigue investigando para lograr modelos más eficaces.

Por otra parte, la coevolución competitiva se refiere al proceso en el que las especies

compiten entre sí para ganar *fitness* una a costa de la otra. Aquí la aptitud de cada individuo está determinada por una serie de competiciones entre oponentes que pueden ser de la misma especie o de especies diferentes. El objetivo es encaminar un proceso al estilo de una carrera armamentista, donde las mejoras en unos individuos causen nuevas mejoras en otros y viceversa. El enfoque competitivo ha despertado una amplia actividad investigativa, temas como la selección de los oponentes y la evaluación de los individuos para diferentes dominios de aplicación, han motivado un amplio análisis (Sims, 1994; Rosin y Belew, 1997b; Miconi y Channon, 2006; Togelius et al., 2007a).

2.2.1. Patologías más frecuentes de los modelos coevolutivos.

Infelizmente no siempre un modelo coevolutivo logra buenos resultados, hay situaciones en las que la “carrera armamentista” no interrumpe su curso de mejora incremental, y otras en las que se ve frustrada parcial o completamente. Después de muchas investigaciones en este tema (Nolfi y Floreano, 1998; Ficici, 2004; Watson et al., 2001; Jong y Pollack, 2004; Bucci y Pollack, 2003; de Jong et al., 2007; Ficici y Pollack, 2003; Ficici, 2004; Miconi y Channon, 2006) se han identificado un conjunto de patologías que son las causantes de la mayoría de las afectaciones.

Existen dos factores claves que influyen directamente en el desempeño de los modelos coevolutivos, ellos son: los objetivos subyacentes, y el gradiente selectivo. El primero se refiere a la existencia de un conjunto de capacidades que deben garantizarse con la búsqueda pero puede ocurrir que intentando optimizar una se afecte otra, lo cual atenta contra la calidad del resultado, y por tanto exige que se tenga especial cuidado al usarlo. El gradiente selectivo, tiene que ver con la información obtenida de la evaluación, enmarca la habilidad de distinguir los individuos en base a su interacción con los demás, permitiendo identificar cuáles tienen un mejor desempeño. En este último factor el método de evaluación que se defina es determinante para guiar la búsqueda. A continuación se explican las principales patologías coevolutivas documentadas en la



bibliografía científica.

Ciclos

A menudo las estrategias que se obtienen como fruto del proceso coevolutivo tienen una relación intransitiva, aunque en esto influye también la naturaleza del problema tratado. Esta intransitividad impide asegurar que una estrategia sea mejor que su predecesora porque haya logrado vencerla en la competencia, puede que ésta sólo sea más sofisticada y logre explotar las debilidades de su antecesora pero no constituya un paso superior en la evolución. De aquí que, uno de los problemas que pueden aparecer en un proceso coevolutivo son los llamados “ciclos”, estos se refieren a la recurrencia a estrategias predecesoras que logran vencer a nuevas estrategias surgidas en el proceso. Por ejemplo, se supone que existe una estrategia A que logra vencer a una B y luego aparece una estrategia C que supera a esta B , después se encuentra una D la cual es superada por la A , entonces al ser A la ganadora volvería a repetirse este conjunto de estrategias una y otra vez. Aquí se estancaría la búsqueda y el proceso quedaría frustrado. Esta patología impone que se tenga especial cuidado en la transitividad de las soluciones, se debe garantizar que exista realmente dicha relación transitiva, y contar con un método de evaluación que sea capaz de identificar cuando una estrategia es completamente superior a otra.

Desencaje de la búsqueda

Es causado por la pérdida del gradiente selectivo durante el proceso coevolutivo, se puede dar el caso de que por ejemplo: no se logre distinguir entre las soluciones pues todas parezcan buenas. Este problema puede provocar el estancamiento total de la búsqueda y hasta la deriva genética de la(s) población(es). Situaciones en las que sea difícil de cuantificar una medida objetiva de *fitness* son vulnerables a esta patología. También se corre el riesgo de padecerla cuando se pierde la diversidad entre



los individuos de la población, y sobre todo en el conjunto de oponentes que guían la coevolución.

Efecto de la Reina Roja

Se refiere a la posibilidad de que a pesar de un cambio evolutivo continuo, el *fitness* de los individuos de una especie puede no variar debido a cambios en el ambiente. Si el medio cambia, las especies se ven obligadas a cambiar también y por tanto ese cambio puede que no represente una mejora de su aptitud sino que sólo sea una necesidad de adaptación al entorno para garantizar la supervivencia en él. En un algoritmo coevolutivo sucede que una solución puede cambiar mucho pero no aumentar su tasa de selección a causa de la evolución de los competidores. Los nuevos individuos pueden ser más sofisticados pero no más capaces que los individuos antiguos debido al nuevo contexto. El efecto de la Reina Roja puede dificultar la distinción entre la dinámica de una “carrera armamentista” y la de un proceso cíclico causado por la intransitividad. Fijémonos que para no ser víctimas de esta patología es necesario que el mecanismo de evaluación de la población nos permita identificar si los nuevos individuos están siendo realmente peores que sus ancestros o simplemente se han tenido que enfrentar a condiciones más difíciles de adaptación que han hecho que su valor de *fitness* sea inferior.

Sobreespecialización

Muchas veces la naturaleza del proceso coevolutivo es multiobjetivo, es decir, la solución debe garantizar diferentes capacidades lo cual implica que hayan diferentes objetivos a tener en cuenta en la búsqueda. La mejora de una capacidad a veces implica el empeoramiento de otra. Esto hace posible que se produzcan individuos que se especializan en derrotar a ciertos tipos de oponentes para quienes exhibirán un desempeño óptimo, sin embargo para otros serán ineficientes. Aquí las soluciones serán el



resultado de un proceso de búsqueda-aprendizaje incompleto donde sólo se enfocaron una parte de todo el conjunto de objetivos a cumplir. Esta sería una situación típica en la que se pueda generar un desencaje de la búsqueda y hasta una pérdida extrema de la diversidad en la población.

A modo de resumen y tratando de relacionar estas patologías podemos decir que el desencaje de la búsqueda ocurre cuando hay pérdida del gradiente, y si este se mantiene a escala evolutiva, puede dar lugar al estancamiento o deriva genética de la(s) población(es). La deriva, por su parte, causa el olvido de características (objetivos en la búsqueda) o la sobre-especialización. Y también la oscilación entre diferentes objetivos puede producir ciclos que frustran completamente el proceso de búsqueda. Éstas han sido las patologías tratadas en la bibliografía consultada como las más frecuentes entre los modelos coevolutivos, su estudio nos da la medida de cuántos detalles hay que tener presente a la hora de diseñar un algoritmo de este tipo. En la segunda parte de esta sección veremos algunos de los métodos que han sido diseñados por los investigadores para detectar y evitar estos fallos.

2.2.2. Remedios para combatir las patologías

La creación de métodos que permitan monitorizar el avance de un proceso coevolutivo para identificar cuándo ha habido algún fallo en él, es una práctica en la que muchos investigadores se han enfocado. Esto ha ayudado en la lucha contra las patologías, pues permite definir bajo qué condiciones son más propensas a ocurrir, y a partir de ello diseñar formas de evitarlas. No se puede asegurar que estén descubiertas todas las “curas” para los padecimientos de los modelos, pero sí se han obtenido varios avances y aún se continúa la búsqueda por obtener soluciones más robustas.

Existen dos tendencias fundamentales en los algoritmos de monitorización, una es la que basa su funcionamiento en determinar el desempeño evolutivo de la población desde un punto de vista general, la otra se enfoca en identificar los individuos más so-



fisticados y a partir de ellos marca el desarrollo evolutivo. Gran parte de estos métodos ofrecen información razonablemente completa sobre una carrera evolutiva, sin embargo ese resultado trae como consecuencia que se requieran muchas evaluaciones de los individuos para poder trazar el progreso.

Uno de los primeros intentos por remediar las patologías de los modelos coevolutivos fue propuesto por Rosin y Belew (Rosin y Belew, 1997a) en el año 1997. Se trataba de un método de archivo llamado *Salón de la Fama* (HoF, por sus siglas en inglés: Hall of Fame) que actúa como un mecanismo de memoria a largo plazo en los algoritmos CC para gestionar el conjunto histórico de campeones (o sea, los mejores individuos obtenidos en cada ciclo coevolutivo) durante la evaluación de los individuos. Otros trabajos investigativos, como el publicado en (Ebner et al., 2010), se han enfocado en el análisis de la dinámica de los procesos coevolutivos para identificar sus debilidades y fortalezas. En este sentido también, en (Miconi, 2009) se describe la importancia de los términos *superioridad* y *avance*, para evitar fallos en el proceso de búsqueda coevolutivo. Más recientemente los autores de (Samotheakis et al., 2013) utilizaron diferentes algoritmos para generar jugadores artificiales y presentaron un conjunto de mediciones para identificar los ciclos en una población de individuos, y un algoritmo que minimiza el efecto de los ciclos en un sistema coevolutivo para el juego Othello.

Son muchos los enfoques que han sido diseñados para combatir las patologías, cada uno se especializa en una específica o incluso, puede tratar varias. Están los que luchan contra el olvido de características y se basan comúnmente en distinguir los individuos en las poblaciones para evitar el olvido de rasgos en la evolución. En el tratamiento del desencaje de la búsqueda predomina la tendencia de mantener las soluciones subóptimas en la población con la idea de conservar la diversidad de los individuos. Como ya se ha comentado, existen los conocidos “métodos de archivo” que se centran en preservar durante varias generaciones los individuos que han mostrado un desempeño exitoso, el objetivo es crear un conjunto de campeones que será tomado como punto de referencia

para evaluar las poblaciones que van surgiendo. En lo que sigue se explicarán algunos de estos “remedios”.

Métodos de medición del *fitness*

Con el fin de conservar la diversidad en un proceso de coevolución competitiva, se proponen diferentes formas de medir el *fitness* de los individuos. Una de ellas es el método de “Compartición Competitiva del Fitness” (“Competitive Fitness Sharing”, CFS, por sus siglas en inglés) (Rosin y Belew, 1997a), en el cual la puntuación resultante del enfrentamiento contra un oponente se normaliza dividiendo por la suma de todas las puntuaciones obtenidas contra dicho rival. Está también el “Fitness Basado en Cobertura”(Juille y Pollack, 1996) que asigna la puntuación comparando con los resultados de otros miembros de la población. Estos métodos ofrecen una vía justa de medir la aptitud de un individuo, pues tienen en cuenta el resultado de los demás miembros de la población. Con el CFS se pueden identificar aquellos individuos que a pesar de no haber tenido un destacado desempeño en la competencia, han sido capaces de derrotar a otros que se mantuvieron invictos o casi invictos, y que posiblemente pasarían inadvertidos si se usara un método tradicional de medición. Es recomendable en un modelo coevolutivo que la evaluación de los individuos sea lo suficientemente representativa del individuo en sí y del entorno que lo rodea, la propuesta del *fitness* subjetivo hecha en (Watson et al., 2001) se corresponde con esta idea. Los métodos de medición abordados en este párrafo, no llegan a abarcar el concepto de *fitness* subjetivo, más bien se quedan siendo una medida objetiva de calidad, pero sí constituyen una aproximación a un concepto más amplio de la evaluación dentro de este plano objetivo.

Hall of Fame (HoF)

Propuesto en (Rosin y Belew, 1997a) es uno de los *métodos de archivo* que resultan útiles para mantener la presión de selección durante el proceso de búsqueda, y para

evitar el olvido de características. Propone que el mejor individuo de cada generación sea preservado con el objetivo de usarlo en la evaluación. Los individuos de la población evaluada son probados contra los oponentes actuales y contra una muestra de ese elenco de la fama. De esta manera se evita la sobreespecialización, pues el éxito de los nuevos miembros del salón depende de que estos estén aptos para vencer a los viejos oponentes. Esta técnica también puede ser usada para monitorizar el progreso coevolutivo, debido a que permite hacer un seguimiento de la calidad de los nuevos individuos, comparándolos con los miembros del Salón de la Fama. En dependencia del dominio tratado se puede variar el criterio de selección de los miembros del salón, y así ajustar la presión selectiva y hasta el coste computacional. Por ejemplo, un posible ajuste de este método puede ser el de combinarlo con la filosofía que se usa en el Torneo de Dominancia y hacer que sólo se incorporen al salón los individuos que logren vencer a todos los miembros que ya están en el elenco.

Memoria Nash

Fue definido en (Ficici y Pollack, 2003) está basado en el principio del Equilibrio de Nash. Su principal objetivo es evitar que ocurra la pérdida de características durante el proceso de búsqueda. Para esto crea una colección de rasgos sobresalientes descubiertos por la búsqueda y la representa como una estrategia mixta, la cual debe ir aproximándose monótonamente a ser una estrategia que garantice el Equilibrio de Nash (Nash, 1951). La memoria de estrategias debe manejar apropiadamente los ciclos intransitivos. Este algoritmo está estrechamente relacionado con la teoría de juegos. En investigaciones realizadas en (Ficici, 2004) se ha comparado su eficacia con la del Torneo de Dominancia, los resultados han mostrado que la Memoria Nash logra una mejor aproximación a la solución del problema. Las características de este algoritmo imponen la necesidad de conceptualizar el problema a tratar dentro del contexto de la teoría de juegos lo cual será conveniente, en mayor o menor medida, dependiendo del

dominio sobre el que se esté trabajando.

Torneo Máster

Fue propuesto como un método de monitorización, se deriva del CIAO (*Current Individual vs. Ancestral Opponents*) (Cliff y Miller, 1995). En el Torneo Máster (Floreano y Nolfi, 1997) el objetivo es medir el desempeño del mejor individuo de la generación enfrentándolo diez veces contra los campeones de las anteriores generaciones, esto requiere que las competiciones no sean deterministas y garanticen resultados diferentes. De las competencias se conforma una gráfica que muestra la capacidad de los campeones en cada generación para derrotar a otros de generaciones previas, y a través de esto se deduce si hubo progreso en el proceso coevolutivo. Este método puede generar un alto coste computacional debido a su complejidad combinatoria.

Torneo de Dominancia

Definido en (Stanley y Miikkulainen, 2002) con fines de monitorización, se basa en la filosofía de que para dar seguimiento al progreso de una carrera coevolutiva se deben identificar las estrategias dominantes. Para que un individuo sea catalogado como dominante tiene que superar a todas las anteriores estrategias dominantes (que a su vez han sido capaces de vencer a sus antecesores). El Torneo de Dominancia suele ser computacionalmente menos costoso que el método anteriormente visto, pues la cantidad de estrategias dominantes es menor que el total de campeones de todas las generaciones.

2.3. Conclusiones del capítulo

En este capítulo se han abordado los conceptos principales que definen el marco teórico de esta tesis. Después de este breve acercamiento a la CE no cabe duda de que se trata de un campo verdaderamente fascinante que proviene de una fuente de

inspiración infinita como es la naturaleza. En los últimos años las metaheurísticas y en particular los AEs han tenido un vertiginoso desarrollo. Cada año se publican nuevos resultados que abren otras líneas de investigación y fomentan las que ya existen. Si tuviésemos que definir una desventaja para los AEs podríamos decir que en dependencia del “afinamiento” que se le quiere dar al proceso de búsqueda pueden llegar a demandar un coste computacional elevado para encontrar soluciones óptimas o casi óptimas. Es por eso que en el campo de los videojuegos, se suelen utilizar en etapas de preprocesamiento, de forma que el tiempo de ejecución de un algoritmo no influya en la fluidez que demanda un sistema de tiempo real. Por otra parte, las ventajas de los AEs radican principalmente en su capacidad de adaptación a contextos muy diversos. También, a pesar de haber mencionado antes el tema de los costes computacionales, el rendimiento de estos algoritmos comparados con el de métodos exactos, a la hora de resolver un problema complejo es muy superior. Los AEs son especialmente útiles cuando nos encontramos con problemas difíciles o altamente irresolubles.

La CE debe entenderse como un concepto general adaptable para la solución de problemas, especialmente adecuada para resolver problemas de optimización, en lugar de una colección de algoritmos listos para usar (Bäck et al., 1997). Y la Coevolución, como una rama de esta área, es también un método que destaca por su alto grado de adaptabilidad. En próximos capítulos se presentarán diferentes enfoques que hemos definido para explorar y explotar los modelos CC en el contexto de los videojuegos. Y se verán ejemplos concretos de su idoneidad en escenarios donde se genera un ambiente de competición.



UNIVERSIDAD
DE MÁLAGA

Capítulo 3

Inteligencia Artificial aplicada a videojuegos

Artificial Intelligence is not a Man versus Machine saga;

it is in fact, Man with Machine synergy.

Sudipto Ghosh

El videojuego es un fenómeno social, cultural e industrial, que vive una grandiosa expansión a nivel mundial, con más de mil quinientos millones de jugadores que generan un mercado global que supera los 90.000 millones de dólares y que crece anualmente a un ritmo de más del 6% (DEV, 2017). El mercado mundial de los videojuegos generó en 2015 unas ventas de 91.800 millones de dólares, según datos de *Newzoo* (Warman, 2016). Esta cifra supera en dos veces y medio la dimensión del mercado del cine (que fue de 38.300 millones de dólares en 2015), según la Asociación Cinematográfica de Estados Unidos y en más de seis veces el mercado de la música (15.000 millones de dólares en 2014), según los datos de la Federación Internacional de la Industria Fonográfica.

CAPÍTULO 3. INTELIGENCIA ARTIFICIAL APLICADA A VIDEOJUEGOS

Esta situación tan favorable ha supuesto una motivación para la investigación aplicada a los videojuegos, la cual ha estado adquiriendo notoriedad a lo largo de estos últimos años, abarcando varias áreas como la psicología y la satisfacción del jugador, el marketing y la gamificación, la IA, los gráficos por computador e incluso la educación y la salud (juegos serios). Actualmente están emergiendo nuevos retos y objetivos dentro del área de los videojuegos, especialmente en el campo de la IA aplicada (Lucas et al., 2015; Yannakakis y Togelius, 2015).

En este capítulo se hará un análisis de las tendencias actuales y futuras de la aplicación de la IA en los videojuegos. Mostraremos ejemplos de varios casos de éxitos, así como las líneas de investigación más atractivas que definen la frontera del conocimiento en este campo y prometen ser las que guiarán el futuro del desarrollo de los videojuegos en los próximos años.

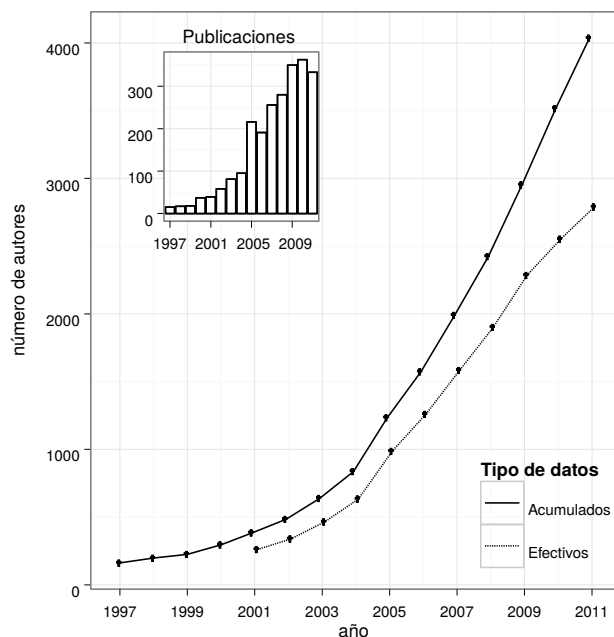


Figura 3.1: Evolución del número de autores y publicaciones en la red de coautoría del área de IA en videojuegos.

Un análisis sobre la red de co-autoría dentro del campo de la Inteligencia Compu-

tacional (IC) en videojuegos (Lara-Cabrera et al., 2014) , concluye que nos encontramos ante un sector activo y vibrante, el cual atrae a nuevos investigadores año tras año y genera nuevas publicaciones con nuevos autores. En dicho análisis se constató un crecimiento estable en el número de autores, el cual se acentúa a mediados de la década 2000-2010 (ver Figura 3.1). También que el número de publicaciones por año de la comunidad ha estado incrementándose desde 2005; y que la comunidad de la Inteligencia Computacional (IC) en videojuegos se encuentra en un estado inicial de desarrollo, formando enlaces y mejorando su cohesión, pero no llega a estar completamente formada; de la misma forma, la industria está empezando a adoptar las técnicas y recomendaciones que el mundo académico les ofrece.

3.1. Problemas de Optimización frecuentes en videojuegos

Tradicionalmente la IA de un videojuego ha sido codificada manualmente mediante sistemas predefinidos de reglas dando lugar a menudo a comportamientos englobados dentro de la llamada *estupidez artificial* (Lidén, 2003) lo que resulta en un conjunto de inconvenientes conocidos, tales como por ejemplo la sensación de irrealidad, la aparición de comportamientos anormales en situaciones no consideradas o el conocimiento de comportamientos predecibles. Actualmente se emplean técnicas avanzadas para solventar estos problemas y se logran *NPCs* con comportamientos racionales que toman decisiones lógicas correctas de la misma forma que lo hace un jugador humano. La principal ventaja que ofrecen estas técnicas es que realizan de forma automática el proceso de búsqueda y optimización de esas estrategias de juegos “inteligentes”.

Los algoritmos bioinspirados son la base de muchos de esos métodos avanzados, pues resultan un enfoque muy apropiado en este sentido, ya que son capaces de producir soluciones de gran complejidad como resultado emergente del proceso de optimización,





Figura 3.2: Captura de pantalla del juego NERO

y su capacidad adaptativa les permite incorporar información proporcionada por el usuario y explotarla de manera no anticipada por el mismo. De ahí que existan muchas propuestas exitosas que siguen esta línea. Por ejemplo, la Coevolución ha sido ampliamente utilizada en la programación de IA para videojuegos.

El aprendizaje computacional también se utiliza a la hora de modelar el comportamiento de los jugadores artificiales. A continuación se destacan algunos ejemplos que pertenecen al estado del arte. Los autores de (Preuss et al., 2010) han empleado mapas auto-organizados para mejorar el manejo de los batallones de unidades en un juego de estrategia en tiempo real; mediante el análisis de los datos obtenidos por los sensores, los autores de (Quadffieg et al., 2010) han diseñado un algoritmo para que un piloto artificial se aprenda el circuito en el que posteriormente tiene que pilotar de forma

autónoma. También se pueden encontrar ejemplos en los que se combinan redes neuronales con los algoritmos bioinspirados descritos anteriormente, como en el caso de (Stanley et al., 2005), artículo en el cual se presenta un sistema para la evolución en tiempo real de redes neuronales con el objetivo de mejorar los comportamientos de los agentes artificiales del juego NERO (ver Figura 3.2). Otro posible uso de las técnicas de aprendizaje computacional, en concreto la minería de datos, se puede encontrar en (Weber y Mateas, 2009), donde se construye un predictor de posibles estrategias tras someter a una base de datos de partidas de *Starcraft* a un proceso de análisis de datos. En (Fernández-Ares et al., 2012), los autores extraen características de los mapas de un juego de estrategia en tiempo real las cuales influyen en el comportamiento de un jugador artificial, que se aprovecha de esta caracterización para obtener una ventaja con respecto a su oponente.

3.2. Tendencias actuales y retos

Esta sección tiene como objetivo enfocar la atención en algunas de las principales tendencias que parecen guiar el futuro de los videojuegos, y de los retos que éstas le imponen a la comunidad científica, centrándonos en la perspectiva de la aplicación de técnicas de IA. Queremos matizar que el universo de aplicaciones de las técnicas de optimización sobre el desarrollo y diseño de videojuegos es enormemente amplio y no pretendemos realizar un recorrido exhaustivo sobre el mismo en este capítulo; de hecho recomendamos al lector interesado en obtener más información, la lectura de otros artículos que ya han sido publicados en la literatura y que sirven como base para conocer el estado del arte de esta área, tales como (Togelius et al., 2011; Yannakakis y Togelius, 2015). Nosotros nos centramos en determinadas áreas de investigación que creemos que van a influenciar significativamente la creación de juegos comerciales en la próxima década; en concreto nos referimos a la Generación Procedural de Contenido; a la Computación Afectiva, la cual afecta directamente a la satisfacción del jugador;

y a la generación de comportamientos o estrategias de toma de decisiones asociados a los jugadores virtuales (es decir, artificiales, no controlados por el jugador humano, que simulan el comportamiento de un posible oponente humano en el juego).

3.2.1. Computación afectiva y satisfacción del jugador

Fue Rosalind Picard quien, en 1995, introdujo el término Computación Afectiva y lo define como el cómputo que relaciona, surge o influye en las emociones (Picard, 1997). En el contexto de los videojuegos aún se investiga en cómo extrapolar el inmenso campo de las emociones hasta el escenario de un juego, las buenas razones de por qué hacerlo están claras (Freeman, 2004), pero los resultados obtenidos hasta el momento son modestos comparado con todo lo que se pretende lograr.

Una de las primeras formas usadas para incorporar emociones en los juegos fue a través de la narrativa, mediante la generación de situaciones dirigidas a “atrapar” (o robarle por completo la atención) al jugador ya sea por los personajes, la presentación de todo tipo de conflictos e historias fantásticas o de la vida real. En esta caracterización destacan entre las sagas preferidas por los usuarios *Final Fantasy* y *Resident Evil*. También están los videojuegos como *Fable* o *Beyond: Two Souls* que se distinguen por tener un alto grado de realismo en las simulaciones e incorporan emociones en algunas de las actuaciones del protagonista, véase en la Figura 3.3 una captura de pantalla de estos juegos donde se observa al personaje mostrando emociones. Este tipo de *afectividad centrada en el personaje principal* lleva detrás un meritorio trabajo artístico que permite la simulación realista de las emociones, por ejemplo en el juego *Beyond: Two Souls* se utilizan técnicas de captura de movimiento para animar al personaje virtual y son actores reales (estrellas de cine como Ellen Page y Willem Dafoe) los que interpretan a los protagonistas de la historia. Este juego en particular tiene un peso narrativo muy grande pues en todo momento el jugador es guiado en la partida por una historia o guion que ha sido predefinido, con lo cual se acotan

CAPÍTULO 3. INTELIGENCIA ARTIFICIAL APLICADA A VIDEOJUEGOS

considerablemente las posibles acciones que puede elegir el usuario mientras juega, y esto hace que sea más fácil para el software controlar el flujo emocional del protagonista ya que los estados emocionales por los que transita son en su mayoría predecibles. En general, este enfoque para la implementación de la afectividad tiene como objetivo principal incrementar la inmersión del usuario en el juego y para eso establece una dinámica emocional entre el jugador humano y el personaje principal; tratando de lograr en todo momento que el jugador se sienta identificado con lo que le transmite su personaje.

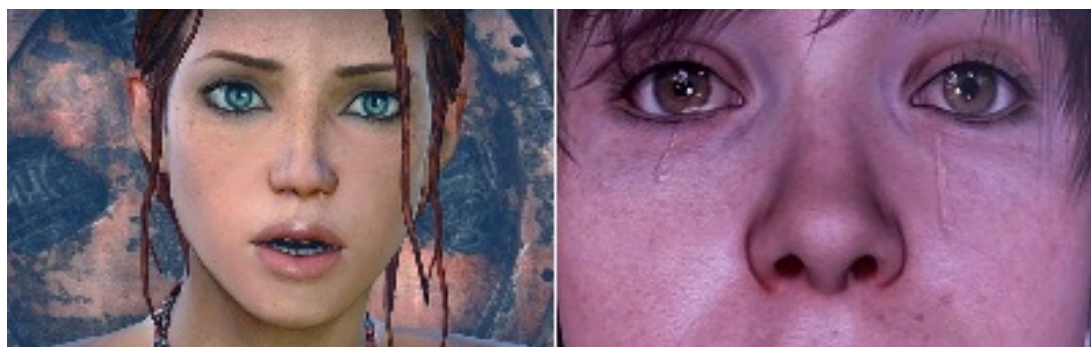


Figura 3.3: Personajes de los juegos *Fable* (izquierda) y *Beyond: Two Souls* (derecha) mostrando emociones

Esa forma de transmitir afectividad al jugador mediante su personaje virtual es una de las tendencias usadas por los juegos comerciales actuales, suele estar relacionada con los géneros de acción-aventura y los juegos basados en roles donde es típico que el jugador tenga total visibilidad del personaje principal casi todo el tiempo. Además de este, existe otro enfoque cuyo objetivo es que los *NPCs* que participan en la partida se comporten como individuos emocionales y, por tanto, sus emociones influyan cuando tome decisiones en la partida. Para lograr esto se hace necesario la existencia de un alto grado de percepción para estos *NPCs* pues ellos tendrían que reaccionar ante todo suceso que ocurra a su alrededor que pudiera afectar su estado emocional, por ejemplo, si se provoca un ruido cerca del *NPC* hay que evaluar si ese sonido podría afectar

su estado de ánimo (asustarle, alarmarle, alegrarle, etc.). Precisamente este aumento en el nivel de percepción del entorno demanda un proceso de toma de decisiones más complejo, que implica encontrar una estrategia de juego que dote a esos *NPCs* de inteligencia para que tomen las decisiones lógicas correctas y a su vez tenga en cuenta sus emociones. Esta pretensión de simular de forma realista el comportamiento humano en los *NPCs* es desde hace mucho un campo de investigación abierto, dentro de él está la rama que se centra en imitar la inteligencia humana (Polceanu et al., 2016). Y por otra parte, está la línea que se dedica al estudio de cómo incorporar las emociones a la simulación, en la cual también se han desarrollado propuestas muy interesantes que ofrecen técnicas y modelos para implementar comportamientos emocionales en los agentes virtuales, ejemplos de algunos de ellos son (Peña et al., 2011b; Peña et al., 2011a; Johansson y Dell’Acqua, 2012). Sin embargo, los mejores resultados están aún por verse y se sigue echando en falta el factor emocional en la IA que exhiben la mayoría de los videojuegos comerciales.

Los dos enfoques analizados hasta este momento manejan la afectividad desde los *NPCs* (ya sean *bots* o jugadores virtuales) y desde el personaje principal que maneja el usuario humano. Ambos enfoques se centran principalmente en lo que ocurre dentro del videojuego y en cómo deben reaccionar los personajes para ser emotivos ante esos hechos. Pero no tienen en cuenta otras reacciones emocionales que expresa el usuario humano mientras esta jugando la partida, las cuales sería necesario tenerlas en cuenta para lograr una verdadera interacción afectiva bidireccional entre el videojuego y el jugador humano. De aquí que exista un tercer enfoque, el de la *Auto-adaptabilidad* en videojuegos afectivos, que está estrechamente relacionado con la temática del *Modelado, Evaluación e Incremento de la satisfacción del jugador* (Yannakakis, 2008; Nogueira et al., 2012a).

La *Auto adaptabilidad* se refiere a la capacidad que puede tener un juego de auto-adaptar la partida a cada jugador, teniendo en cuenta sus preferencias y características,

CAPÍTULO 3. INTELIGENCIA ARTIFICIAL APLICADA A VIDEOJUEGOS

con el objetivo de hacer que esta resulte una experiencia única para cada usuario e influya directamente en el aumento de su satisfacción. En el contexto de la afectividad, se trataría de auto-ajustar el juego según las emociones que va expresando el usuario humano durante la partida, haciendo que los lazos afectivos entre humano y personajes virtuales se construyan y expresen en tiempo real. Este sería el enfoque más completo, que incluye también a los dos anteriores, y que actualmente es un anhelo para muchos investigadores. Todavía no existe un videojuego que permita experimentar un nivel de interacción emocional tan alto.

Los resultados más interesantes relacionados con este tema son los que se han obtenido en el campo del *Modelado, Evaluación e Incremento de la satisfacción del jugador*, los cuales pueden considerarse parte de la computación afectiva en videojuegos pues el concepto de satisfacción del jugador está fuertemente ligado a las emociones humanas. Los trabajos más interesantes en esta rama se centran en su mayoría en crear un modelo formal que represente el comportamiento del jugador y que permita hacer una evaluación del nivel de satisfacción del mismo, todo esto basado en investigaciones psicológicas sobre la satisfacción (Malone, 1980; Csikszentmihalyi, 1991; Sweetser y Wyeth, 2005); luego viene la parte de usar dicho modelo para determinar qué grado de satisfacción experimenta el jugador y proceder entonces a reajustar el juego para mantener o elevar ese grado. Algunas propuestas que han tenido éxito se puede ver en (Yannakakis y Hallam, 2007; Halim et al., 2010; Yannakakis y Hallam, 2009; Yannakakis y Hallam, 2006).

Cada enfoque de los mencionados aquí es un campo abierto de investigación, que tiene dentro varias líneas que también demandan nuevas soluciones, y la optimización es una de ellas. La mayoría de las propuestas exitosas que existen, algunas de las cuales han sido citadas aquí, llevan detrás un proceso de búsqueda y optimización basado en metaheurísticas, para poder explorar el amplio espacio de búsqueda que se genera de dos contextos intrínsecamente complejos: los videojuegos y las emociones.



3.2.2. La generación procedural de contenidos

La generación de contenido por procedimientos (PCG, por sus siglas del inglés: Procedural Content Generation), se refiere a la creación algorítmica de contenido para videojuegos, ya sea con intervención humana o sin ella, tales como mapas, niveles, texturas, personajes, reglas y misiones, pero excluyendo el comportamiento de los personajes no controlados por el ser humano y el motor de juego en sí mismo (se usarán los términos “generación automática de contenidos” y “generación de contenidos por procedimientos” de forma indiferente para referirnos a PCG en este capítulo). Puede tratarse desde contenido totalmente necesario para avanzar en el desarrollo del videojuego hasta aquel que es puramente decorativo. En todo caso, los algoritmos utilizados deben garantizar la generación de contenido que cumpla ciertos requisitos de calidad, adaptándose al tipo de contenido que se pretendiese crear manualmente.

El hecho de producir contenido de videojuegos mediante PCG puede permitirnos reducir sustancialmente el consumo de memoria del juego, algo que hoy en día puede tener un carácter secundario pero que en el pasado motivó el empleo de estas técnicas. Otro motivo importante para el uso de estos algoritmos es reducir el elevado coste que tiene en muchos casos la generación de cierto contenido de los juegos de forma manual e incluso servir como fuente de creatividad a los diseñadores, proponiendo diseños que posiblemente no se les hubiesen ocurrido en primera instancia. Además, si como se ha dicho antes se asegura que el contenido generado por PCG cumple con ciertos criterios, como ajustarse a la habilidad de un jugador, el nuevo contenido puede suponerle un reto constante. Si esta generación de contenido adaptado resulta siempre diversa y se genera en tiempo real, es decir, al mismo tiempo que el juego se ejecuta, pueden conseguirse auténticos juegos infinitos, presentando constantemente a cada tipo de jugador nuevos y distintos retos que ha de superar. Estos beneficios son bien conocidos por la industria, habiéndose usado este tipo de técnicas en el desarrollo de videojuegos comerciales de éxito como la saga *Borderlands*, *Skyrim*, *Minecraft* o *Terraria*.

CAPÍTULO 3. INTELIGENCIA ARTIFICIAL APLICADA A VIDEOJUEGOS

A la hora de generar contenido automático para videojuegos pueden hacerse muchas distinciones globales en lo que respecta a los procedimientos a seguir. Atendiendo al momento en el que se produce la creación de los contenidos, la generación puede ser tanto online, durante la ejecución del juego (con las ventajas que esto puede tener), como offline, durante la fase de desarrollo del mismo.

Con respecto a la finalidad del contenido generado, éste puede ser considerado como necesario para poder progresar en el juego, en cuyo caso se ha de asegurar que es correcto (en el sentido de no proponer objetivos imposibles al jugador), o bien opcional, como pueden ser los elementos decorativos y que por lo general pueden ser mucho menos restrictivos.

También cabe preguntarse por la naturaleza del algoritmo de generación, es decir, si nos encontramos ante un algoritmo puramente estocástico, mediante el cual se crea el contenido a partir de semillas aleatorias, o, por el contrario, un algoritmo determinista, donde el contenido se genera mediante un vector de parámetros. La tercera posibilidad radica en la hibridización de ambas perspectivas, diseñando un algoritmo con una componente estocástica y otra determinista, funcionando conjuntamente.

Si nos fijamos en los objetivos que se pretenden cumplir, el proceso de creación puede hacerse de forma constructiva, asegurando la validez del contenido durante todo el proceso. La otra opción consiste en seguir un esquema de generación y prueba, donde se genera una gran cantidad de contenido el cual se pasa por una fase de validación y posterior descarte de todo aquel que no cumpla con las restricciones. Este último esquema es el más empleado actualmente por la comunidad, y se basa en la búsqueda de los contenidos en el espacio de posibles soluciones. La validación se realiza asignando uno o varios valores al contenido de forma que quede cuantificada el nivel de calidad, según nuestros objetivos, del contenido creado.

Estas técnicas se utilizan comúnmente para generar mapas y niveles, como demuestra el elevado número de artículos dedicados a ello (Hendrikx et al., 2013). Por ejemplo,



Mahlmann et al. (Mahlmann et al., 2012) presentan un generador de mapas para una versión simplificada del juego de estrategia Dune 2, el cual se basa en la transformación de matrices de baja resolución en mapas de mayor resolución utilizando para ello autómatas celulares. Frade et al. (Frade et al., 2008; Frade et al., 2009; Frade et al., 2010) introducen el uso de programación genética para evolucionar mapas de videojuegos (denominado por los autores como programación de terrenos), usando tanto la evaluación subjetiva de los humanos como medidas de calidad extraídas de forma automática como la accesibilidad de los mapas o la longitud de aristas. La evaluación humana del contenido generado por procedimientos es tratada por Liapis et al. en (Liapis et al., 2013a). Por otra parte, Togelius et al. (Togelius et al., 2007b) han diseñado un sistema capaz de generar circuitos de carreras a partir de un vector de parámetros usando una transformación genotipo-fenotipo determinista. Un artículo de Ashlock y McGuinness (Ashlock y McGuinness, 2013) introduce un autómata que busca en el espacio de soluciones posibles de mapas de altura usando un algoritmo evolutivo. En este caso, el diseñador es capaz de definir puntos del mapa que deben ser mutuamente accesibles para influir en la morfología de los mapas generados. Otro generador de mapas basado en un algoritmo evolutivo se define en (Diaz-Furlong y Solis-González Cosío, 2013), pero en este caso la función de aptitud depende de la diferencia entre distintas curvas de dificultad definidas por el diseñador para cada nivel.

Como se puede comprobar, existe un gran número de artículos dedicados a la generación de mapas y niveles, los cuales tienen algo en común: sus algoritmos buscan cumplir varias restricciones relacionadas con la mecánica del juego, tales como la accesibilidad de ciertas zonas de los niveles, el ajuste de la dificultad o el equilibrio entre el nivel de los jugadores. De esta forma, se incrementa la satisfacción del jugador que, a fin de cuentas, es el objetivo principal de los videojuegos.

Además de mapas y niveles, la generación por procedimientos se utiliza para otro tipo de contenido. Por ejemplo, Font et al. (Font et al., 2013) describen los primeros

pasos hacia un sistema capaz de generar juegos de cartas, siendo en este caso las reglas del juego lo que se genera de forma automática. Collins (Collins, 2009) hace una introducción a la generación procedimental de música para videojuegos, estudiando las diferentes estrategias de composición y control que se han usado anteriormente. Otros autores (Onuczko et al., 2006) han creado un prototipo de una herramienta que crea patrones de diseño para misiones del juego de rol *Neverwinter Nights*. Por su parte, García-Ortega et al., (García Ortega et al., 2014) utilizan algoritmos genéticos para la generación automática de historias interesantes en videojuegos.

3.3. Marcos de trabajo

Esta sección está dedicada a presentar algunas de las herramientas o marcos de trabajo que están a disposición de la comunidad científica para la realización de pruebas y validación de los resultados obtenidos durante una investigación. Hoy en día existen multitud de herramientas de libre disposición, por lo que a continuación se presenta un compendio de las más usadas junto a sus características principales, para servir como lista de referencia a investigadores del campo de la IA y los videojuegos.

La herramienta *ORTS* (Open Real-Time Strategy) (Buro, 2002) es un juego de estrategia en tiempo real diseñado específicamente para su uso en la investigación y liberado bajo la licencia GPL (GNU public license). Su protocolo abierto de intercambio de mensajes y el programa cliente permiten a los investigadores analizar el rendimiento de sus algoritmos jugando partidas en un entorno seguro donde la simulación se lleva a cabo en el lado del servidor. Este marco de trabajo proporciona la funcionalidad básica de un juego de este género, la cual puede ser ampliada con facilidad. Otro juego de estrategia en tiempo real muy popular en este campo de investigación es el *Starcraft*¹ el cual a pesar de no ser software libre, cuenta con una librería (*BWAPI*)²

¹<http://us.blizzard.com/en-us/games/sc/>.

²<http://bwapi.github.io>





Figura 3.4: Open Real-Time Strategy

que facilita la conexión del motor del juego con las posibles estrategias de IA. Por su parte, RoboCode³ es una plataforma cuyo objetivo es desarrollar (usando Java o .NET) un robot de combate para luchar contra otros robots similares en tiempo real

*Planet Wars*⁴ y *ANTS*⁵ son dos juegos desarrollados en el marco de la competición de IA organizada por Google en sus ediciones de 2010 y 2011, respectivamente. El primero es un juego (que será abordado con detalles más adelante) de conquista espacial para varios jugadores donde el objetivo es conquistar todos los planetas del mapa, mientras que el segundo es un juego también para varios jugadores donde cada uno de ellos representa a un conjunto de hormigas cuyo objetivo es recoger comida y conquistar los hormigueros del resto de jugadores, en unos mapas que incluyen zonas

³<http://robocode.sourceforge.net/>

⁴<http://planetwars.aichallenge.org/>

⁵<http://ants.aichallenge.org/>

no transitables.

En *Vindinium*⁶ se ofrece la posibilidad de tomar el control de un héroe que luchará contra otros jugadores por el control de unas minas de oro durante un número determinado de turnos. El jugador con mayor cantidad de oro al final de la partida se convierte en el ganador. Lo más destacable de este marco de trabajo para estrategias de IA, además de su aspecto visual, es el elevado número de lenguajes de programación que soporta, pudiendo programar las estrategias en más de 25 lenguajes distintos.

Por su parte, *SpelunkBots*⁷ es un conjunto de herramientas en C++ construídas sobre el código fuente original del juego de plataformas *Spelunky*, que permite a los investigadores desarrollar estrategias de IA para determinar el comportamiento del personaje principal a lo largo de las diferentes pantallas del juego. La herramienta ha sido desarrollada por Daniel Scales (Scales y Thompson, 2014).

3.3.1. Competiciones

A lo largo de los últimos años han ido apareciendo diferentes competiciones donde los investigadores tienen la oportunidad de comparar sus estrategias y algoritmos en escenarios y juegos específicos. A continuación se detallan algunas de las más importantes junto con una breve descripción:

- **BotPrize**⁸: Competición consistente en desarrollar un jugador artificial para *Unreal Tournament* que engañe al resto de jugadores humanos haciéndoles creer que él también es humano.
- **Starcraft AI Competition**⁹: Competición anual de jugadores artificiales para *Starcraft* cuyo objetivo es ser el mejor jugador y derrotar al resto de oponentes.

⁶<http://vindinium.org/>

⁷<http://t2thompson.com/projects/spelunkbots/>

⁸<http://botprize.org/>

⁹<http://webdocs.cs.ualberta.ca/~cdavid/starcraftaicomp/>



- **Simulated Car Racing Competition**¹⁰: En esta ocasión el objetivo es diseñar un piloto artificial de carreras que competirá en un conjunto de grandes premios contra otros pilotos artificiales.

Existe un problema con estas competiciones: los retos que se presentan son muy específicos y estrechamente ligados al juego sobre el cual se monta la competición. De esta forma, las estrategias de IA ganadoras se sobre-especializan en explotar las características del juego en sí, pero arrojando un bajo rendimiento cuando se usa ésta estrategia en otro juego. Por tanto, otro posible reto sería diseñar competiciones de estrategias de IA genéricas, que sean capaz de dar buenos resultados no solo en un único juego o entorno, sino en varios de ellos, algo que ya se está empezando a hacer en la competición *GVG-AI* que indicamos a continuación:

- **GVG-AI**¹¹: La *General Video Game AI Competition* es una competición donde las estrategias de IA, o controladores, deben ser capaz de jugar a varios tipos de juegos y escenarios, intentando ser lo más genérico posible.

3.4. Trabajos relacionados con la aplicación de la Coevolución en videojuegos

Esta sección se dedica a discutir sobre la aplicación de la Coevolución Competitiva como método de búsqueda y optimización en videojuegos. Diversos juegos han servido de escenario como experimentación para probar nuevos modelos coevolutivos; algunos son simples como los juegos de mesa y otros, como los RTS, generan un amplio espacio de búsqueda e involucran dominios intrínsecamente complejos.

Como se ha mencionado en el Capítulo 2, un modelo coevolutivo se basa en la interacción entre diferentes individuos que pueden pertenecer (o no) a la misma especie.

¹⁰<http://cig.dei.polimi.it/>

¹¹<http://www.gvgai.net/>

CAPÍTULO 3. INTELIGENCIA ARTIFICIAL APLICADA A VIDEOJUEGOS

Al estudiar los sistemas coevolutivos publicados en la literatura científica se observa una tendencia hacia el uso de modelos que coevolucionan poblaciones pertenecientes a una misma especie. Esto significa que todos los individuos tienen la misma estructura genética o codificación. En dicha tendencia se pueden apreciar dos enfoques. El primero utiliza una única población y durante el proceso de evaluación sus individuos se enfrentan siguiendo un mecanismo de selección, de esta forma las relaciones reproductivas emergen como un proceso natural. El segundo caso emplea poblaciones diferentes. Es muy usual que las poblaciones se traten como linajes y se distingan entre ellas por pequeños detalles como pueden ser variaciones del ciclo coevolutivo (por ejemplo, el proceso de generación de individuos, la función de aptitud, los operadores genéticos, etc.). Otra variante de este segundo enfoque consiste en generar varias poblaciones con las mismas características y condiciones evolutivas. Es interesante ver cómo en todos los casos mencionados, aunque se trabaje con una especie única, es posible producir Coevolución “interespecífica” (según las clasificaciones definidas en (Dawkins y Krebs, 1979)) siempre que se produzca la competencia en poblaciones que no tengan relaciones reproductivas o parentales.

Como muestra de los enfoques antes explicados está por ejemplo el trabajo sobre funciones de aptitud competitivas en el juego *Tic Tac Toe* publicado en (Angeline y Pollack, 1993). También en (Reynolds, 1994) se propone un modelo coevolutivo para generar estrategias de IA en un juego de evasión-persecución. En (Sims, 1994) se diseñó un sistema coevolutivo que permitía la evolución tanto de la morfología como de los comportamientos de criaturas artificiales a través de la competencia en un ambiente depredador-presa. Otra propuesta se hizo en (Ashlock et al., 2012), esta vez se trataba de un método coevolutivo básico que permitía encontrar estrategias de juego para juegos de suma-cero y de suma-no-cero.

Escenarios más complejos que emanan de los juegos de estrategia también han sido considerados para experimentar con la Coevolución. Por ejemplo, en (Smith et al., 2010)



se coevolucionaron a los jugadores virtuales de un juego tipo conquista-la-bandera con el objetivo final de entrenar a los jugadores humanos con los “enemigos artificiales” resultantes. Otra apuesta interesante se hizo en (Miles y Louis, 2006), presentando un sistema de toma de decisiones espaciales en el que los jugadores se enfrentaban primero contra los oponentes estáticos codificados manualmente y más tarde contra otra población de jugadores que coevoluciona.

El juego *Tempo* fue objeto de experimentación en (Avery y Michalewicz, 2007) donde se propuso un sistema coevolutivo basado en dos poblaciones que permitió la interacción con un jugador humano así como la adaptación del proceso de auto aprendizaje a las acciones humanas, todos los individuos (y el modelo del jugador también) compartían el mismo esquema de codificación. La misma autora propuso más recientemente en (Avery y Louis, 2010) el uso de la Coevolución para crear un controlador táctico para pequeños grupos de entidades, la representación de las tácticas adaptativas se basaban en Mapas de Influencia, lograron generar una entidad autónoma que juega en coordinación con el resto del equipo para alcanzar los objetivos comunes. El juego *Tempo* también se ha utilizado con el objetivo de mejorar la creación de agentes inteligentes en (Johnson et al., 2004) y (Avery, 2008). Otra perspectiva interesante fue presentada en (Dziuk y Miikkulainen, 2011) donde se han explorado varios métodos para de configurar automáticamente un proceso coevolutivo, la idea se centra en ir modificando la función de aptitud así como el entorno durante la evolución.

Más estrechamente relacionados con nuestro trabajo están los artículos que abordan el uso de métodos de archivo como una alternativa para evitar el efecto de las patologías, y presentan propuestas interesantes para optimizar su desempeño. Por ejemplo, (Nerome et al., 1998) propone un modelo CC que introduce el concepto de “paquete”, definiéndolo como un conjunto de buenos individuos; y el mejor paquete sería aquel que contenga el menor número de estrategias que aportan, al mismo tiempo, el mayor número de victorias. Otras investigaciones hacen propuestas concretas de nuevos

métodos de archivos como el *Layered Pareto Coevolution Archive* (de Jong, 2004) y el *Coordinate System Archive* (Jaskowski y Krawiec, 2010). También hay otras propuestas más simples que se basan en el uso de una memoria de campeones básica como la propuesta de (Lichocki, 2008) que emplea un enfoque adaptado del Hall-of-Fame (HoF) y lo compara con el *Single Elimination Tournament* (Angeline y Pollack, 1993); o la de (Livingstone, 2005) en la que el autor presenta un sistema que define diferentes niveles de una IA jerárquica que coevoluciona en un entorno de juego RTS simple y se aplican métodos de archivo para evitar algunas de las patologías de la Coevolución utilizando oponentes fijos y no evolutivos durante el proceso de evaluación. Otro ejemplo más reciente se definió en (Samothrakis et al., 2013) donde se presentaron un conjunto de medidas para identificar los ciclos en una población y se propuso un algoritmo que disminuye los efectos de esta patología en un sistema coevolutivo para el juego *Othello*, en los experimentos se utilizaron diferentes algoritmos para generar *IAs*, incluyendo uno que utiliza un método de archivo simple con tamaño fijo, que obtuvo buenos resultados.

La mayoría de los enfoques aquí mencionados utilizan un modelo coevolutivo basado en una sola especie y es, hasta donde sabemos, la variante CC más utilizada en la literatura. Sin embargo, hay modelos computacionales coevolutivos basados en la interacción multiespecies, y producen una Coevolución interespecífica más cercana a la que ocurre en la naturaleza. Esta implementación de la Coevolución es menos común en los juegos, aunque se pueden encontrar un par de propuestas. Una de ellas es la presentada en (Cardona et al., 2013) para el concurso de *PacMan versus Ghost* donde los autores coevolucionaron dos poblaciones diferentes, una para las estrategias de *PacMan* y la otra para el equipo de *Ghost*. En este enfoque las poblaciones eran diferentes especies, aunque ambas compartían la misma naturaleza (es decir, ambas poblaciones evolucionaban jugadores virtuales). Otro enfoque interesante se describió en (Togelius et al., 2007a), en él se comparó el rendimiento entre un sistema coevolutivo de nueve poblaciones y otro de una sola población, el objeto de optimización en ambos casos



eran controladores para un juego de carreras de coches, cada población representaba un controlador, de nuevo se trata de un enfoque donde las poblaciones tienen la misma naturaleza (es decir, todas representan “controladores”).

En el Capítulo 5 presentaremos el algoritmo MuCCCo (Nogueira et al., 2016), se trata de un algoritmo CC que emplea poblaciones de diferente naturaleza. Este enfoque es precisamente uno de los aportes más significativos de esta investigación. Lo más complejo en el modelo diseñado es establecer la relación de competición entre las poblaciones que son tan distintas. En trabajos previos (Nogueira et al., 2012b; Nogueira et al., 2013; Nogueira et al., 2014a) hemos publicado otros resultados de nuestra experimentación con algoritmos coevolutivos que tenían como objetivo la búsqueda y optimización de estrategias de IA para juegos RTS.

Tenga en cuenta que esta sección no menciona otras formas de Coevolución que también se han aplicado en los videojuegos, debido a que no están directamente relacionadas con nuestra investigación, por ejemplo, (Cook et al., 2012) describe un modelo coevolutivo basado en la cooperación y no en la competición que es lo que abordamos en esta tesis.

3.5. Videojuegos que han sido objeto de experimentación en esta tesis

Los dos próximos capítulos de esta memoria estarán dedicados a explicar los algoritmos que han sido diseñados en esta tesis para dar cumplimiento a nuestros objetivos de investigación. Todas las propuestas definidas fueron ampliamente probadas y analizadas para evaluar seriamente su desempeño. Para hacer dichas pruebas se emplearon dos videojuegos de tipo RTS, el *Robot Wars* y el *Planet Wars*, ambos serán explicados en esta sección. Y se ofrecerán detalles de cómo fueron integrados en los métodos coevolutivos. Aclaremos que a partir de esta Sección, usaremos los términos *bot*, *NPC*

y *jugador virtual* indistintamente para referirnos a las IAs de un videojuego.

3.5.1. El juego *Robot Wars*

Es un videojuego RTS que simula el enfrentamiento de dos ejércitos militares sobre escenarios de dimensiones limitadas y con múltiples obstáculos. A diferencia de los videojuegos estándares aquí los jugadores son todos *bots*, el humano no interviene en las partidas. Es más bien un escenario idóneo para probar la eficacia de las estrategias de IA diseñadas para controlar el comportamiento de los jugadores virtuales. Para la visualización tridimensional de los entornos se emplea el motor gráfico *OGRE*, que permite alcanzar la fluidez que exige una simulación de tiempo real con su correspondiente mecanismo de acción-reacción. Cada ejército está formado por un conjunto de unidades con características específicas. El motor de búsqueda evolutivo es capaz de encontrar, bajo las reglas del juego, estrategias ganadoras que proporcionen la victoria de uno de los ejércitos sobre el contrario.

Composición de las tropas

Cada ejército se compone de un número diferentes unidades (incluyendo un general). En la configuración inicial, los dos ejércitos tienen el mismo número de unidades y el ejército que elimina primero al general rival será considerado el ganador. Las unidades pueden ser de diferentes tipos:

- **Infantería:** La unidad básica y normalmente más numerosa, apta para defender al General aliado o atacar al General enemigo. Pueden atacar a cualquier unidad de tierra, aunque su capacidad para hacer daño a los blindados es bastante limitada. No pueden atacar a las unidades aéreas.
- **Blindados:** Aptas para destruir a la infantería enemiga o plantar cara a los blindados enemigos. Pueden atacar a cualquier unidad terrestre. No pueden atacar a las unidades aéreas.



- **Aéreos:** Tienen gran potencia de fuego y son inmunes a las unidades terrestres. Ideales para hacer rápidos desplazamientos para atacar o defender posiciones según las necesidades de la batalla.
- **General:** El General es un tipo de unidad especial que ambos bandos poseen y representa el centro de mando. El General puede ser atacado y destruido como cualquier otra unidad, con la salvedad de que si es destruido el bando pierde la partida automáticamente. No tiene apenas potencia de ataque, por lo que no es adecuado para atacar a otras unidades, pero si puede defenderse devolviendo la respuesta a un ataque si se encuentra en peligro.

Composición del escenario de batalla

Todos los escenarios de batalla (ver Figuras 3.5 y 3.6) están cercados por un conjunto de rocas que marcan sus límites. En el interior se pueden encontrar zonas representadas por llanuras, que son las únicas accesibles por las unidades. También existen áreas inaccesibles en forma de árboles, agua, rocas, etc. que hacen las veces de obstáculos que las unidades deben sortear para moverse por el escenario. Por tanto, toda área del escenario es clasificada por zona accesible o inaccesible, sin posibilidad de ninguna otra combinación.

Reglas de juego

Cada batalla se desarrolla siempre con el enfrentamiento de dos ejércitos sobre un mapa previamente seleccionado. En cada turno, un bando tiene derecho a realizar una acción sobre cada una de las unidades que tenga operativa. Cada unidad dispone de una serie de parámetros que especifican sus cualidades en la batalla:

- **Energía:** Indica si la unidad puede ejecutar una acción o si por el contrario deberá permanecer descansando hasta que su energía se recupere (para recuperarse deberá ejecutar la acción NADA cuando le llegue el turno).

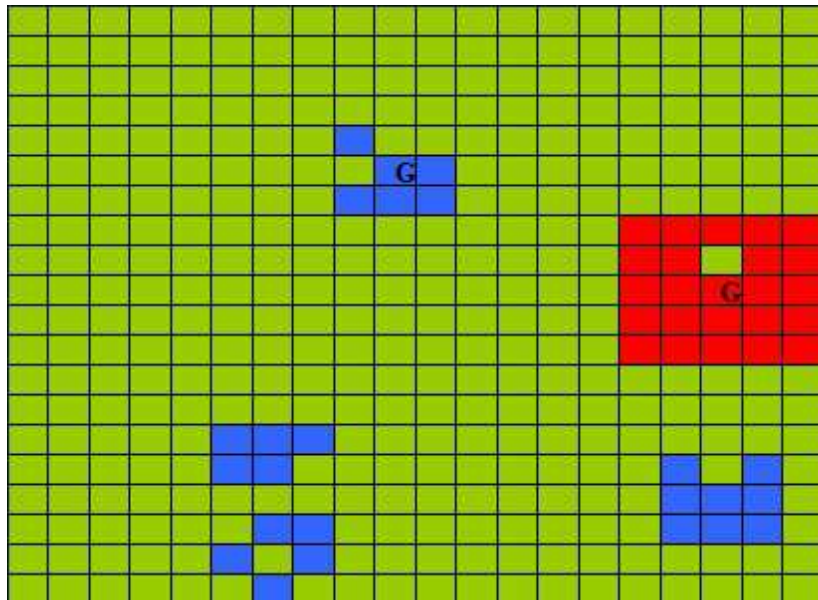


Figura 3.5: Un mapa sencillo sin obstáculos del juego *Robot Wars*

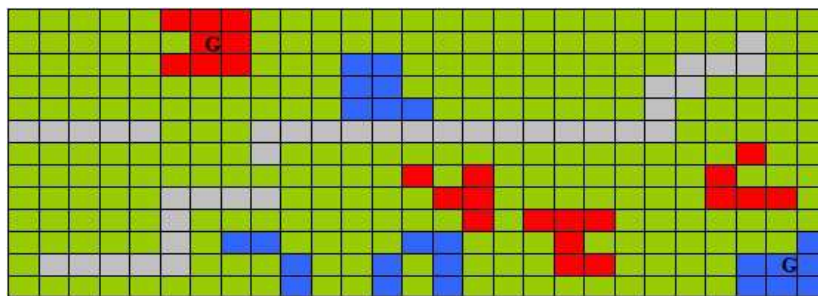


Figura 3.6: Un mapa con obstáculos del juego *Robot Wars*

CAPÍTULO 3. INTELIGENCIA ARTIFICIAL APLICADA A VIDEOJUEGOS

- **Salud:** Indica la vida restante de la unidad, cuando llega a cero la unidad muere, representando una baja para su ejército.
- **Velocidad de desplazamiento:** Especifica la movilidad/agilidad de una unidad sobre el escenario de batalla, se mide en el número de casillas que la unidad puede desplazarse hacia su objetivo en un solo turno.
- **Potencia de combate:** Indica la cantidad de salud que puede restar a una unidad rival tras atacarla en un determinado turno. Este parámetro es variable según las características tanto de la unidad atacante como las de la unidad atacada. No es lo mismo que un soldado dispare con su fusil al blindaje de un tanque a que un tanque dispare con su cañón a un soldado. Este parámetro pretende simular las ventajas/desventajas existentes en el combate según el contexto en el que se desarrolla la batalla y las características ofensivas/defensivas de cada unidad. Este parámetro está estrechamente vinculado con la salud, si la unidad atacante no goza de buena salud sus ataques se verán reducidos en potencia. Esta reducción será proporcional a su estado de salud, por ejemplo, si la unidad atacante se encuentra al 50% de salud entonces sólo dispondrá de un 50% de su potencia de combate original.
- **Radio de visión:** Mide la capacidad que tiene un determinado tipo de unidad para detectar o ver unidades amigas y enemigas en el mapa. Se mide en número de casillas, tomando como origen la posición de la propia unidad.
- **Radio de combate:** Es el alcance máximo que tiene una unidad para atacar a otra. Se mide en número de casillas, tomando como origen la posición de la propia unidad.

Las acciones que cada unidad puede realizar independientemente de sus especialidades y cualidades, son las siguientes:



CAPÍTULO 3. INTELIGENCIA ARTIFICIAL APLICADA A VIDEOJUEGOS

- **Atacar:** Ataque sobre una unidad enemiga cercana. La forma sobre la que se ataca, su alcance y potencia de ataque dependerá del tipo de la unidad atacante y de su estado de salud. Esto es, cuanto más salud tenga mayor será el daño que infringirá sobre una unidad enemiga cuando realice el ataque.
- **Avanzar:** Avance hacia la posición donde se encuentra la unidad del General enemigo. La cantidad de casillas que se avancen dependerá del tipo de unidad que sea.
- **Retroceder:** Avance hacia la posición donde se encuentra la unidad del General amigo. La cantidad de casillas que se avancen dependerá del tipo de unidad que sea.
- **Replegar:** Avance hacia la unidad amiga más cercana. Útil para reorganización y también para unir fuerzas con otras unidades amigas desperdigadas por el escenario.
- **Nada:** Cada vez que una unidad consume un turno realizando alguna acción de las anteriores ésta pierde un punto de energía, cuando su energía llega a cero la unidad no puede realizar más acciones a no ser que recupere dicha energía. La forma de hacerlo es mediante la ejecución de esta acción cuando le llegue el turno, de este modo recargará su energía y podrá volver a realizar cualquier otra acción en el siguiente turno.

El objetivo del juego consiste en conseguir destruir el ejército del enemigo sufriendo el menor número de bajas posibles. Las condiciones de fin de la partida son:

- Uno de los bandos pierde a todas sus unidades (sin incluir al General). La victoria pasa a ser del bando contrario.
- Uno de los bandos pierde a su General. La victoria pasa a ser del bando contrario.



Codificación de las soluciones

Las estrategias se codifican mediante una matriz de cuatro dimensiones, cada eje de la matriz representa uno de los siguientes parámetros:

- Tipo de unidad: identifica el tipo de unidad y puede tener siete posibles valores: general, infantería, blindados, aviación, antiaéreos y artillería.
- Cercanía relativa: especifica la cercanía al general amigo y al general enemigo. Se representa mediante un valor binario: 0 si la unidad está más cerca del general enemigo que de su general; 1 en el caso contrario.
- Ventaja numérica: también se representa mediante un valor binario, que si vale 0 indica que en el entorno cercano a una unidad concreta existe una mayor proporción de unidades amigas que de enemigas, y valdrá 1 en el caso contrario. Para calcular esto se define que: cada unidad u ubicada en una posición (x_u, y_u, z_u) del mapa tiene su propio rango visual que abarca cualquier posición que esté dentro de un alcance máximo τ ; de esta forma, cada unidad sólo recibe información de lo que está contenido dentro de su rango visual.
- Salud/energía: define la salud de la unidad, la cual establece cuantos turnos seguidos puede asumir esa unidad sin descansar. Tiene tres posibles valores: alta, media y baja.

Cada posición de la matriz actúa a modo de gen y almacena una acción de entre las siguientes:

- *Atacar*: ataca a un enemigo cercano.
- *Avanzar*: avanza hacia el general enemigo.
- *Retroceder*: avanza hacia el general amigo.
- *Replegar*: avanza hacia la unidad amiga más cercana.



- *Nada*: descanso por falta de energía.

La elección de la acción que se va aplicar a una unidad del ejército cuando le llega su turno está determinada por las coordenadas que le correspondan en los ejes de la matriz. Toda la matriz representa una estrategia que controla de forma determinista el comportamiento de un ejército durante el juego. De forma específica para cada unidad hay doce posibles estados diferentes (es decir, $2 \times 2 \times 3$, sería el total de las combinaciones considerando las últimas tres dimensiones de la matriz). Básicamente, en un turno del juego cada unidad del ejército ejecutará la acción que corresponda según su estado (respecto a las cuatro dimensiones). En cualquier instante del juego, el comportamiento del ejército será la suma de todas las acciones llevadas a cabo por cada una de las unidades que lo constituyen.

Generador de batallas

El módulo de generación de las batallas recibe como entrada un fichero XML donde se especifican las características del mapa que será generado posteriormente. Para el diseño del mapa se tiene en cuenta el número de unidades de cada tipo que dispondrá cada bando y el número de batallas que se deben generar. Se recibe como entrada la ruta del fichero de mapas donde se almacena la configuración (casilla a casilla) del mapeado de batalla, que es un fichero de texto en formato ASCII donde cada línea representa una fila de la matriz y cada carácter el contenido que debe tener la celda asociada. La interpretación del fichero se hace como sigue:

- #: indica una roca
- |: indica un árbol
- ~: indica agua
- , : indica llano o suelo accesible



CAPÍTULO 3. INTELIGENCIA ARTIFICIAL APLICADA A VIDEOJUEGOS

Con la información anterior el Generador de batallas devuelve como salida un conjunto de ficheros de texto ASCII. Estos ficheros siguen exactamente el mismo formato que los ficheros de mapas recibidos en la entrada, pero con la salvedad de que en ellos están ubicados las diferentes unidades de cada ejército. El Generador se encarga de ubicar a cada unidad en celdas de tipo suelo accesible, teniendo en cuenta los parámetros de entrada recibidos. De este modo el carácter (,) de un suelo accesible será sustituido por un carácter numérico (del 0 al 9) que representa al tipo de unidad y bando al que pertenece, para ello se sigue una codificación muy sencilla:

- 0 a 4: unidad del bando 1
- 5 a 9: unidad del bando 2

El mecanismo que sigue el Generador para realizar la distribución de las unidades responde a las diferentes formaciones que cada bando debe seguir. Un tipo de formación puede ser la *romana*, donde las unidades se distribuyen todas en un único bloque homogéneo rodeando a su general; y la de *guerrilla* que distribuye a las unidades en diferentes bloques dispersos por el escenario de batalla. El Generador siempre devuelve tantos escenarios de batallas como tipos de formaciones y casos distintos se hayan solicitado, teniendo en cuenta que para cada caso solicitado se generan tantos mapas de batallas como combinaciones diferentes se puedan realizar de las formaciones entre los bandos.

Simulación de las batallas

El simulador recibe como entradas las matrices de estrategia de ambos bandos y el mapa donde se desarrolla la batalla. Teniendo en cuenta las reglas de juego, la posición inicial, cantidad y tipos de unidades militares que poseen ambos bandos; se realiza una simulación en memoria del desarrollo de la batalla, devolviendo el resultado cuando ésta finaliza. Como finalización se entiende al estado alcanzado en el cual uno de los





Figura 3.7: Captura de pantalla de la simulación de una batalla en el juego *Robot Wars*

bandos obtiene la victoria, o bien a aquel en el que se alcanza un umbral de iteraciones tras el cual el simulador termina devolviendo empate. Aclaremos que esta simulación es determinista, por lo que no se genera la problemática del tratamiento de incertidumbre asociada a las evaluaciones con componente estocástico. En cada iteración se recorren todas las unidades operativas (no destruidas por el enemigo) de cada bando y se elige la acción que se debe realizar dependiendo del contexto en el que se encuentre. La Figura 3.7 muestra una captura de pantalla que corresponde con la simulación de una batalla en el juego.

3.5.2. El juego *Planet Wars*

Planet Wars es un juego de tipo RTS de dos jugadores que fue propuesto en el concurso internacional de *Google AI Challenge 2010*¹². En ese concurso los participantes compiten con una estrategia de IA creada para controlar uno de los *bot* del juego. En esta competición, el tiempo real se dividió en turnos de un segundo, a fin de que los

¹²<http://planetwars.aichallenge.org>

jugadores pudiesen jugar de forma secuencial, sin embargo, las acciones se suceden al mismo tiempo en la simulación. *Planet Wars* se puede considerar como una instancia sencilla de los juegos RTS, pues en él se genera la dinámica típica de este género pero las posibles acciones y el terreno están muy bien acotados, y esto hace que los posibles estados lógicos estén ser más controlados. Es por ello que en los últimos años dicho juego ha sido ampliamente usado en trabajos de investigación de IA.

El escenario del juego está definido por un mapa de dos dimensiones que contiene varios planetas. Los planetas pueden estar conquistados por uno u otro jugador, o ser neutrales (es decir, que no pertenecen a ningún jugador, son planetas libres). Un planeta tiene dos características principales: el número de naves y la tasa de crecimiento. La primera, como su nombre lo indica, es la cantidad de naves que existen en ese planeta, no se distingue por tipos, así que todas las naves son iguales y representan un mismo valor de fuerza. Los planetas que han sido conquistados aumentan su cantidad de naves en cada turno, ese aumento está definido por la tasa de crecimiento t , donde $t \in \{1, 5\}$, indica el número de naves que se suman al planeta en un turno de juego.

Véase la Figura 3.8, en ella se muestra una captura de pantalla del juego, los planetas rojos pertenecen al jugador 1, los verdes al jugador 2, y el resto son mundos neutrales. Aunque *Planet Wars* es un RTS, la implementación en el campeonato lo ha transformado en un juego basado en turnos, los jugadores disponen de un tiempo (número de turnos) determinado para conseguir la victoria. En cada turno un jugador recibe el estado actual del juego (es decir, información acerca de los planetas y las naves que existen en cada uno), la toma de decisiones se lleva a cabo usando sólo esa información actualizada del mapa, no se permite almacenar ninguna información sobre estados anteriores del mapa. De esta manera, la estrategia de IA que controla a los *bots* tiene que estar pensada para enfrentarse en cada momento a un mapa totalmente desconocido.

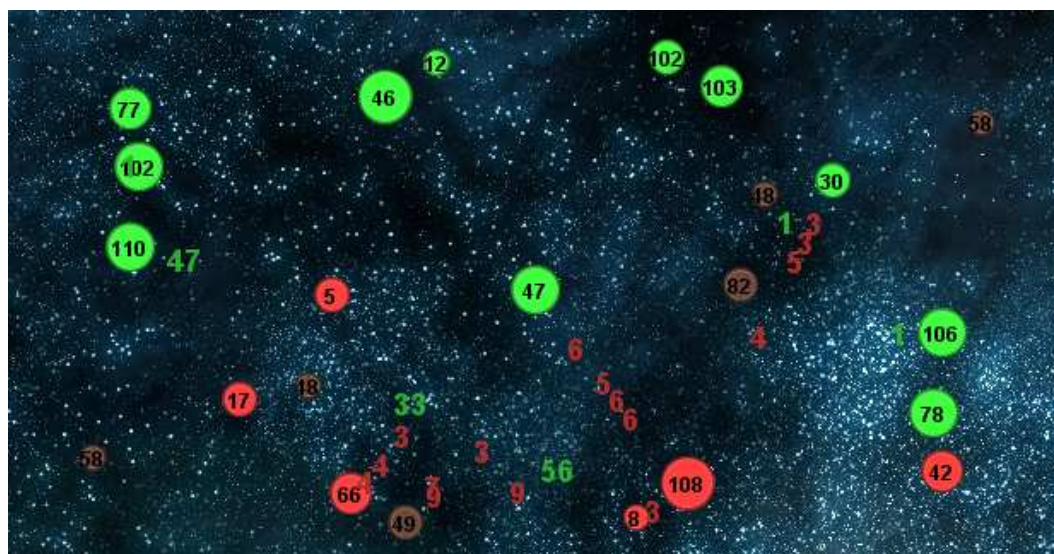


Figura 3.8: Captura de pantalla del *Planet Wars*: los planetas tienen un número que indica la cantidad de naves que hay en ese mundo. Los colores identifican la pertenencia de los planetas y las naves a uno u otro jugador.

Reglas de juego

Los *bots* sólo pueden hacer uso de una acción durante su turno de juego: enviar naves (o sea, flotas) a otros planetas y para ello deben indicar: el planeta destino, el origen (planeta desde el cual partirán las naves y que debe ser propiedad del jugador que ejecuta la acción), y el número de naves a enviar (que debe ser mayor que la cantidad de naves que habrá en el planeta destino en el momento en que llegue la flota allí o de lo contrario, no podrá conquistarlo).

Las flotas, pueden necesitar más de un pseudo-turno para llegar a su planeta destino (el tiempo es directamente proporcional a la distancia entre el planeta de origen y el destino). Cuando la flota llega un planeta enemigo o neutral, se produce un enfrentamiento, en el cual cada nave es sacrificada para destruir una nave enemiga, en otras palabras, resulta victoriosa aquella flota que albergue más naves. En caso de que el planeta destino sea del propio jugador, ambas flotas se unen, sumando sus naves. Las reglas que rigen la partida son las siguientes:

- El jugador con más naves al final del juego gana.
- Si uno de los jugadores pierde todas sus naves, la partida finaliza y el otro jugador gana.
- Si un jugador excede el límite de tiempo sin emitir la acción correspondiente a ese turno, pierde el juego.
- Si un jugador lanza un envío de flotas con más naves de las que posee el planeta emisor, pierde la partida.
- Si ambos jugadores tienen el mismo número de naves cuando el juego termina, se considera un empate.

Codificación de las soluciones

En esta sección se explicará la codificación que fue definida para representar un estrategia de juego. Dicha codificación se va a emplear más tarde (en las secciones que siguen) para poblar algoritmos coevolutivos cuyo objetivo será el de encontrar estrategias de juego óptimas para gobernar un *bot* en el *Planet Wars*. El proceso de búsqueda en estos casos se enfoca en optimizar las reglas que guían la toma de decisiones para que en cada turno el *bot* seleccione la mejor acción de acuerdo a su estado.

El estado de un *bot* en cada instante del juego está determinado por la ventaja (con respecto a su oponente) en términos de la tasa de crecimiento y la cantidad de naves. Ambas métricas de ventaja tienen tres valores posibles: ‘alto’, ‘nulo’ o ‘bajo’. De esta manera, para un jugador p (respecto a su oponente o), GR_p (respecto a GR_o) es la tasa de crecimiento total que depende de la tasa individual que posee cada planeta del jugador p . Sea: $\Delta GR_{po} = GR_p - GR_o$; entonces si $\Delta GR_{po} > 0$, significa que el jugador p tiene una ‘alta’ ventaja sobre su oponente o en términos de “tasa de crecimiento”; ‘nula’ es caso de un empate (i.e., $\Delta GR_{po} = 0$) y ‘baja’ si $\Delta GR_{po} < 0$.



El cálculo de la ventaja respecto a la cantidad de naves es similar, pero considerando NS_p en lugar de GR_p como el número total de naves propiedad del jugador p en cada instante del juego. En otras palabras, $\Delta NS_{po} = NS_p - NS_o$. Sin embargo, se consideran diferentes umbrales para los valores de ventaja, de modo que si $\Delta NS_{po} > 10$, el jugador p tiene una ventaja ‘alta’, ‘nula’ si $0 \leq \Delta NS_{po} \leq 10$ y ‘baja’ en otro caso. Nótese que consideramos que una diferencia de 10 naves no es lo suficientemente significativa para distinguir entre los dos jugadores, a este valor se llegó a través de nuestra experiencia con el juego.

Un jugador virtual (es decir, una estrategia de juego) es definida como una matriz bidimensional (ver Figura 3.9), en la cual la primera dimensión representa la ventaja respecto al “número de naves” (ΔNS_{po}) y la otra dimensión se refiere a la ventaja en términos de la “tasa de crecimiento” (ΔGR_{po}). Como se explicó anteriormente cada eje tiene tres posibles valores para cada ventaja: ‘alta’, ‘nula’ y ‘baja’. Cada celda de la matriz actúa como un gen y almacena una de las siguientes acciones:

1. Atacar al planeta enemigo más fuerte (AS, por sus siglas en inglés de *Attack Stronger*) (el planeta más fuerte es el que tiene la mayor cantidad de naves).
2. Atacar al planeta enemigo más débil (AW, por sus siglas en inglés de *Attack Weakest*).
3. Atacar al planeta enemigo más cercano (AC, por sus siglas en inglés de *Attack Closest*).
4. Conquistar al planeta neutral más fuerte (CS, por sus siglas en inglés de *Conquer Stronger*).
5. Conquistar al planeta neutral más débil (CW, por sus siglas en inglés de *Conquer Weakest*).

6. Conquistar al planeta neutral más cercano (CC, por sus siglas en inglés de *Conquer Closest*).
7. Seguir al enemigo (FE, por sus siglas en inglés de *Follow Enemy*), en esta acción se mandan tropas hacia uno de los planetas donde que el enemigo u oponente está enviando tropas en el turno actual.

De esta manera, la matriz representa una estrategia que puede controlar de forma determinista el comportamiento de un *bot*, ejecutando la acción que corresponda en cada instante de juego. Para un jugador virtual existen nueve posibles estados (i.e., 3×3 , todas las combinaciones considerando las dos dimensiones de la matriz). Teniendo una estrategia de juego, la toma de decisiones se reduce a ejecutar la acción que está almacenada en la matriz para el estado en que el *bot* percibe que se halla en ese instante del juego. Véase en la Figura 3.9 un ejemplo de la matriz que muestra la distribución de las acciones para cada posible estado, por ejemplo, cuando el jugador tiene una alta ventaja en términos de “tasa de crecimiento” y “número de naves” ($\Delta GR_{po} > 0$ y $\Delta NS_{po} > 10$) la acción correspondiente es “AS”.

	$\Delta NS_{po} > 10$	$0 \leq \Delta NS_{po} \leq 10$	$\Delta NS_{po} < 0$
$\Delta GR_{po} > 0$	AS	AC	AW
$\Delta GR_{po} = 0$	AS	AC	CW
$\Delta GR_{po} < 0$	FE	AS	AW

Figura 3.9: Una matriz de acciones en el juego *Planet Wars* que fue generada a partir de uno de los algoritmos que presentaremos más adelante en esta memoria.

Si se analiza el espacio de búsqueda que se debe explorar para encontrar las mejores soluciones (i.e., posibles estrategias de juego) vemos que su tamaño es de $7^9 =$

$40353607 \in [2^{25}, 2^{26}]$, por lo cual no debería ser evaluado de forma exhaustiva por el coste computacional que esto supondría, por tanto es más conveniente utilizar técnicas metaheurísticas que optimicen el proceso de búsqueda.

3.6. Conclusiones del capítulo

En general, las tendencias futuras de los videojuegos han abierto un camino en el cual la IA clásica y la IA para videojuegos convergen (en algunos casos concretos) hacia el objetivo común de imitar el comportamiento humano. De una parte la IA y de la otra la Computación Afectiva se complementan en los videojuegos para hacer que crezca la expectativa de que en un futuro no muy lejano un *bot* estará a la altura de cualquier concursante del célebre *Test de Turing*. En este sentido, las metaheurísticas han sido protagonistas en muchos de los casos de éxitos que están documentados en la literatura científica. La Coevolución, y en particular su enfoque competitivo, destaca de forma relevante, mostrando sus potencialidades en el área de los videojuegos e incitando a la búsqueda de modelos más eficientes.

La PCG es un campo de creciente interés para la comunidad científica, con un elevado número de publicaciones. Pero no sólo es atractivo para los científicos, la industria del videojuego está incorporando de forma satisfactoria muchos de los avances que se consiguen en el ámbito investigativo.

Existen multitud de áreas asociadas con el empleo de técnicas de Inteligencia Artificial/Computacional, no tratadas de forma específica en este capítulo, que suponen otros desafíos para los investigadores y entre las cuales podemos mencionar: el modelado del comportamiento del jugador humano, la definición de lenguajes de modelado para la descripción de videojuegos, creación de métodos de narrativa computacional, diseño de juegos asistidos por IA, y un largo etc.. No cabe duda de que nos enfrentamos a retos realmente estimulantes, no sólo para un futuro cercano sino para el actual presente.



CAPÍTULO 3. INTELIGENCIA ARTIFICIAL APLICADA A VIDEOJUEGOS

En los siguientes capítulos se describen los resultados del trabajo de investigación realizado en esta tesis. Consideramos que con nuestros aportes hemos contribuido al avance en algunas de las líneas de investigación aquí discutidas y esperamos continuar haciéndolo en trabajos futuros.



Capítulo 4

Generación de estrategias de juegos mediante Coevolución

Bless all forms of intelligence

The Instructor (in The Animatrix)

Una de las líneas de investigación más interesante en el desarrollo de videojuegos es la creación de técnicas de IA para los juegos RTS. Las características de este género y la complejidad del espacio de búsqueda que genera (un posible estado describe un escenario de juego muy amplio, con cientos de unidades que actúan simultáneamente, y muchos miles de posibles posiciones para cada uno de los cientos, posiblemente miles, de unidades) hacen que el diseño de IA para este tipo de juego sea al mismo tiempo un reto muy interesante y una tarea difícil de manejar. Esta complejidad viene dada por la dificultad intrínseca que poseen los grandes espacios de búsqueda y por la propia naturaleza de estos juegos puesto que, a diferencia de los juegos tradicionales, en estos se pueden realizar un número indeterminado de movimientos de forma simultánea



CAPÍTULO 4. GENERACIÓN DE ESTRATEGIAS DE JUEGOS MEDIANTE COEVOLUCIÓN

(Corruble et al., 2002).

Los juegos RTS proporcionan un amplio abanico de problemas difíciles para el diseño de la IA, por ejemplo: la planificación en un mundo con información incompleta, el auto-aprendizaje, el modelado del comportamiento de los oponentes, y el razonamiento espacial y temporal sólo por nombrar algunos. Uno de los retos más significativos es el de la creación de *bots* que simulen el comportamiento humano y no necesiten hacer trampas para obtener un resultado ventajoso respecto al oponente humano. Citando a Lucas et al. (Togelius et al., 2013), esto implica, básicamente, “Restringir la información disponible para el jugador controlado por la IA a la información que un jugador humano puede ser capaz de reunir en el juego, y limitar también sus posibles acciones a las del jugador humano”. En todo momento, el *bot* debe representar un desafío para el humano, independientemente de sus habilidades y de la estrategia que adopte para jugar.

Este capítulo trata sobre la generación automática de IA para los NPCs en videojuegos RTS. Las propuestas que se presentarán utilizan la Coevolución como metaheurística que guía el proceso de búsqueda y auto-aprendizaje pues consideramos que es una de las técnicas más interesantes dentro del área de la CE para ser aplicadas en juegos. Como en las demás partes de esta tesis, nos centraremos en el enfoque competitivo de la Coevolución, en el cual los individuos compiten entre sí para mejorar su valor de *fitness* a costa de empeorar el de su oponente. Y esta simulación de la relación depredador/presa (la misma que tiene lugar en la naturaleza) va generando un proceso al estilo de una “carrera armamentista” en el que la mejora de algunos individuos estimula la mejora en los oponentes, y viceversa.

Como se ha visto en capítulos anteriores, la Coevolución ha venido mostrando resultados exitosos en varios trabajos de investigación en el marco de los videojuegos. Pero también son muchos los trabajos que se han dedicado al estudio de las patologías que afectan los modelos coevolutivos. Es la evaluación una de las etapas más importantes



y arriesgadas en la Coevolución, en ella se produce la interacción entre las partes de la carrera armamentista y es ahí donde se decide el foco a seguir en la optimización. Patologías como, los ciclos, el desencaje de la búsqueda, la sobre-especialización, entre otras, tienen su lugar garantizado si el mecanismo evaluativo no es lo suficientemente eficaz.

Esta necesidad de definir modelos coevolutivos robustos fue el tema principal que motivó la investigación llevada a cabo en este capítulo. Los modelos que presentaremos aquí están inspirados en la idea de Rosin y Belew de utilizar un Salón de la Fama (HoF, por sus siglas en inglés) a modo de una “memoria histórica” que permitan usar campeones de antiguas generaciones en la evaluación de los individuos de generaciones futuras. Diferentes variantes del enfoque HoF han sido definidas (Lichocki, 2008; Avery, 2008) en la comunidad científica, y es objetivo de este trabajo proponer nuevos enfoques del HoF que optimicen su desempeño.

El aporte principal de esta parte de la memoria es el diseño de nuevos enfoques que mejoran la eficiencia y la eficacia del HoF como mecanismo de manejo de memoria a largo plazo dentro del ciclo coevolutivo. A lo largo de este capítulo se describirán los nuevos enfoques del HoF que han sido definidos y se discutirán los resultados obtenidos en los experimentos realizados en dos juegos RTS, el *Robot Wars* (ver Sección 3.5.1) y el *Planet Wars* (ver Sección 3.5.2). Estos resultados han sido previamente publicados en (Nogueira et al., 2012b; Nogueira et al., 2013; Nogueira et al., 2014a).

4.1. El enfoque completo del Salón de la Fama (HoFC) y su aplicación al *Robot Wars*

Las siglas HoFC se refieren al enfoque “completo” del Salón de la Fama, se trata de un método que combina el HoF con la filosofía del Torneo de Dominancia (Sección 2.2.2) estableciendo que: en una generación n el individuo que se incorpore al Salón

CAPÍTULO 4. GENERACIÓN DE ESTRATEGIAS DE JUEGOS MEDIANTE COEVOLUCIÓN

de la Fama tiene que haber vencido al elenco completo del bando rival, el cual estaría integrado por el conjunto I_1, I_2, \dots, I_n donde cada I_k es el individuo dominante de la generación k .

El HoFC es un algoritmo genético de tipo $(\mu + 1)$ al que se le aplica un enfoque CC. La población inicial se genera aleatoriamente. El bucle generacional tiene un máximo de 300 generaciones, en cada iteración se seleccionan dos padres utilizando torneo binario, luego se lleva a cabo la recombinación genética basada en el cruce de *Bernoulli*, le sigue la mutación por variación aleatoria de bits, y por último un reemplazo elitista de tipo “mejor por peor”.

La búsqueda se lleva a cabo mediante la coevolución por turnos de múltiples estrategias en cada ejército. El objetivo es encontrar una estrategia ganadora y luego tomarla como base en la siguiente ejecución para hallar otra estrategia en el bando contrario que gane a la anterior. De esta manera, en cada turno coevolutivo uno de los dos ejércitos evoluciona y el resultado de esta evolución debe ser una “estrategia victoriosa” que logró vencer a las estrategias enemigas enfrentadas. Si al concluir el turno evolutivo no se obtiene una solución, se echa a andar otra vez la Coevolución para ese mismo ejército, hasta obtener una estrategia victoriosa o hasta alcanzar el límite máximo de coevoluciones. El siguiente turno coevolutivo le corresponderá al otro ejército y los enemigos a enfrentar por esa población serán el conjunto de estrategias victoriosas que ha encontrado el bando contrario en la carrera coevolutiva recién terminada. Así sucesivamente se van alternando los turnos, al final se obtienen dos conjuntos de soluciones (uno para cada bando) que contienen las estrategias victoriosas de cada ejército. En adelante llamaremos “bando A” al ejército que comienza siendo el protagonista (bando amigo) de la primera carrera coevolutiva, y “bando B” al que inicialmente ejerce de oponente (bando enemigo).

La evaluación se hace enfrentando cada individuo de la población que está realizando el turno coevolutivo contra todos los miembros del HoF del bando oponente,



y promediando el resultado que obtengan en las batallas. Veamos los detalles de este cálculo en la siguiente ecuación:

$$P_i = \frac{\sum_{j=1}^k R_{ij} + Vbajas_{ij} + Vsalud_{ij}}{k} \quad (4.1)$$

Donde k es la cantidad de miembros del HoF del bando oponente; R_{ij} es el resultado que obtiene el individuo i en su enfrentamiento contra el oponente j , su valor puede ser: victoria (1000 puntos), empate (500 puntos), o derrota (0 puntos). $Vbajas_{ij}$ y $Vsalud_{ij}$ contienen la ventaja que obtuvo la estrategia i con respecto a j , en cuanto al número de bajas y la salud de las unidades respectivamente.

El empleo del HoF garantiza un mecanismo de evaluación que tiene en cuenta a los viejos campeones, y propicia que la búsqueda evolutiva se base en superar a los mejores resultados hallados hasta ese momento, creando una relación de transitividad entre las estrategias del elenco. Pero independientemente de estas ventajas se puede pensar en la posibilidad de diseñar una manera más eficiente de emplearlo. Tengamos en cuenta que al inicio de la carrera coevolutiva la presión de selección es menor que en un punto más avanzado de ésta donde exista mayor cantidad de oponentes. Esto provoca que con la marcha de las iteraciones coevolutivas los nuevos individuos tenderán a ser más robustos que los primeros insertados en el salón, incluso aunque sus valores de *fitness* no sean muy diferentes.

De lo anterior se pueden sacar dos puntos claves. El primero es que en el elenco de la fama podrían haber individuos que fueron campeones en su momento pero que en iteraciones coevolutivas posteriores no tengan mucho que aportar al proceso de aprendizaje. Esto hace que su permanencia en el conjunto sea digna de valorar, pues pensemos que cuantos más miembros existan en la memoria de oponentes, más enfrentamientos habrá que hacer durante la evaluación, y mayor será el consumo computacional del algoritmo. El otro punto es que la medida del *fitness* empleada podría tener un mayor alcance si

abarcase otros parámetros. Por ejemplo, una buena práctica a la hora de implementar una función para evaluar el desempeño de un individuo es tener en cuenta elementos de su entorno que pueden influir directamente en su valor de *fitness*. Para ilustrar esto retomemos el proceso evaluativo del HoFC, pensemos en que el salón tiene estrategias que son más fáciles de vencer que otras, sin embargo, no existe distinción entre vencer a un oponente fácil y hacerlo con uno complejo. Si la función del *fitness* empleada en el HoFC tuviese en cuenta los elementos del entorno en que se desarrolla el individuo evaluado -uno de ellos podría ser la facilidad con que es vencido cada oponente- brindaría una medida más fiable y representativa de la situación real del individuo, además permitiría distinguir cuáles miembros del elenco son los más robustos.

A continuación se explican los algoritmos que han sido diseñados a partir del modelo básico HoFC. Con ellos se pretende analizar y explorar los dos puntos claves antes discutidos.

4.1.1. Algoritmo reducido del Salón de la Fama (HoFR)

La principal motivación de este algoritmo fue determinar en qué medida es necesario que se emplee un enfoque completo del Salón de la Fama en el algoritmo HoFC. El HoF, como se ha mencionado, resulta útil para garantizar la transitividad entre las soluciones, lo que permite coevolucionar las poblaciones mediante un proceso de aprendizaje continuo que perfecciona las estrategias de forma incremental. Pensar en la posibilidad de lograr estos resultados con una mayor eficiencia, nos condujo al diseño de un enfoque “reducido” (HoFR) del método del HoF.

Recordemos que el enfoque completo tiene en cuenta todos los miembros del elenco a la hora de evaluar los individuos, esto implica que en cada iteración coevolutiva se deben realizar tantos enfrentamientos por cada individuo como cantidad de miembros tenga el salón. En cambio, un enfoque reducido plantea que no es necesario evaluar al individuo contra cada miembro del salón, sino que sólo se hace con un subconjunto

pequeño de dicho elenco. El objetivo del algoritmo HoFR es minimizar el número de

Algoritmo 2: Algoritmo HoFR

```

1 cargar_batalla();
2 inicializar_simulador();
3 inicializar_bandos();
4 for ncoevolucion = 0 to maxCoevoluciones - 1 do
5     inicializar_pop(poblacion);
6     while generacion < maxGeneraciones  $\wedge$   $\neg$ es_victoria(mejor_individuo)
7         do
8             padres  $\leftarrow$  seleccionar(poblacion);
9             hijos'  $\leftarrow$  cruzar(padres, pX);
10            hijos''  $\leftarrow$  mutar(hijos', pM);
11            poblacion'  $\leftarrow$  reemplazar(poblacion, hijos'');
12            poblacion  $\leftarrow$  poblacion';
13            evaluar(poblacion);
14            mejor_individuo  $\leftarrow$  obtener_mejor_individuo(poblacion);
15            if es_victoria(mejor_individuo) then
16                bando_amigo.adicionar_al_salon_fama(mejor_individuo);
17                limite_coevoluciones  $\leftarrow$  0;
18                cambiar_bandos(); // se le asigna el turno coevolutivo al
19                    bando que es ahora el bando enemigo
20            else
21                incrementar(ncoevolucion);
22                incrementar(generacion);
23            end if
24        end while
25    end for

```

enfrentamientos que se llevan a cabo para la evaluación de los individuos durante la carrera coevolutiva, y comprobar si de esta manera se sigue logrando la transitividad entre las soluciones. El enfoque completo del HoF hace que en el paso coevolutivo k cada posible solución del jugador A se evalúe enfrentándose a $\{B_1, B_2, \dots, B_k\}$ donde B_i es la solución obtenida por el jugador B en el paso coevolutivo i (y que se obtuvo a su vez enfrentándose a $\{A_1, \dots, A_{i-1}\}$). Luego se le halla la media aritmética a la puntuación que acumula el individuo en sus k enfrentamientos y en función de esta se le asigna un resultado que puede ser: victoria, empate o derrota. Por su parte, el enfoque reducido

propone que en el paso k los individuos del bando A se enfrenten únicamente a B_k , como asumiendo que se dará una transitividad en la evaluación y que si A vence a B_k también vencerá a $\{B_1, \dots, B_{k-1}\}$.

La transitividad entre las soluciones victoriosas no tiene por qué darse necesariamente, pero es posible que en la práctica se logre. Al menos se espera que el HoFR permita realizar una exploración más amplia del espacio de búsqueda a un coste computacional mucho menor, pues en cada paso coevolutivo el coste de la evaluación se hace constante -en un único combate- mientras que con un Salón de la Fama completo en el paso k para cada evaluación se realizan k combates con las k previas soluciones del rival. Véase en el Algoritmo 2 el esquema general de funcionamiento del HoFR.

Para inicializar la población, se generan los individuos a partir de la mejor solución encontrada en el paso coevolutivo anterior. La motivación de esta idea viene del clásico problema del Viajante de Comercio (TSP, del inglés Travelling Salesman Problem), en el que puede ser provechoso modificar la matriz de distancia con respecto a una ejecución anterior. Es posible que la solución óptima sea similar a la que se tenía anteriormente, por lo que generando perturbaciones de dicha solución se puede partir de soluciones de calidad.

El problema que se plantea aquí difiere del TSP, en este caso no hay una traslación tan directa entre la representación de una solución y su rendimiento, pues una perturbación de una estrategia puede dar lugar a que funcione de forma muy distinta; pero se puede asumir que aunque en menor medida dicha relación existe, y a partir de esto, emplear el mismo principio. Entonces se parte de dos soluciones para dos participantes del juego A y B . El primer paso sería optimizar A para vencer a B , y para ello se comienza con una población inicial que consiste en perturbaciones de A . En el siguiente paso se haría lo mismo con B , y así sucesivamente. Esta inicialización puede ayudar a propiciar la transitividad entre las soluciones ganadoras, pues se garantiza similitud en los genes de los nuevos individuos con respecto al campeón anterior, de

manera que una combinación de acciones que haya sido provechosa en la coevolución anterior puede reutilizarse.

Experimentos del HoFR en el juego *Robot Wars*

El objetivo de los experimentos es valorar si el enfoque reducido mejora en algún aspecto los resultados del algoritmo HoFC, y sobre todo comprobar si logra mantener la relación transitiva entre las soluciones.

Tabla 4.1: Parámetros usados en los experimentos

Parámetro	Valor
<i>ejecuciones</i>	15
<i>maxCoevoluciones</i>	100
<i>maxGeneraciones</i>	300
<i>tamañoPoblación</i>	100
<i>pX</i>	0,9
<i>pM</i>	$1/\text{cantidad_Genes}$

Para esto se realizan varias ejecuciones de ambos algoritmos y luego se comparan los resultados en cuanto a la duración del ciclo coevolutivo, la calidad de las soluciones obtenidas, la transitividad en el conjunto solución, y el tiempo de ejecución. Los parámetros usados en todos los experimentos que realizaremos en esta investigación se muestran en la Tabla 4.1.

Resultados de la duración de los ciclos coevolutivos

Este indicador se refiere a la cantidad de carreras coevolutivas que logra un algoritmo. Permite determinar cuál de ellos obtiene un ciclo coevolutivo más activo y logra una mayor exploración del espacio de búsqueda. Es necesario aclarar que para

CAPÍTULO 4. GENERACIÓN DE ESTRATEGIAS DE JUEGOS MEDIANTE COEVOLUCIÓN

esta prueba se utilizaron valores diferentes a los mencionados en la explicación de la Ecuación (4.1) para puntuar los resultados de victoria (en este caso es de 2000 puntos) y empate (en este caso fue de 1000 puntos). Es por eso que el rango del valor del *fitness* que se observa en las Figuras 4.1 y 4.2 oscila entre los 1100 y 2700 puntos. Note además que en ambas figuras aparece marcado el valor de 2000 en el eje del *fitness* para indicar que ese es el umbral a partir del cual se considera “victoriosa” una solución.

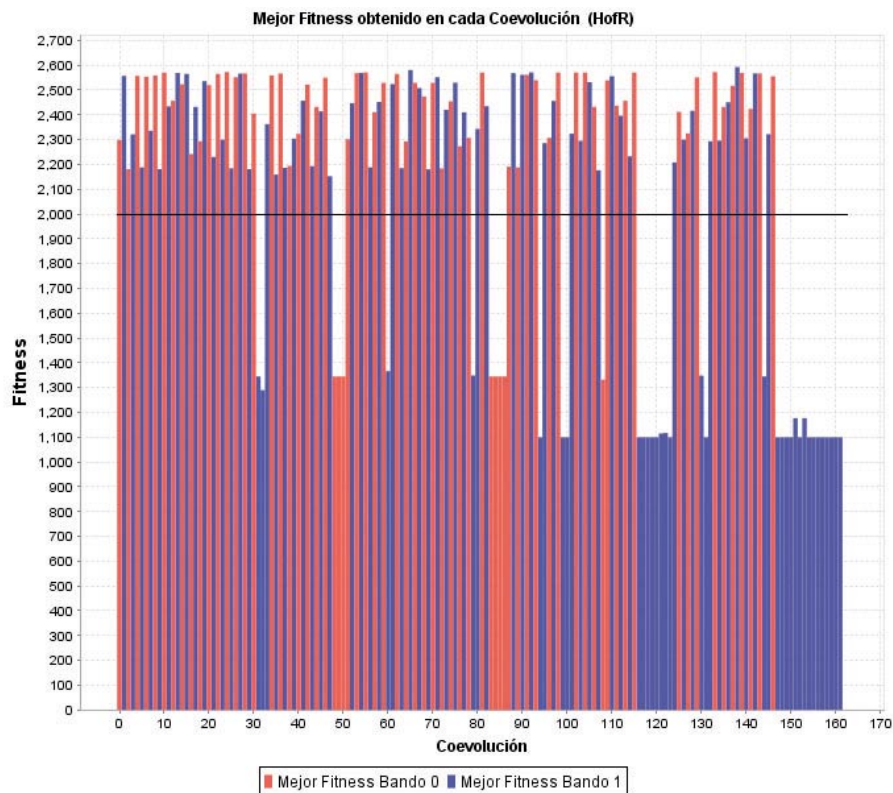


Figura 4.1: Cantidad de coevoluciones logradas en una ejecución del HoFR

Recordemos que el ciclo coevolutivo itera mientras no se alcance el límite de coevoluciones fallidas (que en este caso es de 15), el cual se refiere al número de coevoluciones consecutivas que realiza un bando sin lograr obtener una solución que venza al rival. Cuando uno de los bandos alcanza ese límite de coevoluciones sin éxito, se termina la ejecución. Vea en ambas figuras que llegado a un punto el bando azul ya no puede su-

CAPÍTULO 4. GENERACIÓN DE ESTRATEGIAS DE JUEGOS MEDIANTE COEVOLUCIÓN

perar el resultado del bando rojo y el algoritmo termina después de haberse ejecutado 15 coevoluciones continuas del bando azul.

Observe en las figuras 4.1 y 4.2 que para el caso del HoFR el ciclo coevolutivo es más activo. Este resultado se debe a que en este enfoque disminuye la presión de selección con respecto al HoFC, pues es menor la cantidad de enemigos que se deben vencer para coronarse como campeón. Esto hace que casi todas las coevoluciones logren una solución victoriosa, por eso se extiende la duración del ciclo coevolutivo.

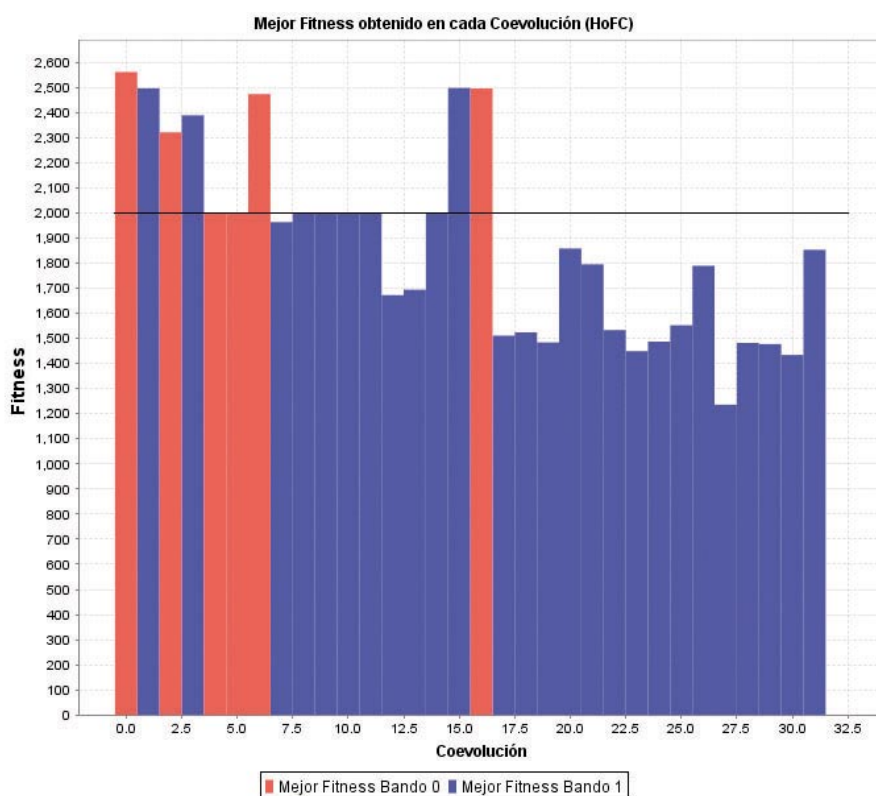


Figura 4.2: Cantidad de coevoluciones logradas en una ejecución del HoFC

Por su parte, el algoritmo HoFR alcanzó más de 160 coevoluciones, las de mejor *fitness* oscilaban su valor entre los 2150 y casi 2600 puntos. En el caso del HoFC se obtiene un número de coevoluciones mucho menor, de un total de 32 sólo siete lograron encontrar una solución vencedora, aquí el *fitness* de las soluciones oscilaba entre los

2300 y casi 2600 puntos; lo que demuestra que a pesar de no tener un ciclo coevolutivo tan activo, la calidad de las estrategias vencedoras eran muy similares entre ambos algoritmos.

Este resultado puede considerarse como que en el HoFR se logra hacer una exploración mayor del espacio de búsqueda, y se pudiera valorar como un buen resultado si este incremento conduce a buenas soluciones finales. Veamos lo que ocurre con las otras pruebas realizadas antes de llegar a conclusiones.

Resultados de la calidad de las soluciones

Para evaluar la calidad de las soluciones llamemos $\{A_0, \dots, A_{n-1}\}$ y $\{B_0, \dots, B_{n-1}\}$ a los miembros del Salón de la Fama obtenidos por el HoFC para el bando A y el bando B respectivamente, y $\{A'_0, \dots, A'_{n-1}\}$ $\{B'_0, \dots, B'_{n-1}\}$ a los del HoFR. Se harán dos pruebas, la primera se basará en realizar los enfrentamientos: $(A \text{ vs } B)$ y $(A' \text{ vs } B)$ para cada una de las quince ejecuciones, y para la segunda se enfrentarán $(A \text{ vs } B')$ y $(A' \text{ vs } B')$. De esta manera se analizará el desempeño de las estrategias obtenidas por cada algoritmo frente a un mismo oponente.

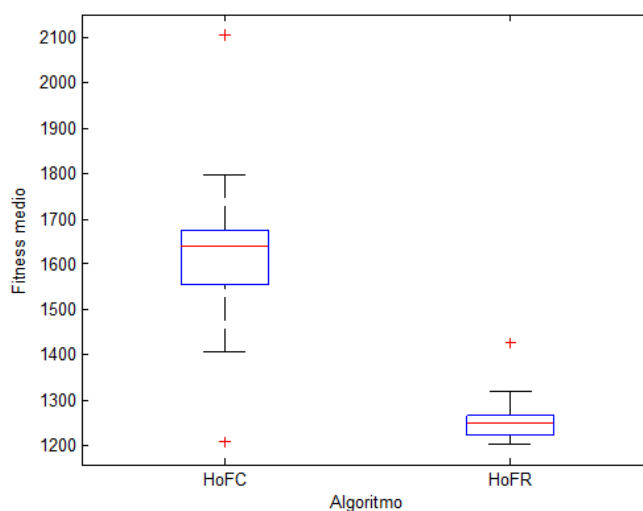


Figura 4.3: Resultados de los enfrentamientos contra un bando (el B) del HoFC (o sea, $(A \text{ vs } B)$ y $(A' \text{ vs } B)$)

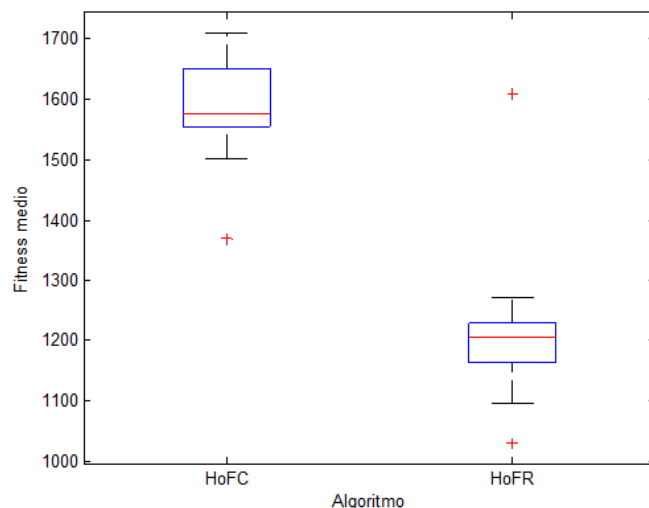


Figura 4.4: Resultados de los enfrentamientos contra un bando (el B') del HoFR (o sea, (A vs B') y (A' vs B'))

En la Figura 4.3 se muestran los resultados de la primera prueba de calidad realizada. El conjunto de datos analizados está formado por los valores de la media aritmética del *fitness* que obtuvo cada algoritmo en los enfrentamientos. La gráfica muestra claramente como el HoFC supera en todos los casos al HoFR. Para validar desde un punto de vista estadístico esta diferencia se aplicó el test de la suma de rangos de *Wilcoxon* que devolvió los valores de: $h = 1, p = 0,000024$ demostrando así que la diferencia entre los valores de *fitness* es estadísticamente significativa.

La Figura 4.4 muestra los resultados de la segunda prueba de calidad. Al igual que en la prueba anterior, el algoritmo HoFC supera el desempeño del HoFR; el test de *Wilcoxon* corroboró estos resultados al refutar la hipótesis nula para un valor de $p = 0,000057$. Por tanto, se puede concluir que las estrategias obtenidas mediante el HoFC tienen mayor calidad que las logradas por el algoritmo HoFR.

Resultados del chequeo de transitividad

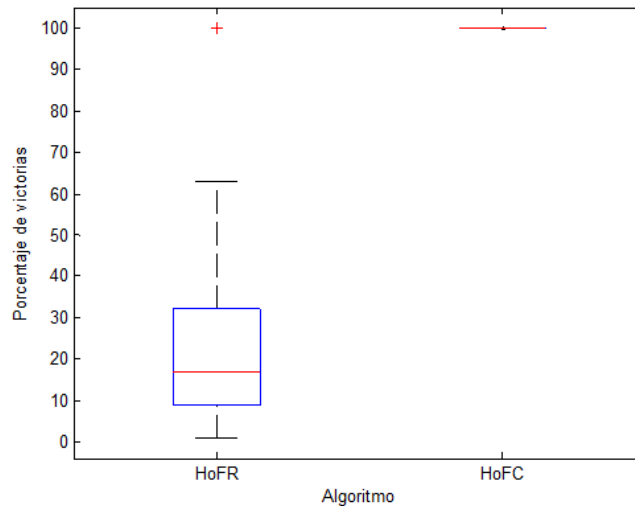


Figura 4.5: Prueba de transitividad en los enfoques HoFC y HoFR

Recordemos que para alcanzar la condición de solución victoriosa e incluirse al elenco de la fama en el HoFC era necesario superar todas o la mayoría de las soluciones del elenco, con lo cual se aseguraba que los nuevos miembros eran tan buenos o mejores (en relación al *fitness*) que los anteriores. Por otra parte, en el HoFR para ingresar en la memoria de las estrategias victoriosas basta derrotar a la última solución agregada. Al inicio valoramos que exista la posibilidad de que la propiedad transitiva garantice que esta última estrategia sea capaz de derrotar a todos sus oponentes ancestrales. Si esto se cumple no hay razón para que en el HoFR se obtengan soluciones de menor calidad; pero hemos comprobado en la práctica que los resultados son otros. Veamos qué sucede cuando se le aplica una prueba de chequeo de transitividad a los conjuntos de soluciones.

La prueba de transitividad se hace para una ejecución del algoritmo, en ella se enfrenta cada estrategia A'_i del elenco contra el conjunto $\{B'_0, \dots, B'_{i-1}\}$ (que serían los miembros del HoF del bando opuesto), de esta manera se comprueba si todas las estrategias del conjunto son capaces de vencer a sus oponentes ancestrales.

La Figura 4.5 muestra el comportamiento del porcentaje de victorias que obtuvo

cada estrategia analizada, para ambos algoritmos. Este porcentaje tiene en cuenta el total de oponentes ancestrales que debía vencer el individuo y la cantidad de victorias que realmente logró. Un resultado ideal es el que obtiene el HoFC, nótese que todas las estrategias logran un 100% de victorias, lo cual significa que los miembros del elenco pueden vencer a todos sus oponentes ancestrales, por tanto, la transitividad entre las soluciones se da perfectamente. Un resultado muy diferente se logra para el HoFR, aquí cada estrategia muestra un comportamiento aleatorio que no guarda relación con el de sus antecesores, sólo la primera estrategia insertada en el salón logra obtener un 100% de victorias, del resto ninguna alcanza ese valor.

El hecho de que la transitividad no se logre entre las estrategias del conjunto solución del algoritmo HoFR provoca que la presión selectiva durante la búsqueda sea insignificante con respecto a la del HoFC y esto, sin duda, frustra el proceso de aprendizaje continuo e impide la mejora iterativa de las soluciones. En cada coevolución se obtiene una estrategia que solamente es capaz de vencer al individuo más nuevo del elenco. Así que es esta la causa de que las soluciones logradas por el enfoque reducido sean inferiores a las que logra el enfoque completo. Y de aquí se concluye que sí es necesario emplear un enfoque Salón de la Fama completo, o al menos uno que tenga en cuenta durante la evaluación a la mayoría de los miembros del elenco, para poder garantizar la transitividad.

Resultados del tiempo medio de las coevoluciones

Por último se analizan los resultados de las pruebas de medición del tiempo medio (en segundos) que tardan las ejecuciones de cada algoritmo. La Figura 4.6 evidencia que el HoFR logra las coevoluciones en un tiempo más corto que el HoFC. Este comportamiento se debe a los resultados vistos anteriormente que hacen que en el HoFR sea más fácil encontrar una estrategia victoriosa. Este resultado habría sido exitoso de no haberse afectado la calidad de las soluciones con la minimización de la cantidad de

enfrentamientos.

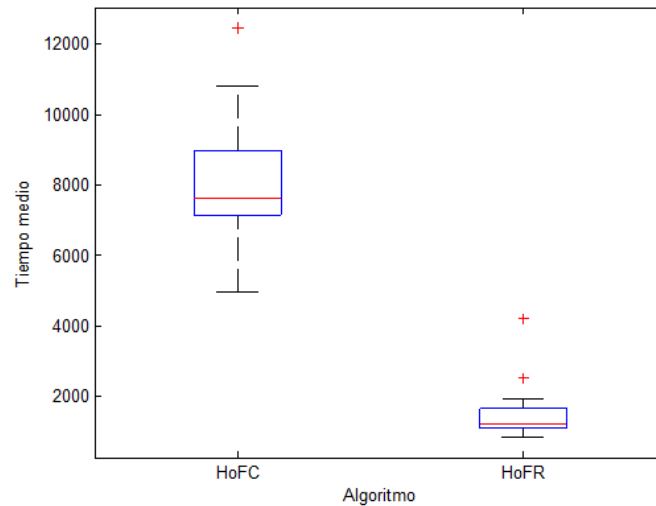


Figura 4.6: Tiempo medio (en segundos) que tardan las ejecuciones de cada algoritmo

El HoFR demostró la necesidad de emplear el enfoque completo del HoF, o al menos uno que tenga en cuenta la mayoría de los miembros del elenco, para poder lograr la transitividad en el conjunto solución. Al fallar la propiedad transitiva, la continuidad del proceso de aprendizaje se frustra. Por otra parte, el consumo computacional del algoritmo disminuye notablemente con la reducción de la cantidad de enfrentamientos en la evaluación de los individuos. Es hora de pensar en otra propuesta que también reduzca los enfrentamientos, pero sin llegar a afectar la transitividad. Así se lograría un mejor rendimiento del algoritmo y se mantendría la calidad de las soluciones. La siguiente sección describe un nuevo algoritmo que pone en práctica esta idea.

4.1.2. Algoritmo que optimiza el Salón de la Fama (HoFU)

El primer intento por mejorar los resultados de la solución coevolutiva que se usa originalmente en el juego fue el HoFR, y no dio buenos resultados. El algoritmo HoFU (del inglés: Hall of Fame Updating) es un segundo intento de hallar un enfoque del HoF que sea más eficiente que el original y que mantenga la calidad en el proceso de

búsqueda y optimización.

Los resultados del algoritmo HoFR enfatizaron la necesidad de propiciar la transitividad entre las soluciones victoriosas en el ciclo coevolutivo. También mostraron la relación directa que existe entre el consumo computacional del algoritmo y el número de enfrentamientos que se hacen durante la evaluación de los individuos. La vía usada para garantizar la transitividad en el HoFC es enfrentar a cada individuo contra todos los miembros del HoF, y el intento del HoFR de minimizar esta cantidad de enfrentamientos no tuvo éxito. Sin embargo, es posible diseñar otra variante que también logre una disminución del número de oponentes a vencer, sin que se llegue a afectar la calidad de las soluciones. Es ese precisamente el objetivo que persigue el algoritmo abordado en esta sección.

En el HoFU se propone también una disminución de la cantidad de enfrentamientos pero no tan drástica como la del HoFR. La idea es realizar una actualización del HoF durante el ciclo coevolutivo, cuyo objetivo será eliminar aquellos individuos que no aportan mucha diversidad al conjunto y que tienen mayor cantidad de derrotas con respecto a los demás miembros. De esta manera se va disminuyendo la cantidad de miembros de la memoria y con ello el número de enfrentamientos que se llevan a cabo en el proceso de evaluación. Con esta idea se pretende reducir el tiempo de ejecución del algoritmo, con respecto al del HoFC, pero sin afectar la calidad de las soluciones. Y para ello se garantiza que la transitividad en las estrategias del HoF se logre perfectamente, pues al evaluar cada individuo se tienen en cuenta todos los miembros del salón en el bando opuesto, tal y como hace el HoFC. De esa manera el proceso de aprendizaje no se frustra, porque cada nuevo miembro del elenco pasa por casi las mismas pruebas que sus antecesores y está obligado a superarlos a todos, o la inmensa mayoría de los oponentes ancestrales.

La función de *fitness* que emplea este algoritmo es diferente a la usada en el HoFR y el HoFC. Aquí se combina la puntuación que logra un individuo en las batallas con el

principio del Competitive Fitness Sharing (CFS) (véase la Sección 2.2.2). Con esto se pretende lograr una evaluación más justa de los individuos pues se tendrá en cuenta el desempeño de los otros miembros de la población que permitirá identificar con mayor certeza las estrategias robustas, y con ello ayudar a que la búsqueda se dirija mejor.

El esquema evolutivo que emplea el HoFU es el mismo visto para el HoFR pero agregándole el método de actualización del HoF, y variando el mecanismo inicialización de la población, pues en el HoFU se parte de individuos generados aleatoriamente. Seguidamente se explican los métodos que son particulares de este nuevo algoritmo.

Proceso de actualización del Salón de la Fama

Este método garantiza que en el elenco sólo permanezcan las estrategias más robustas, y preserva la diversidad entre sus miembros. A la hora de determinar cuál de las estrategias del salón será eliminada, se tienen en cuenta dos factores: la cantidad de derrotas y la influencia del individuo en la diversidad del HoF. El primero se refiere al número de derrotas obtenidas por esa estrategia en el paso coevolutivo anterior, y el segundo a la contribución de diversidad que aporta ese individuo al conjunto de soluciones. Durante la actualización, si hay que elegir entre eliminar dos soluciones, se quita aquella que es más similar a las otras estrategias del elenco. Esto conlleva a que sea necesario definir esa medida de diversidad, y la manera en que se usará en relación con la calidad.

Ese valor de *diversidad* que un individuo aporta es calculado a través de la distancia genotípica. Para el cálculo se maneja la memoria de campeones como una matriz en la que cada fila representa una solución y cada columna de un gen (es decir, una acción en la estrategia), de forma que el valor de entropía para una columna j sería:

$$H_j = - \sum_{i=1}^k (p_{ij} \log p_{ij}) \quad (4.2)$$

donde p_{ij} es la probabilidad de la acción i en la columna j , y k indica el tamaño de la memoria. Finalmente, la entropía de todo el conjunto se define como:

$$H = \sum_{j=1}^n H_j \quad (4.3)$$

Cuanto mayor sea el valor de H mayor será la diversidad del conjunto (cuyo tamaño es n). Para determinar la contribución de diversidad que ofrece una solución específica, se calcula el valor de la entropía con esta solución dentro del conjunto, y el valor con ella fuera del conjunto, y la diferencia de estos dos valores representará la contribución de diversidad ese individuo.

Algoritmo 3: Actualización del Salón de la Fama

```

1 for  $i = 0$  to  $i < longitud(bando.salón\_de\_la\_fama)$  do
2   |  $miembro \leftarrow bando.salón\_de\_la\_fama[i]$ ;
3   |  $lista\_derrotas[i] \leftarrow obtener\_derrotas(miembro)$ ;
4   |  $lista\_diversidad[i] \leftarrow obtener\_diversidad(miembro)$ ;
5 end for
6  $lista\_dominadas \leftarrow obtener\_dominadas(lista\_derrotas, lista\_diversidad)$ ;
   | // Obtener el individuo con más derrotas
7  $p \leftarrow obtener\_peor\_calidad()$ ;
8 if  $lista\_dominadas \neq null$  then
9   |  $eliminar\_primer\_elemento(lista\_dominadas)$ ;
10 else
   | // Eliminar el individuo con mayor cantidad de derrotas
11   |  $eliminar(p)$ ;
12 end if

```

Con respecto a la relación de la diversidad y la calidad en el proceso de actualización, se emplea un enfoque ‘multiobjetivo’, donde cada solución tiene una medida de calidad y otra de diversidad (o sea, de influencia en la diversidad). Si para alguna solución S hay otra solución S' que tenga mejor calidad y mayor diversidad se dice que S está dominada. La solución que se elimina es una elegida aleatoriamente de entre las dominadas, y si no hay ninguna dominada se elimina la de peor calidad. La frecuencia

de actualización es un importante parámetro que define cada cuántas coevoluciones será actualizado el elenco, por ejemplo, si decimos que la frecuencia será de tres coevoluciones, significa que el HoF se actualiza siempre que se le agreguen tres nuevos miembros. Esta frecuencia es la que acota la cantidad de veces que se va a eliminar un individuo del salón, y es a la vez lo que influye en la reducción del tiempo de ejecución del algoritmo. Véase el Algoritmo 3 que muestra el funcionamiento general de este método.

Incorporación de la Compartición Competitiva del Fitness (CFS)

Como se había mencionado anteriormente, la función de *fitness* en este algoritmo incorpora el enfoque del CFS. Para lo cual se contabilizan la cantidad de derrotas de los miembros del HoF en cada carrera coevolutiva. Lo cual permitirá identificar cuáles de las estrategias oponentes han sido más difíciles de vencer en cada bando. En el Algoritmo 4 se muestra el pseudocódigo que describe el funcionamiento de este método.

Algoritmo 4: Cálculo del *fitness* de un individuo en el método HoFU

```

1 for oponente = 1 to longitud(bando_enemigo.salon_de_la_fama) do
2   | puntuacion ← simular_batalla(oponente, individuo);
   | actualizar_derrotas(oponente);
   | aplicar_penalizacion(puntuacion, oponente);
3 end for
4 fitness ← calcula_media_aritmetica(puntuacion);
   asignar_fitness(individuo, fitness);

```

La idea principal de la nueva función es que una derrota contra el oponente X tiene más importancia si hay otros individuos que sí lograron derrotar a X . Basándonos en este argumento se define un valor de penalización N para cada individuo i (para $1 \leq i \leq k$) en la población y se calcula de la siguiente manera:

$$N_i = 1 - \frac{\sum_{j=1}^k \frac{v_{ij}}{V(j)}}{k} \quad (4.4)$$

donde $v_{ij} = 1$ si el individuo i de la población logra vencer al campeón j del HoF (con cardinalidad k) y $v_{ij} = 0$ en el caso contrario; y

$$V(j) = \sum_{i=1}^n v_{ij}$$

es el número de individuos de la población que vencieron al oponente j . De esta manera, $N_i \approx 0$ si el candidato i vence a todos los oponentes del HoF y la solución i es además de los pocos en lograrlo; $N_i = 1$ si i no logra vencer a ningún oponente; y $0 < N_i < 1$ dependiendo de cuántas veces i gana y cómo de usual es derrotar a ciertos oponentes a los que se enfrenta. Entonces el *fitness* de un candidato quedaría como:

$$F_i = P_i - \omega N_i \tag{4.5}$$

donde P_i es el resultado obtenido en las batallas según la Ecuación (4.1), y $\omega \in \mathbb{N}$ es un coeficiente usado para escalar N_i con el fin de hacerlo más representativo respecto al valor de P .

Para diseñar esta función de evaluación se tuvieron en cuenta las deficiencias que le fueron identificadas al HoFC. La idea de tener en cuenta el entorno del individuo en el método de evaluación se ha puesto en práctica con la función de *fitness* propuesta. Fijémonos que aquí la puntuación que recibe un individuo cuando le gana a un oponente fácil de vencer, es menor que la asignada cuando vence a un rival más fuerte. De esta manera se puede diferenciar mejor el rendimiento de los individuos de la población, y además se hace posible distinguir a los miembros más robustos del HoF.

Experimentos del HoFU en el juego *Robot Wars*

Varias pruebas fueron realizadas al HoFU para evaluar su desempeño. En lo que sigue se describen y analizan los experimentos. En todos los casos se enfoca una comparación entre los resultados del HoFU y los del HoFC, con el objetivo de comprobar



si el nuevo algoritmo logra mejoras en algún aspecto. Los indicadores a tener en cuenta en las comparaciones son: la duración del ciclo coevolutivo, la calidad de las soluciones, y el tiempo de ejecución. Los parámetros usados en las ejecuciones son los mismos que se vieron anteriormente en la Tabla 4.1.

Resultados de la duración del ciclo coevolutivo

Para este indicador se procede igual a como se hizo en las pruebas del HoFR, recordemos que la información interesante aquí es la cantidad de coevoluciones que se logran en el ciclo. La Figura 4.7 muestra el resultado de esta prueba, véase que en ambos algoritmos el ciclo coevolutivo se comporta de manera similar. Hay algunas iteraciones en las que uno u otro algoritmo se destaca, pero el test de *Wilcoxon* confirmó que las diferencias no son de relevancia estadística, y esto también se observa en el *boxplot* de la figura cuando se solapan los intervalos de confianza de las medianas.

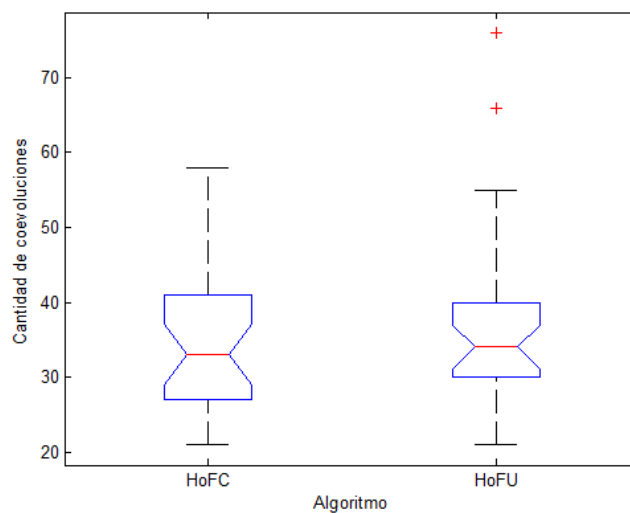


Figura 4.7: Duración del ciclo coevolutivo del HoFC y el HoFU

Recordemos que en el HoFR los ciclos coevolutivos sí se hacían muy largos con respecto al HoFC, esto se debía a que la presión de selección del primero era mucho

menor que la del segundo. En este caso la diferencia entre las presiones selectivas no es tan grande y pudiera ser esta la causa de la similitud en los resultados. Sin embargo, no se debe obviar que en esta prueba las coevoluciones más activas las logró el HoFU, lo cual pudiera ser un indicio de que aquí se centra mejor la búsqueda con el uso de la CFS.

Para analizar la calidad de los individuos y el tiempo de ejecución del algoritmo se realizan dos experimentos diferentes, en ambos se emplea la configuración de los parámetros vista anteriormente. En relación al tiempo medio de las coevoluciones, la prueba se desarrolla igual a como se explicó para el HoFR. Para la evaluación de la calidad se enfrenta uno de los conjuntos soluciones de cada algoritmo contra un oponente fijo que es también un conjunto solución, en este caso específico se produce el enfrentamiento del bando A de ambos algoritmos contra el bando B del HoFC, y luego contra el bando B del HoFU.

Experimento 1 del HoFU en el juego *Robot Wars*

En este experimento se utilizó en la función de *fitness* un valor de $\omega = 20$, recordemos de la Ecuación (4.5) que ω es el factor de escala que se multiplica por la penalización N para hacer significativo el resultado con respecto a la puntuación que obtiene el individuo en las batallas. La frecuencia de actualización empleada fue de cuatro coevoluciones, esto significa que (para ambos bandos) cada cuatro nuevos miembros adicionados al HoF se ejecutará el mecanismo de actualización. En lo que sigue se analizarán los resultados de este experimento.

Experimento 1: resultados de la calidad de las soluciones

La Figura 4.8 muestra los resultados del enfrentamiento de los bandos A (del HoFC y el HoFU) contra el bando B del HoFC. Observemos que HoFU obtiene mejores valores de *fitness*, la diferencia entre las medianas de ambos conjuntos es significativa y

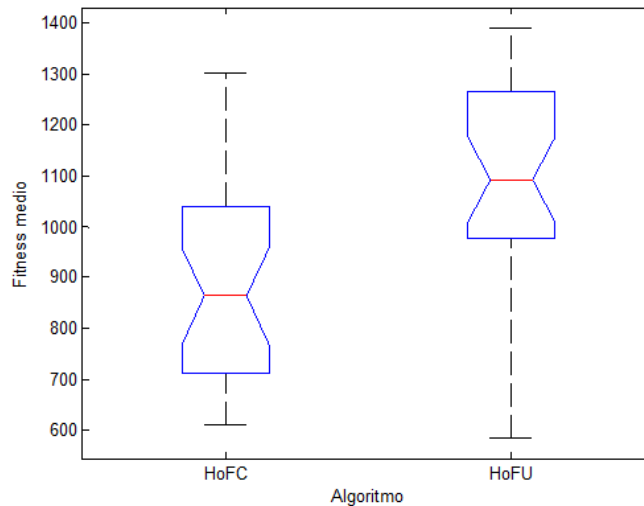


Figura 4.8: Fitness medio del enfrentamiento de los bandos *A* (del HoFC y el HoFU) contra el bando *B* del HoFC

así lo confirmó el test de *Wilcoxon* con un valor de $p=0.000092$. Por tanto, podemos afirmar que en esta prueba el comportamiento del HoFU para las treinta ejecuciones de pruebas fue superior al del HoFC. En la segunda prueba de calidad (Figura 4.9) el rival era el bando *B* del HoFU, aquí se ve que hay una mayor dispersión en el HoFU. Dado que la mediana sigue siendo un poco mejor, y el test de suma de rangos de *Wilcoxon* demostró que la diferencia no es significativa con un valor de $p=0.4733$, podemos afirmar que en promedio dicha dispersión no perjudica, y que HoFU tiene mejor capacidad para encontrar soluciones en el rango alto de calidad (aunque también puede producir soluciones eventualmente de menor calidad). Recordemos que el objetivo de aplicar la coevolución en el juego es encontrar estrategias que le aseguren la victoria a uno u otro bando, y que más tarde serán empleadas por un jugador virtual. Así que si se hacen múltiples ejecuciones de los algoritmos para quedarnos con las mejores soluciones, estaríamos concentrándonos en la parte alta de la distribución y ahí el HoFU es mejor y nos puede proporcionar estrategias más fuertes para conformar el ejército de *bots*.

De lo anterior se puede concluir que la pérdida de calidad que hubo en el HoFR

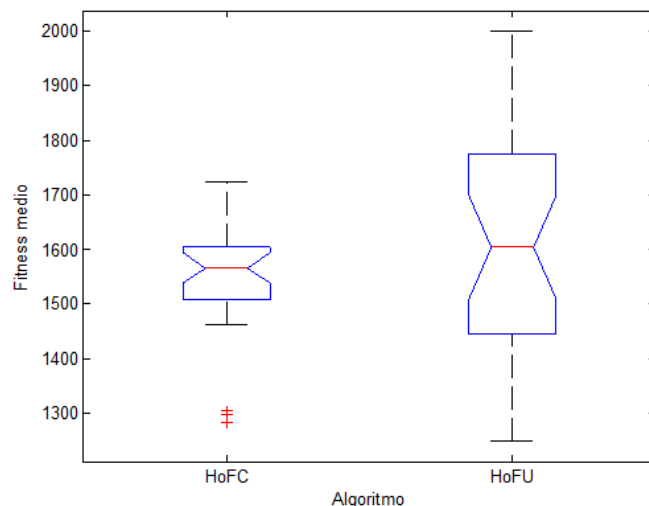


Figura 4.9: Fitness medio del enfrentamiento de los bandos A (del HoFC y el HoFU) contra el bando B del HoFU

aquí no existe. Esto es muestra de que la utilización del método del CFS ayuda a que la búsqueda se enfoque mejor y se obtengan soluciones más potentes. Con este resultado también se comprueba que el hecho de eliminar del Salón de la Fama aquellos individuos que han tenido mayor cantidad de derrotas y no aportan mucha diversidad al conjunto, no afecta la calidad de las soluciones, a pesar de que los individuos en el HoFU se someten a una menor cantidad de enfrentamientos durante la evaluación.

Experimento 1: resultados del tiempo medio de las coevoluciones

La medición del tiempo en este experimento mostró que el HoFU no empeora el tiempo con respecto al HoFC, en la Figura 4.10 se observa que el tiempo medio en las coevoluciones logrado por ambos algoritmos es similar, al aplicar el test de *Wilcoxon* se obtuvo un valor de $p = 0.2870$, con lo cual se demuestra que la diferencia entre los conjuntos de valores no es estadísticamente significativa. Sin embargo, no debemos obviar que el límite superior del *boxplot* para el HoFC es mayor que el del HoFU, recordemos que este último obtuvo en esas ejecuciones soluciones superiores y ahora

vemos que lo hizo sin disminuir el rendimiento. Hasta aquí se puede afirmar que en este

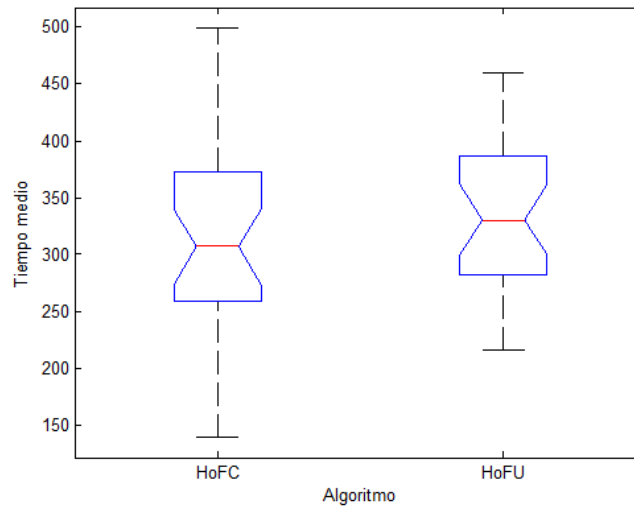


Figura 4.10: Tiempo medio (en segundos) que tardaron las coevoluciones del Experimento 1 en el juego *Robot Wars*

experimento el HoFU fue capaz de lograr mejores valores de *fitness* que el HoFC, sin afectar el tiempo de ejecución del algoritmo. Por tanto, la inclusión de la CFS ayudó a que la búsqueda de las estrategias se enfocara mejor en encontrar soluciones robustas; y por otra parte la eliminación de aquellos miembros del Salón de la Fama que no tenían mucho que aportar al proceso de aprendizaje ni a la diversidad del conjunto, evitó que el aumento de la presión de selección por la variación en la fórmula del *fitness* incrementara el tiempo de ejecución del algoritmo.

Experimento 2 del HoFU en el juego *Robot Wars*

El segundo experimento utiliza una frecuencia de actualización de tres coevoluciones, de manera que cada tres nuevos miembros incluidos al HoF se elimina de él el que tenga peor calidad y aporte menor diversidad al conjunto. Esto hace que en este experimento se quiten más miembros en la memoria de estrategias victoriosas que en el Experimento 1 (donde la frecuencia era de cuatro coevoluciones). También hay diferencia en la fun-

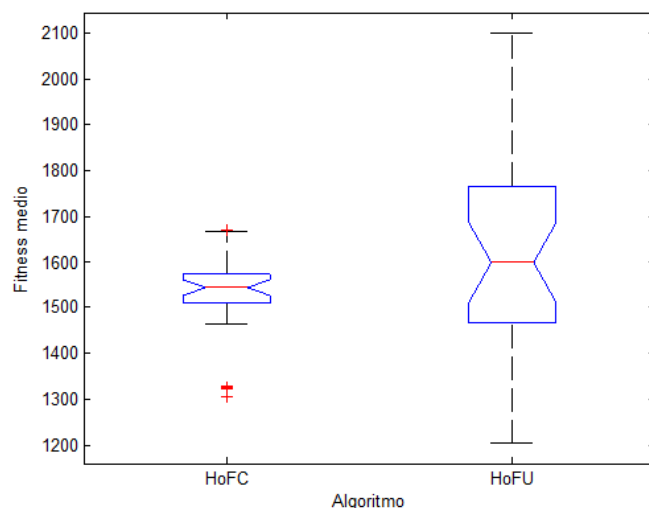


Figura 4.11: Fitness medio del enfrentamiento del conjunto A del HoFC y HoFU contra el conjunto B del HoFC

ción de *fitness*, en este caso se emplea un valor de $\omega = 10$ haciendo que la penalización N sea menos significativa que en el experimento anterior, con lo cual se disminuye un poco la presión de selección durante el proceso de búsqueda.

Experimento 2: resultados de la calidad de las soluciones

Las Figuras 4.11 y 4.12 muestran el comportamiento de ambos algoritmos en dos pruebas de evaluación de la calidad realizadas. En la primera se enfrentaron los bandos A contra el bando B del HoFC y en la segunda lo hicieron contra el bando B del HoFU.

Se puede apreciar en las figuras que la calidad media obtenida en los enfrentamientos para ambos algoritmos es similar, véase que hay solapamiento vertical entre las muescas de los *boxplots* lo que indica que las diferencias respecto a las medianas no son significativas, y esto también lo confirmó el test de *Wilcoxon* con valores de $p = 0,1120$ y $p = 0,5862$ respectivamente. De modo que aquí no hubo mejoras en relación a la calidad, pero tampoco se empeoró. El hecho de que se haya disminuido el coeficiente ω en la función del *fitness* hace que no sea tan notable la penalización N en la Ecuación (4.5)

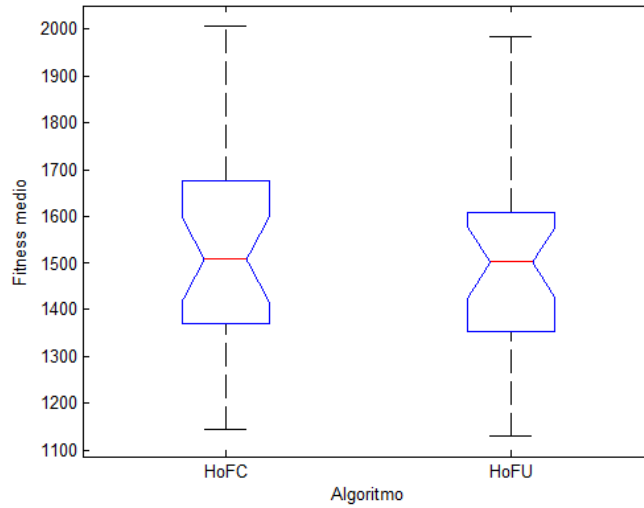


Figura 4.12: Fitness medio del enfrentamiento del conjunto A del HoFC y HoFU contra el conjunto B del HoFU

y esto impide aprovechar la ventaja de la CFS; que como se percibió en el experimento anterior, ayuda a enfocar la búsqueda hacia la estrategias verdaderamente robustas.

Experimento 2: resultados del tiempo medio de las coevoluciones

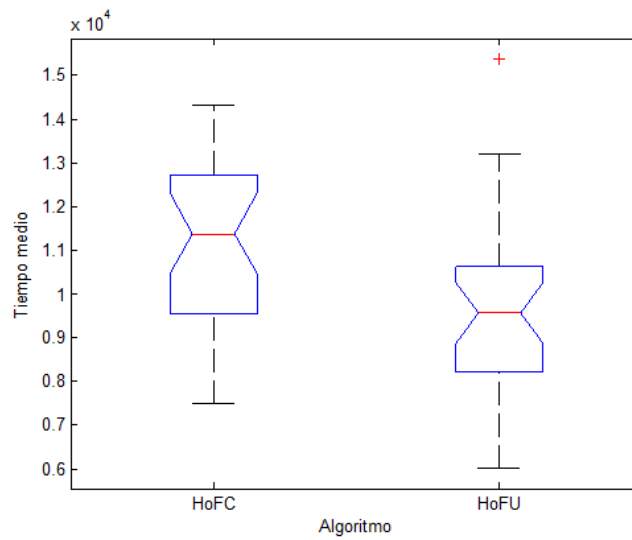


Figura 4.13: Tiempo medio de las coevoluciones en el Experimento 2

La prueba de análisis del tiempo de ejecución arrojó los resultados que muestra la Figura 4.13, nótese que en la mayoría de las ejecuciones el tiempo medio que tardan las coevoluciones en el HoFU es menor que el del HoFC. Aquí las muescas que definen el intervalo de confianza de la mediana para ambos *boxplots* no se solapan, así que las diferencias son válidas y el test de *Wilcoxon* lo comprobó con $p = 0.0095$. El valor atípico que se observa para el HoFU se corresponde con una ejecución que alcanzó 76 coevoluciones y esto hizo que se lograra una carrera evolutiva de mucha calidad donde la presión de selección aumentaba gradualmente y extendía las coevoluciones. Con estos resultados se puede afirmar que en este experimento el HoFU obtuvo mejor rendimiento que el HoFC, pues logró un menor tiempo de ejecución en sus coevoluciones sin afectar la calidad de las soluciones. Esto se debe a que se actualiza más frecuentemente el Salón de la Fama lo cual hace que se eliminen una mayor cantidad de miembros del elenco y con ello disminuya el número de enfrentamientos realizados durante la evaluación.

A modo de conclusiones se puede decir que el algoritmo HoFU fue un segundo intento por superar el desempeño del HoFC y esta vez los resultados obtenidos fueron alentadores. La incorporación de la CFS ayudó a centrar la búsqueda hacia las estrategias más destacadas de la población. Esta ventaja permitió que en uno de los experimentos el valor de la calidad de las soluciones del HoFU fuese mayor que la del HoFC sin aumentar el tiempo de ejecución del algoritmo con respecto al del HoFC. En un segundo experimento se obtuvo un mejor rendimiento del HoFU y a la vez se lograron soluciones de calidad similar a las del HoFC. Este último resultado fue debido a que disminuyó más el número de enfrentamientos durante la evaluación de los individuos, y por otra parte se hizo menos significativo el aporte de la CFS con el decremento del coeficiente ω en la fórmula del *fitness*. La conclusión principal que podemos extraer es que mediante el ajuste de los parámetros (frecuencia de actualización y coeficiente de penalización) el algoritmo HoFC se puede configurar para lograr mejores soluciones o menores tiempos de ejecución.

4.2. Otros enfoques de optimización del HoF y su aplicación al *Planet Wars*

En esta segunda parte del capítulo, se profundiza en el análisis del enfoque optimizado del HoF que como se comprobó en la sección anterior contribuye a que aumente la eficacia del proceso de búsqueda. En esta sección se continuará apostando por mantener la transitividad entre las soluciones encontradas pues se comprobó anteriormente que el mecanismo de memoria contribuye a no romper esta relación entre los individuos de diferentes generaciones y es un punto a favor para combatir los ciclos que se pueden generar en el modelo. A continuación se explicarán los enfoques que serán probados y se analizarán sus resultados usando como plataforma de pruebas otro juego distinto al anterior, esta vez se trata del *Planet Wars* (presentado en la Sección 3.5.2).

4.2.1. Esquema general del enfoque HoFCC

El Algoritmo 5 muestra el esquema general del método HoFCC (del inglés: *Hall of Fame Competitive Coevolutionary algorithm*) que es el algoritmo básico con el que se va a experimentar en esta sección. Este algoritmo se basa en la idea de optimizar el salón de la fama y con él se profundiza en el estudio de los indicadores que determinan cuáles campeones permanecen y cuáles no.

Un cambio relevante del enfoque HoFCC es que se van a integrar individuos que son completamente externos al proceso de aprendizaje de la población, es decir, no comparten ningún vínculo genético, pues fueron estrategias creadas manualmente. De esta forma se fomenta la distinción entre el conjunto de entrenamiento y el de evaluación de los individuos, lo cual favorece al proceso de optimización y búsqueda de *bots* competitivos que se autoadaptan para vencer a sus enemigos (co)evolucionados.

La base del ciclo coevolutivo del HoFCC es un algoritmo genético de estado estacionario (Líneas 11-17). Los operadores genéticos que se aplican serán definidos en la

Algoritmo 5: Algoritmo HoFCC(α, λ)

```

1  $nCoev \leftarrow 0$ ;  $A \leftarrow jugador_1$ ;  $B \leftarrow jugador_2$ ;  $\phi \leftarrow umbralIndividuoVictorioso$ ;
2  $HoF_A \leftarrow \emptyset$ ;  $HoF_B \leftarrow OponenteInicial()$ ;  $HoC \leftarrow Botsexpertos()$ ;
3  $pop \leftarrow EVALUAR(HoF_B)$ ; // Se evalúa la población inicial
4 while  $nCoev < maxCoevFallidas \wedge NOT(timeout)$  do
5   if  $nCoev \bmod \lambda = 0$  then
6      $PURGE_\alpha(HoF_A)$ ; // Se actualiza el HoF cada  $\lambda$  coevoluciones
7      $pop \leftarrow GENERACIONALEATORIA()$ ; // Se inicializa aleatoriamente la
        población
8      $i \leftarrow 0$ ;
9      $ganadorEncontrado \leftarrow falso$ ;
        // Se escoge un experto aleatoriamente
10     $e \leftarrow OBTENEREXPERTO(HoC)$ ;
11    while ( $i < maxGeneraciones$ )  $\wedge \neg encuentraGanador$  do
12       $progenitores \leftarrow SELECCIONAR(pop)$ ;
13       $hijos' \leftarrow CRUZAR(progenitores, p_X)$ ;
14       $hijos'' \leftarrow MUTAR(hijos', p_M)$ ;
15       $pop' \leftarrow REEMPLAZAR(pop, hijos'')$ ;
16       $pop \leftarrow pop'$ ;
17       $pop \leftarrow EVALUAR(HoF_B)$ ;
18      if ( $fitness(mejor(pop)) \geq \phi$ )  $\wedge$  ( $VencerExperto(mejor(pop), e)$ ) then
        //Ganador encontrado!
19         $ganadorEncontrado \leftarrow true$ ;
20         $HoF_A \leftarrow HoF_A \cup \{mejor(pop)\}$ ;
21         $temp \leftarrow A$ ;  $A \leftarrow B$ ;  $B \leftarrow temp$ ; // Intercambio de los roles de
        las poblaciones
22      else
23         $e \leftarrow OBTENEREXPERTO(HoC)$ ;
24         $i \leftarrow i + 1$ ;
25      end if
26    end while
27    if  $ganadorEncontrado$  then
28       $nCoev \leftarrow 0$ ; // Se inicia una nueva búsqueda
29    else
30       $nCoev \leftarrow nCoev + 1$ ; // Se continúa la búsqueda
31    end if
32 end while

```

sección de experimentos. A continuación se explican algunas de las variables fundamentales del algoritmo:

- *maxCoevFallidas* representa el límite de coevoluciones continuas que se pueden ejecutar sin encontrar una solución de victoriosa.
- *maxCoevoluciones* indica el número máximo de coevoluciones totales que puede realizar el algoritmo.
- *maxEvaluaciones* es el límite de las evaluaciones que puede realizar el algoritmo.
- *numEvaluaciones* es el número de evaluaciones que se han realizado hasta ese momento.
- La condición de parada (Línea 4) está asociada a:
$$timeout = (nCoev > maxCoevoluciones) \vee (numEvaluaciones > maxEvaluaciones)$$

En este algoritmo se incorpora un nuevo conjunto a tener en cuenta durante la evaluación de los individuos, se trata del *HoC*. Es una memoria de “expertos”, es decir, *bots* especializados, que no guardan ninguna relación genética con los miembros de la población evaluada, pues han sido definidos manualmente. Con este *HoC* se pretende aumentar la eficacia del proceso evaluativo, diversificando los oponentes que tiene que vencer un individuo para coronarse campeón.

Respecto a la evaluación de los candidatos, para una población específica p (donde $p \in \{\text{jugador 1, jugador 2}\}$), el *fitness* de un individuo es calculado mediante un conjunto de enfrentamientos que tienen lugar entre éste y cada miembro de un subconjunto del HoF del oponente; y además una batalla contra un *bot* experto, escogido al azar del conjunto de expertos nombrado en el Algoritmo 5 como HoC (del inglés *hall-of-celebrities*). El cálculo matemático usado para evaluar el rendimiento de un individuo s plantea que:

$$fitness(s) = \frac{\sum_{j=1}^k (r_j^s - nTurn_s(j))}{k} + \beta_s(e) \quad (4.6)$$

donde $k \in \mathbb{N}$ es la cardinalidad del subconjunto de oponentes (o sea, del HoF del oponente), $r_j^s \in \mathbb{R}$ devuelve ϕ puntos si la estrategia s derrota a h_j del HoF del oponente (i.e. caso victorioso), y 0 si h_j gana a la estrategia s ; $nTurn_{sj} \in \mathbb{N}$ indica el número de turnos consumidos; y $\beta_s(e) \in \mathbb{N}$ es un bono (que lo hemos denominado como β) de puntos extras que s obtiene si logra vencer al “bot experto” e .

Basándonos en este esquema general mostrado en el Algoritmo 5 se han diseñado otras tres variantes del HoFCC que pretenden optimizar el desempeño de este método básico y de los que se presentaron en la sección anterior. En lo que sigue de explican estas propuestas.

4.2.2. Optimización del HoF basado en la diversidad (HoFCC-Diversity)

En esta, como en todas las propuestas que se harán en el capítulo, el HoF actúa como un mecanismo de memoria a largo plazo. Pero esa memoria será optimizada mediante diversos mecanismos que son en esencia los que van a distinguir las tres variantes que se explicarán aquí. Para este primer caso, ese proceso de actualización del conjunto de campeones se hará eliminando a los miembros que proporcionan menos diversidad al elenco. Ese valor de *diversidad* que un individuo aporta es calculado siguiendo el mismo enfoque explicado en la sección anterior para el método HoFU con las Ecuaciones (4.2 y 4.3).

El número de individuos que se eliminarán de la memoria debe ser definido por el programador como un valor porcentual (α) que representa la porción del HoF que se va a retirar; en otras palabras, el HoF (con cardinalidad $\#HoF$) está ordenado de forma decreciente según el aporte de diversidad de cada miembro y los últimos $\lceil \frac{\#HoF \times n}{\alpha} \rceil$ individuos de esa conjunto ordenado se eliminan. Este proceso de actualización del HoF también está marcado por un parámetro que llamamos “frecuencia de actualización”

(λ), por ejemplo, un valor de $\lambda = 3$ indica que el HoF será actualizado cada tres turnos coevolutivos.

La motivación principal de esta propuesta es la apuesta por mantener diversidad entre los miembros de la HoF, y al mismo tiempo reducir (o mantener un valor aceptable para) el tamaño de la memoria. Con esa idea, se asume que el hecho de eliminar individuos no afectarán la calidad de las soluciones finales que encuentre el algoritmo, pues en el proceso de búsqueda se fomenta a que permanezcan en la memoria las estrategias más valiosas que puedan aportar beneficios a las generaciones futuras y que sean las más integrales en cuanto a calidad y diversidad.

4.2.3. Optimización del HoF basado en la calidad de las soluciones (HoFCC-Quality)

En esta versión, seguimos un enfoque similar al aplicado en HoFCC-Diversity pero ahora el HoF se ordena con respecto a una medida de calidad que se define como el número de derrotas que un individuo obtuvo en la última coevolución. En el algoritmo esta métrica se controla mediante un variable contador asociada a cada uno de los miembros del HoF y se encarga de controlar la cantidad de derrotas que obtiene ese individuo durante la fase evaluativa de la población oponente.

Basándonos en nuestra experiencia de juego hemos asumido que esta métrica representa la robustez de un individuo, y se ha usado con el objetivo de potenciar que en el HoF se mantengan sólo las soluciones más adaptadas y potentes. Para lograr esto se lleva a cabo el proceso de purga de la memoria de campeones, esta vez centrado en eliminar las soluciones más débiles. De la misma manera que en el HoFCC-Diversity, los parámetros λ y α deben ser configurados para definir la frecuencia y el alcance del método de purga sobre el conjunto HoF.

4.2.4. Optimización del HoF con enfoque multiobjetivo (HoFCC-U)

Siguiendo la línea de optimizar la memoria de campeones, esta variante al igual que con el HoFU de la sección anterior, propone un enfoque multiobjetivo donde cada solución tiene asociado un valor de diversidad y uno de calidad, basados en los indicadores que se han explicado en los enfoques anteriores. Partiendo de esto, se realiza la purga de un porcentaje (α) del conjunto de soluciones dominadas (de acuerdo a indicadores multiobjetivos). En caso de que el subconjunto de soluciones dominadas esté vacío, entonces el HoF se ordena de acuerdo con la medida de calidad y las soluciones con el valor más bajo serán las eliminadas.

Al igual que en los algoritmos anteriores (HofCC-Quality y HofCC-Diversity) la frecuencia de actualización del HoF es un parámetro importante que debe ser definido. Esta propuesta utiliza la función de aptitud presentada en la Ecuación (4.5)) recordemos que su idea principal es que una derrota contra el oponente X tiene más importancia si hay otros individuos que sí lograron derrotar a X y siguiendo ese principio, se define una penalización N que se aplica a la hora de evaluar a los individuos. Véase en la Tabla 4.2 el valor que se usa para el coeficiente ω de la Ecuación (4.5).

4.2.5. Experimentos y resultados

Esta sección describe el análisis experimental realizado para las tres variantes de la HoFCC descritas en la sección anterior. Todos los experimentos fueron ejecutados usando el juego *Planet Wars* (ver Sección 3.5.2). Se consideraron cinco mapas, todos pertenecen a la colección de mapas que fueron diseñados en el marco del *Google AI Challenge 2010* (específicamente, mapa 1, mapa 10, mapa 20, mapa 30, mapa 50). Para evaluar el *fitness* de un individuo, éste tendrá que enfrentarse a sus oponentes en tres batallas consecutivas en cualquiera de estos mapas. Si el individuo derrota al oponente en todas las batallas entonces será considerado el ganador del enfrentamiento.

Para inicializar el HoF se emplea una estrategia aleatoria (es decir, creamos una

Tabla 4.2: Parámetros del ciclo coevolutivo para el HoFCC

Parámetro	Valor
$maxCoevFallidas$	5
$maxCoevoluciones$	15
$maxGeneraciones$	20
$tamañoPoblación$	15
$maxEvaluaciones$	10000
p_X	0,75
p_M	$1/longitudCromosoma$
ϕ	1500
β	500
ω	50
λ	3

matriz de acción aleatoria), y el conjunto de los expertos (el HoC) estará compuesto por todos los *bots* predefinidos que se ofrecían en el concurso Google AI Challenge 2010, estos son: BullyBot, RandomBot, DualBot, ProspectorBot y RageBot; y además se ha incluido el GeneBot, que fue diseñado y posteriormente optimizado a través de algoritmos evolutivos para participar en la competición, y que logró posicionarse dentro del 20% (o sea, el top-20%). Seleccionamos este *bot* porque es un oponente muy competente, le aporta diversidad al HoC, su código fuente estaba disponible y ha sido ampliamente descrito en la literatura científica (Fernández-Ares et al., 2011; Mora et al., 2012).

A continuación, se explica la configuración de estos experimentos, y posteriormente se discutirán los resultados obtenidos.

4.2.6. Configuración de los experimentos

Nueve instancias de cada algoritmo fueron utilizados, tres para cada uno de los enfoques: HoFCC-Diversity, HoFCC-Quality y HoFCC-U, en ellas se varía los valores de $\alpha \in \{10\%, 30\%, 50\%\}$. La notación $HoFX\alpha$ –donde $X \in \{Div, Qua, U\}$ – se utiliza para denotar cada una de estas nueve variantes. En todos los casos, establecemos $\lambda = 3$

y realizamos diez ejecuciones por cada instancia. La tabla 4.2 muestra los valores de parametrización del ciclo coevolutivo. La base de este ciclo es un algoritmo genético de estado estacionario (GA) –note que esto corresponde a las líneas 11-17 en el Algoritmo 5– En él se emplea torneo binario para la selección, cruce uniforme como método de recombinación genética, una mutación de tipo *Bit-flip* y un mecanismo de reemplazo elitista.

El análisis de los resultados expuestos en esta sección ha sido guiado por los indicadores que se explican a continuación:

- Mejor *fitness*: indica el valor más alto de *fitness* encontrado por el algoritmo.
- Fitness medio: es el valor de aptitud promedio alcanzado durante los ciclos coevolutivos.
- Número de evaluaciones: indica el número total de batallas que fueron ejecutadas durante el proceso de evaluación.
- Número de victorias: indica el número total de victorias obtenidas en un combate de tipo *Todos vs. Todos* que enfrenta a las mejores soluciones encontradas por cada una de las versiones del HoFCC.

A continuación se analizan los resultados obtenidos en diez ejecuciones de cada una las nueve versiones del HoFCC empleando los indicadores antes mencionados. El análisis estadístico de los datos se hace mediante un test no paramétrico, el *Kruskal-Wallis* (Kruskal y Wallis, 1952), con un intervalo de confianza del 95%. Cuando este test detecta diferencias estadísticamente significativas en las distribuciones se aplica el test de *Dunn–Sidak* (Sokal y Rohlf, 1995) para determinar cuáles pares de medianas son significativamente diferentes y cuáles no.

4.2.7. Análisis de los resultados

La Figura 4.14 muestra los resultados del *fitness* de la mejor estrategia encontrada por cada una de las instancias del algoritmo. La prueba de Kruskal-Wallis muestra que no hay diferencias significativas entre los valores (véase la primera fila en la tabla 4.3). Obsérvese la existencia de tres valores atípicos (con valor de aptitud 0) en los casos HoFU50, HoFU30 y HoFDiv30; esto indica que en algunas ejecuciones no se encontró ninguna estrategia ganadora después de completar el número máximo de coevoluciones fallidas.

Se puede ver en la Figura 4.15 el comportamiento del *fitness* promedio para cada instancia de algoritmo. Luego de analizar los resultados con el test de Kruskal-Wallis se confirma que las diferencias entre los valores son estadísticamente significativas (ver segunda fila en la tabla 4.3). Según las conclusiones del método Dunn–Sidak, los valores extremos de la distribución son los más distintivos: HoFDiv50 tiene diferencias significativas con respecto a HoFDiv30, HoFU10, HoFU50 y HoFU30; por su parte, el HoFU30 las tiene con HoFQua10, HoFQua50, HoFDiv10 y HoFDiv50. Note que los algoritmos que se enfocan en la diversidad obtienen los mejores resultados.

Observe también que HoFDiv50 logra una vez más el mejor rendimiento. Por otra parte, las versiones híbridas del algoritmo no parecen ser muy competitivas, esto podría deberse a que es más difícil obtener un alto valor de aptitud debido a la penalización que se aplica a la puntuación obtenida por los individuos de acuerdo con la Ecuación (4.4). Por tanto, esta disminución del valor objetivo del *fitness*, no tiene por qué indicar que se ha afectado la calidad del individuo (esto se verificará posteriormente en otra prueba).

En el caso del número de evaluaciones (ver Figura 4.16), hay diferencias significativas (véase tercera fila en la Tabla 4.3) entre HoFDiv30 y HoFQua30. Observamos, sin embargo, que el número de evaluaciones es directamente proporcional a la duración del ciclo coevolutivo. Téngase en cuenta de que para estos experimentos fijamos una cuota

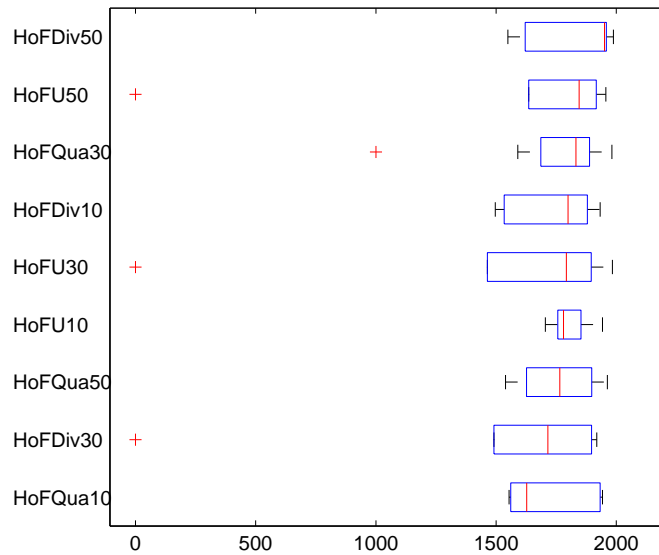


Figura 4.14: Distribución de los mejores valores de *fitness* logrados por cada algoritmo. Como es usual, cada caja representa el segundo y el tercer cuartil de la distribución, la mediana está marcada con una línea vertical y los valores atípicos se indican con un signo de ‘+’.

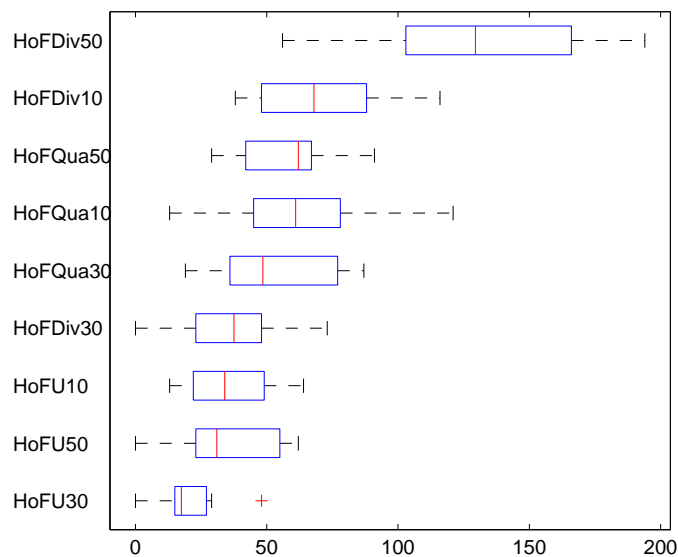


Figura 4.15: Distribución del *fitness* medio obtenido en cada algoritmo.

Tabla 4.3: Resultados del test de Kruskal-Wallis para todos los indicadores.

Indicador	p -value
Mejor <i>fitness</i>	0,4605
<i>Fitness</i> medio	1,356e-007
Número de evaluaciones	0,0266
Número de victorias	0,0093

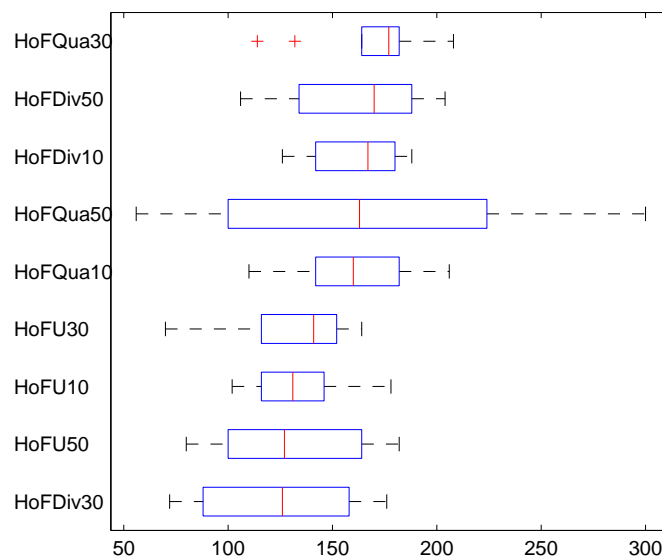


Figura 4.16: Distribución del número de evaluaciones empleadas por cada algoritmo.

máxima de cinco coevoluciones consecutivas sin éxito, y cuando se alcanza este límite el algoritmo se detiene. Se observó que los ciclos coevolutivos no eran demasiado largos, esto podría ser una consecuencia directa de la especialización durante la búsqueda lo que hace más difícil el logro de mejores soluciones. Al completar los experimentos, se cuenta con diez campeones (que serán los mejores individuos almacenados en el HoF) por cada instancia de algoritmo (es decir, uno por cada ejecución), el objetivo es hacer un torneo de tipo *Todos vs. Todos* que nos permita hacer un análisis de la calidad de los individuos. De esta forma cada individuo se enfrenta en tres batallas (cada una de ellas en un escenario distinto) contra los diez campeones de los otros algoritmos.

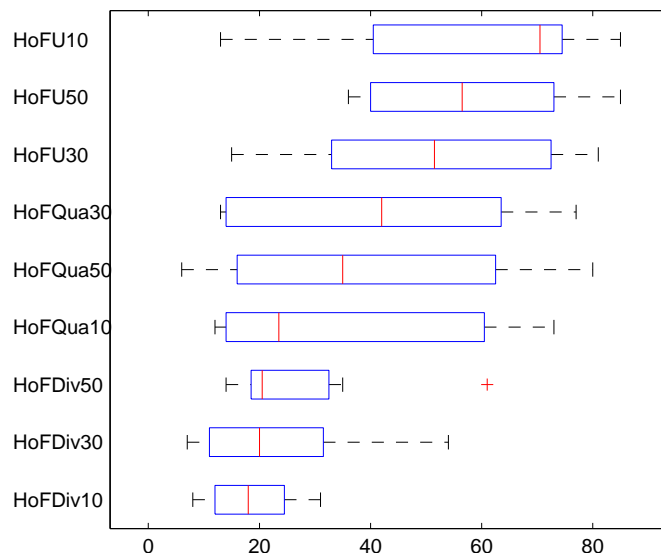


Figura 4.17: Distribución del número de victorias obtenido por cada algoritmo en el torneo de *Todos vs. Todos*

La figura 4.17 muestra los resultados de este torneo. Obsérvese cómo cada familia de algoritmos se distingue claramente de los demás (esto no pudo percibirse en nuestro trabajo anterior sobre el juego *Robot Wars*). Según la prueba de Kruskal-Wallis hay diferencias significativas entre los valores (véase la cuarta fila en la Tabla 4.3). Las principales diferencias son entre HoFDiv y HoFU. Todas las instancias del HoFU obtienen los mejores resultados, lo que significa que los algoritmos basados en el enfoque multiobjetivo son los más eficaces, mientras que las versiones basadas en la métrica de diversidad mostraron un desempeño más pobre en los combates directos. Ciertamente, esto parece contradecir los resultados que se muestran en las Figuras 4.14 y 4.15 (ver discusión sobre esto en la subsección siguiente) aunque para nosotros siempre fue un resultado esperado que confirma que los jugadores virtuales más fuertes y más consistentes son los generados a partir del uso combinado de las dos métricas.

Observamos que en la prueba de *Todos vs. Todos*, cuyo objetivo fue comprobar la robustez de los “campeones”, los resultados de los algoritmos no están en sintonía con

CAPÍTULO 4. GENERACIÓN DE ESTRATEGIAS DE JUEGOS MEDIANTE COEVOLUCIÓN

los otros indicadores que se centran en el análisis de los valores de *fitness*. Esto puede ser un signo de que el proceso coevolutivo se ve afectado por la aparición de ciclos y esto hace que una solución con una alta aptitud podría ser teóricamente considerada como más fuerte y luego en la práctica no lo será, de hecho la puntuación de la aptitud no está directamente relacionada con el concepto de fuerza (o eficiencia en combate). Los jugadores virtuales más fuertes (según el torneo final) fueron los obtenidos por la familia de algoritmos que utilizan el enfoque multiobjetivo basado en valores de diversidad y calidad como métrica para guiar la búsqueda mientras que los algoritmos basados en una sola métrica (especialmente aquellos que consideraban los valores de diversidad) no mostraron buenos resultados.

Los valores bajos que mostraron los *bots* (en las gráficas de comportamiento del *fitness*) generados de los algoritmos HoFU pueden haber sido causados por la penalización que se aplica a la puntuación en la función de aptitud según la Ecuación (4.4). Los resultados obtenidos en este capítulo proporcionan evidencia concluyente a favor de las hipótesis establecidas en los capítulos previos sobre el juego *Robot Wars*. Y el hecho de haber incluido más batallas en este último experimento durante la fase de evaluación de los individuos, unido a la aplicación del concepto de CFS a la función de *fitness*, y el aumento de la presión en la búsqueda a través de la inclusión de un archivo de expertos causó una reducción en la aparición de ciclos. En los experimentos anteriores sobre el juego *Robot Wars* hemos asumido que se mantenía la transitividad entre las soluciones del HoF y por tanto, la última solución agregada al HoF sería la más fuerte y mejor adaptada; y las soluciones se evaluaban sólo teniendo en cuenta sus resultados al enfrentar a los miembros del HoF del oponente y esto muy probablemente haya favorecido la aparición de ciclos pues no había ningún elemento externo en el conjunto de evaluación. Basándonos en esto, para este segundo experimento se propuso usar el *HoC*, con el objetivo de incluir en el conjunto de evaluación agentes externos que no tengan relaciones evolutivas con los individuos evaluados lo cual previene la aparición

de ciclos.

Respecto al porcentaje de actualización del HoF (es decir, 10 %, 30 %, o 50 %), no se observaron diferencias significativas entre las variantes de los algoritmos. En este caso, creemos que se debe al hecho de que el HoF no alcanza un tamaño significativo porque en los experimentos se establece un límite de coevoluciones totales, por lo que el HoF no puede crecer significativamente y esto no permite explorar ampliamente los matices que podrían generarse dentro de la configuración del proceso de actualización del HoF.

4.3. Conclusiones del capítulo

Encontrar algoritmos que logren reducir el efecto de las patologías inherentes a los modelos coevolutivos es una línea de investigación abierta actualmente. En este capítulo se exploró el uso del método de archivo HoF para conservar a los campeones encontrados en cada paso coevolutivo y utilizarlos en el proceso de evaluación que guía la búsqueda. El estudio empírico realizado sobre un juego *Robot Wars* -que tenía limitaciones intrínsecas debido a su simplicidad- demostró que el conocido problema de los ciclos siguió apareciendo en el proceso coevolutivo debido a la falsa creencia de una relación de transitividad entre los campeones de la memoria en el enfoque HoFR. Por otra parte, el HoFU propuso un mecanismo de actualización del elenco de campeones que tiene como objetivo eliminar a los individuos que contribuyen menos al proceso de optimización. Una nueva función de aptitud, inspirada en el Competitive Fitness Sharing, también se usó en esta versión multiobjetivo y demostró ser de gran ayuda en el enfoque de los objetivos de búsqueda.

En la segunda parte del capítulo se empleó el juego *Planet Wars* con el ánimo de extender el trabajo previo de experimentación. Se introdujeron cambios en el proceso coevolutivo, variando el método de evaluación para identificar mejor aquellas soluciones que son realmente fuertes. El nuevo enfoque HoFCC mantiene el uso del HoF,

pero incorpora una memoria adicional (el *HoC*) que contiene otros *bots* especializados (es decir, *bots* expertos, o lo que es lo mismo, jugadores virtuales “experimentados” u optimizados, bien automáticamente o manualmente) que se utilizan para evaluar a los individuos de la población. El concepto de HoC permitió aumentar la presión selectiva en la búsqueda coevolutiva, pues los *bots* expertos se supone que han sido especializados en otros contextos independientes (por ejemplo, mediante otras técnicas de optimización o predefinidos en base a conocimientos de jugadores humanos expertos) al del individuo evaluado, además no tienen ninguna relación evolutiva con la población.

El juego *Planet Wars* permitió una experimentación más profunda y por lo tanto el logro de resultados más consistentes. Los resultados obtenidos en los experimentos distinguen claramente el rendimiento de cada familia de algoritmos en la búsqueda de soluciones fuertes. En este sentido, nuestra versión HoFCC multiobjetivo mostró un rendimiento consistente. Esta variante multiobjetivo se guiaba por la calidad del candidato a la solución (con respecto a HoF y HoC) y la diversidad del Salón de la Fama. Los algoritmos que usaban las métricas de calidad y diversidad de forma independiente mostraron una marcada diferencia pues los que se enfocaban en el indicador de calidad lograron encontrar individuos más competitivos que las variantes guiadas por la métrica de diversidad.

El trabajo futuro consistirá en analizar el desempeño de nuevos modelos coevolutivos y la aplicación de los descritos aquí en otros juegos RTS, incorporando nuevas métricas para el proceso de actualización de HoF, tratando de diseñar nuevos mecanismos de evaluación con el objetivo de reducir los efectos de patologías coevolutivas y explotando el potencial de los métodos de archivos al combinarlos con otros enfoques que contribuyen a la mejora de la calidad de las soluciones.

Capítulo 5

Generación simultánea de contenidos y estrategias para juegos usando Coevolución

*If you want something new,
you have to stop doing something old.*

Peter F. Drucker

Los diseñadores de juegos reconocen la IA como una de las principales vías para convertir buenos juegos en best-sellers. La aplicación de la IA, y la Inteligencia Computacional (IC) (como la representante de la IA que estudia las técnicas computacionales inspiradas en la naturaleza para el aprendizaje y la optimización, tales como los algoritmos evolutivos, redes neuronales artificiales, la inteligencia de enjambre y la lógica difusa, por poner algunos ejemplos) en juegos es un campo de investigación que plantea retos importantes para la comunidad científica (Lara-Cabrera et al., 2013c;

CAPÍTULO 5. GENERACIÓN SIMULTÁNEA DE CONTENIDOS Y ESTRATEGIAS PARA JUEGOS USANDO COEVOLUCIÓN

Lara-Cabrera et al., 2014; Togelius et al., 2013). No obstante, la aplicación más tradicional de la IA en los juegos se limitaba a la búsqueda de mecanismos para gobernar el comportamiento de los jugadores no humanos durante la partida, con el objetivo de dotar de inteligencia al enemigo y, en consecuencia, aumentar la satisfacción del jugador humano quien por su parte exige un rival que exhiba un comportamiento inteligente. Sin embargo, el estado actual del quehacer investigador requiere la revisión de la vigencia del término de IA para juegos y debería ser enfocado hacia “la mejora de las áreas de investigación y desarrollo no tradicionales que van más allá del control de los NPCs (Yannakakis, 2012) como la IA/IC que se aplican a la mayoría de los aspectos del diseño y desarrollo de juegos” (Lucas, 2009).

Como se vio en capítulos anteriores, la PCG consiste en la aplicación de técnicas de IA para generar automáticamente el contenido del juego y representa uno de los temas más interesantes de la comunidad de la IC aplicada a videojuegos. Ya que el contenido del juego es uno de los factores más importantes para mantener a los jugadores “enganchados al mundo de los juegos” (Hendrikx et al., 2013).

Por otra parte, como se ha explicado en el Capítulo 2, la *Coevolución* apuesta por la interacción entre varios individuos que pueden ser de una única o de diferentes especies. Su aplicación en el área de los videojuegos ha sido bien recibida y ha abierto un nuevo campo de estudio. Varios modelos han tenido éxito en distintos juegos, pero también han aflorado algunos inconvenientes a causa de las patologías intrínsecas de los modelos coevolutivos que son difíciles de resolver.

Parte del trabajo de esta tesis ha estado enfocado en la creación de nuevos algoritmos que permitan mejorar el desempeño de los modelos CC. En el capítulo anterior se explicaba el análisis realizado sobre el uso de los métodos de archivos para la generación de jugadores virtuales en los juegos *Robot Wars* y el *Planet Wars*. Se comprobaba que la diversidad y el crecimiento del Hall of Fame (HoF) pueden influir en la calidad de las soluciones obtenidas, además de que la implicación de nuevos indicadores de calidad en

la función de aptitud ayudan a evitar la aparición de ciclos en el proceso de búsqueda. Los algoritmos diseñados permitieron encontrar individuos que eran capaces de superar a sus enemigos en diferentes escenarios de juegos, lo cual constituye un granito de arena para alcanzar el objetivo principal en el desarrollo de videojuegos que es garantizar (y si es posible incrementar) la satisfacción del jugador humano.

Es bien sabido que la satisfacción del jugador no es fácil de medir, ya que depende de muchas variables como: la personalidad, edad, cultura, habilidades, preferencias, el género, etc.. Aunque se sabe que los juegos que resultan demasiado fáciles o muy difíciles para los jugadores no tendrán éxito, las decisiones de diseño relacionadas con la dificultad de un juego son complicadas de tomar, pues los jugadores tienen diferentes niveles de destreza que van desde un perfil de principiante hasta a aquellos con habilidades profesionales. Por lo tanto, cómo encontramos la forma de equilibrar un juego para que satisfaga la amplia gama de perfiles de usuarios. Sin duda, para esto, la capacidad de adaptación del juego al jugador es un tema clave a considerar. Y partiendo de esta idea, la propuesta que vamos a presentar en este capítulo representa un paso más allá de nuestro trabajo anterior, que además de abordar la búsqueda de IA (es decir comportamientos inteligentes para controlar los NPCs) también incluye la generación automática de contenidos para juegos y permite un acercamiento a esa anhelada adaptabilidad de las partidas.

5.1. Modelo propuesto: MuCCCo

El modelo que se definirá en este capítulo es un intento de implementar la coevolución interespecífica (salvando las distancia con los procesos coevolutivos que ocurren en la naturaleza), por lo que se manejarán (dos) especies que son intrínsecamente diferentes (respecto a la naturaleza de sus dominios). Una de ellas agrupa a los individuos de tipo *bot* cuya codificación se corresponde con la de una estrategia de juego y el otro conjunto representa el contenido del juego (que podría ser mapas o niveles, por

ejemplo). Nótese que la relación de competitividad entre las poblaciones no tiene nada que ver con la situación típica de dos especies competidoras como *depredador-presa* o *parásito-huésped*, en la que el *fitness* de cada individuo está directamente relacionado con el de su competidor (Floreano et al., 1998). Por lo tanto, la manera en que se va a establecer la competición entre el contenido del juego y los *bots* es un reto importante en esta propuesta y hace que sea novedosa en el campo de la IA aplicada a videojuegos.

El principal aporte de este modelo consiste en la capacidad de permitir la generación automática de IA para juegos (es decir, estrategias de juego que rigen el comportamiento del NPC), y a su vez ayudar a los diseñadores a crear contenido que se adapte dinámicamente para alcanzar objetivos específicos. Imaginemos, por ejemplo, un mapa/nivel que se adapta para favorecer a los jugadores novatos frente a los más experimentados, de manera que todos ellos puedan disfrutar de la partida (es decir, que la experiencia de juego no resulte ni demasiado fácil para el jugador experimentado, ni muy difícil para los jugadores novatos).

Desde la perspectiva del diseñador, nuestro enfoque puede ser visto como un modelo para la adaptación dinámica del juego de acuerdo con el perfil de un jugador específico a través de PCG. Hasta lo que nuestro conocimiento alcanza, esta es la primera vez que un algoritmo de CC se utiliza para manejar la coevolución de contenido del juego, así como la IA del juego. Hay una sola, condición obligatoria para la viabilidad de este enfoque, la existencia de una relación competitiva entre las especies que co-evolucionan es estrictamente necesaria.

Es importante destacar que nuestro sistema coevolutivo ha sido diseñado con un enfoque genérico que lo hace aplicable a cualquier género de juego que promueva la competición entre los jugadores. En este capítulo se explica una prueba de concepto para demostrar la viabilidad de la propuesta usando como instancia a un juego RTS. También, se demuestra a través de una amplia fase experimental, que es posible generar el contenido que se adapta a los objetivos específicos, además de estrategias de IA que

son capaces de vencer a algunos de los mejores *bots* descritos en la literatura científica para el juego de prueba empleado.

Los resultados mostrados en este capítulo han sido previamente publicados en (Nogueira et al., 2016). A continuación, se presenta el modelo general definido para un Algoritmo CC de Múltiples Contenidos (MuCCCo) y más adelante se describen también tres variantes de este algoritmo.

5.1.1. Descripción del esquema básico de MuCCCo

La propuesta MuCCCo emplea dos poblaciones para generar automáticamente tanto las estrategias de un juego como su contenido. Desde el punto de vista del diseño, el proceso de generación de contenido es guiado por restricciones o criterios de efectividad que están asociados con un objetivo específico como puede ser mejorar la diversidad del contenido (Liapis et al., 2013b), o satisfacer indicadores estéticos o preferencias humanas con respecto al balance de la dificultad en la partida (Lara-Cabrera et al., 2013b), por citar sólo algunos ejemplos.

En un sistema CC tradicional, la IA evoluciona con el objetivo de encontrar “buenos” jugadores virtuales que superen a los campeones de las poblaciones adversarias. Nuestra propuesta parte de este enfoque, pero el escenario del juego está influenciado por los individuos de la población que representan el “contenido”. Es decir, que no sólo se va a evolucionar con el fin de obtener buenas estrategias de IA sino que parte del juego en sí mismo (en este caso el escenario) se verá determinado por la búsqueda coevolutiva. La principal particularidad de nuestro modelo es que estos dos dominios (estrategias y contenido) son de diferentes naturaleza e imponen objetivos distintos en un proceso de optimización. Precisamente por esto, en el ciclo coevolutivo se genera una interrelación entre las poblaciones que no es estándar.

Los dos dominios a evolucionar no son intrínsecamente competitivos (por su propia naturaleza) y la coevolución del contenido del juego no puede ser guiada por una

confrontación directa contra los campeones de la población de *bots*. Además, incluso en el caso de un juego multijugador, la generación de contenido requiere tener un objetivo específico que determine cuándo un individuo (de la población de contenido) se puede calificar de campeón al ser empleado como “escenario de juego” en las batallas entre los *bots*. Aquí, en este contexto se pueden coevolucionar múltiples poblaciones que contengan estrategias de juego (es decir, la IA de los NPCs) y múltiples poblaciones que representen contenido diverso del juego (Véase la Figura 5.1). Tiene sentido fijar una estrategia de IA que marque los objetivos (seguramente impuestos por el diseñador del juego) que van a dirigir la evolución de los mapas (digamos del contenido en general).

Considere este *bot* fijo (FRB, por sus siglas en inglés: Fixed Rival Bot), por ejemplo, como el modelo de un perfil de jugador y que los *bots* - de las otras poblaciones de IAs del juego - van a ser entrenados durante la evolución para derrotar a los jugadores con ese perfil; mientras que el contenido del juego también evoluciona conducido por objetivos relacionados con este FRB. Por ejemplo, el contenido del juego podría adaptarse para favorecer (o desfavorecer) al FRB si éste representa un modelo de jugador principiante (resp. a un jugador hábil). Esto básicamente significa que el escenario del juego se adapta para satisfacer una decisión de diseño, es decir, en este caso, impidiendo que un juego sea demasiado difícil (o fácil) para el principiante. Por supuesto, se pueden pensar otros objetivos para la evolución del contenido del juego con respecto al FRB(s).

Vea en la Figura 5.1 la explicación visual de cómo el modelo MuCCCo se puede ajustar a juegos multijugador y a múltiples FRB. Para un juego de n -jugadores, tenemos $n - 1$ poblaciones de IA que evolucionan de acuerdo con algunos objetivos (de nuevo, asociados a decisiones de diseño) relacionados con todos y cada uno de los FRBs. También se puede tener m poblaciones, una para cada tipo de contenido, que establecerían el escenario del juego para las batallas entre los $n - 1$ jugadores y los FRBs. La disposición de estas batallas puede variar dependiendo de los objetivos del diseño.

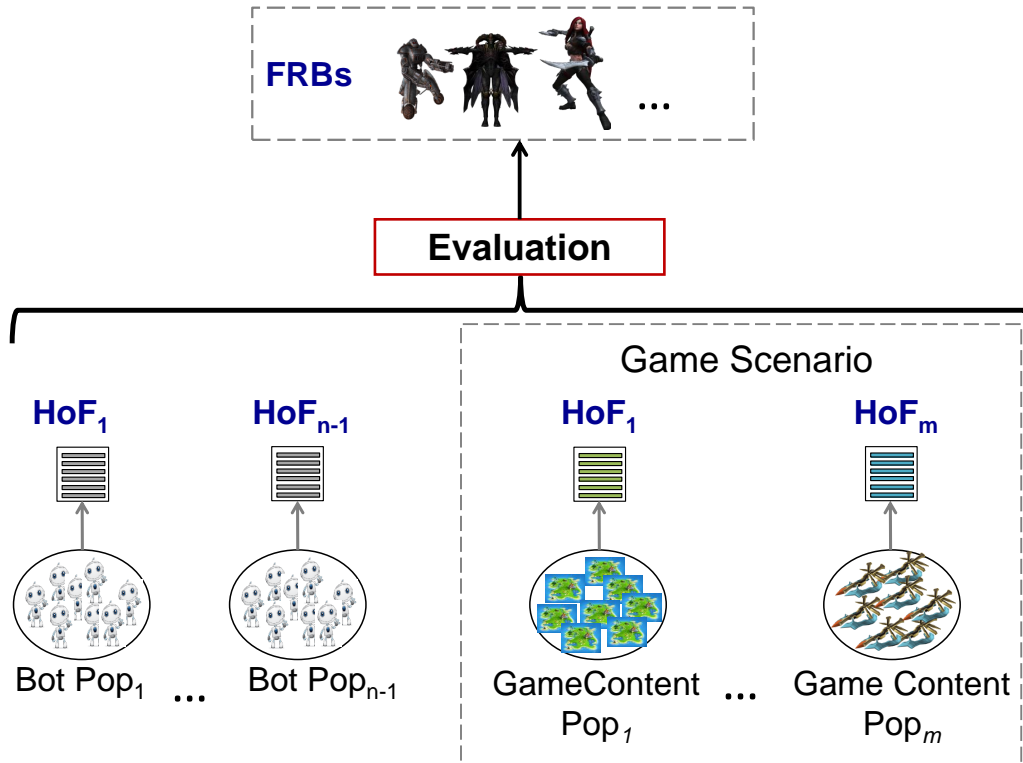


Figura 5.1: Esquema del modelo coevolutivo multi-contenido para un juego de n -jugadores

Tenga en cuenta que nuestro objetivo no está dirigido a la coevolución multijugador aunque la evolución del *bot* podría adaptarse fácilmente a la evolución multi-*bot* mediante la fijación del contenido del juego y siguiendo el esquema mostrado en (Togelius et al., 2007a); sino más bien, está enfocado en permitir la adaptación del contenido del juego de acuerdo con políticas de diseño específicas.

El modelo propuesto podría aplicarse a cualquier juego, el FRB sería la semilla para dirigir el proceso de especialización, el diseñador podría utilizar solo un FRB o varios dependiendo de los objetivos que quiera fomentar en el proceso de búsqueda. Obsérvese también que el perfil de un jugador específico puede extraerse utilizando técnicas de modelado estándar o avanzadas (lo cual está más allá del objetivo de esta tesis) que

podrían combinarse perfectamente con la propuesta realizada aquí.

El esquema MuCCCO gestiona dos poblaciones, una que codifica contenido del juego, y la otra estrategias de IA. Cada población utiliza su propio HoF como un mecanismo de memoria a largo plazo para mantener a los individuos ganadores encontrados en cada paso coevolutivo, y cada miembro del HoF se utiliza en el proceso de evaluación de la población contraria.

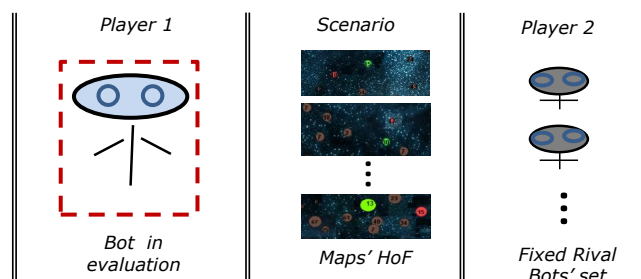
MuCCCo emplea el esquema básico del HoFCC presentado en el capítulo anterior y mantiene los parámetros: α , que indica (en porcentaje) la porción del HoF que se eliminará (durante el proceso de actualización del HoF) con el objetivo de mantener sólo a los campeones más valiosos, y λ , que define la frecuencia de ejecución de esta actualización del *HoF*.

Uno o varios FRB pueden ser considerados durante la evaluación. Para explicarlo a continuación vamos a considerar un solo FRB y más adelante emplearemos varios de ellos en los experimentos. El funcionamiento sería: en el paso coevolutivo r_{th} , cada candidato en la población de *bots* se enfrenta a un rival fijo *FRB* en $r - 1$ batallas ejecutadas sobre cada mapa de $\{map_1, map_2, \dots, map_{r-1}\}$, donde map_i es el campeón de la población de mapas que se encontró en el paso coevolutivo i -th ($1 \leq i \leq r - 1$). Por su parte, cada candidato en la población de contenido del juego también se evalúa en la iteración coevolutiva r_{th} mediante un conjunto de batallas, en el escenario impuesto por el candidato (o sea, sobre el mapa que representa el candidato en sí mismo), entre el *FRB* y cada miembro del HoF de *bots*: $\{bot_1, bot_2, \dots, bot_{r-1}\}$.

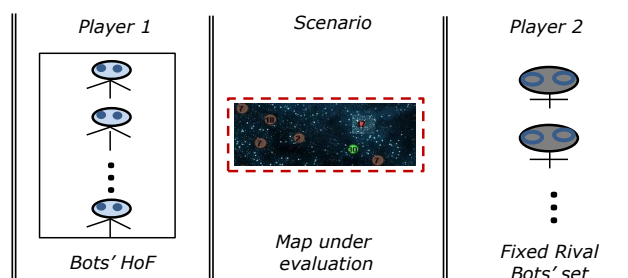
La Figura 5.2 muestra cómo se configura el escenario de batalla para evaluar cada población en un juego de dos jugadores, 'palyer 1' y 'player 2' denotan cada uno de los jugadores considerados en una batalla. La función de aptitud de los *bots* evalúa la capacidad que tiene un individuo para derrotar a los oponentes (o sea, cada *FRB*) sobre un conjunto finito de mapas, mientras que la función de aptitud de los mapas evalúa cuánto la naturaleza de un mapa puede favorecer al oponente fijo cuando éste

CAPÍTULO 5. GENERACIÓN SIMULTÁNEA DE CONTENIDOS Y ESTRATEGIAS PARA JUEGOS USANDO COEVOLUCIÓN

está compitiendo contra un conjunto de *bots* en diferentes batallas.



(a) Evaluación de los *bots*



(b) Evaluación de los Mapas

Figura 5.2: Configuración del escenario de batallas para ambas poblaciones (el individuo que está siendo evaluado se resalta con líneas discontinuas)

El Algoritmo 6 muestra el esquema de MuCCCo adaptado (por simplicidad) a un juego de 2 jugadores. Inicialmente el objetivo es encontrar un individuo ganador en la población de *bots* (es decir, la población seleccionada como activa – Línea 1 – que será la primera que evoluciona). La población marcada como activa es la que será evolucionada en el ciclo coevolutivo, y ese rol (de activa o no activa) se irá alternando entre las dos poblaciones (Línea 20) hasta el final del proceso.

El HoF de la población no activa inicialmente (es decir, los mapas) se inicializa con algunos individuos (generados al azar o manualmente) (Línea 2, método `PRELOADMAPS()`) mientras que el HoF del *bot* está inicialmente vacío (Línea 2).

CAPÍTULO 5. GENERACIÓN SIMULTÁNEA DE CONTENIDOS Y ESTRATEGIAS PARA JUEGOS USANDO COEVOLUCIÓN

Llegado el momento, el HoF de la población que está evolucionando se actualiza (con el fin de que se sólo se mantengan en él los campeones más robustos) de acuerdo con algunos criterios de idoneidad (Líneas 4-6). La inicialización de la población activa se realiza aleatoriamente en la Línea 7, y luego se evalúa la calidad de sus candidatos. Entonces comienza un proceso evolutivo estándar que trata de encontrar un individuo victorioso en la población activa (Líneas 11-24).

Si se encuentra un individuo victorioso (Línea 17), se agrega al HoF de la población activa (Línea 19) y el proceso se inicia de nuevo pero con los roles de las poblaciones invertidos (Línea 20). En caso contrario, si no se encuentra ningún individuo ganador, se reinicia el proceso de búsqueda manteniendo los mismos roles en las poblaciones. Si después de un cierto número de pasos coevolutivos seguidos no se encuentra ningún individuo ganador en la población activa, se considera que la búsqueda se ha estancado y la coevolución termina (véase la condición de parada del ciclo en la Línea 3). Al final de todo el proceso obtenemos como resultado dos conjuntos de soluciones ganadoras (es decir, dos HoF) una por con cada población.

Una estrategia de juego se considera ‘ganadora’ si es capaz de vencer a los FRBs en todos los mapas que pertenecen a HoF de la población de mapas. Por otro lado, un mapa es considerado un individuo victorioso si cumple con dos restricciones: la primera es que el FRB tiene que derrotar (o empatar) al enfrentarse a cada miembro del HoF de *bots* en batallas donde dicho mapa (es decir, el mapa evaluado) es el escenario de juego. La segunda establece que el FRB en su resultado en las batallas contra los *bots* no puede superar cierto límite de planetas conquistados; esta segunda restricción se impone para evitar la generación de mapas corruptos que no ofrezcan ninguna posibilidad de ganar a los *bots*, pues en experimentos preliminares se dieron casos de mapas que eran totalmente favorecidos por el proceso de búsqueda y ofrecían una ventaja muy marcada al FRB, en la Figura 5.3 se pueden ver estos casos.

En resumen, tenemos que los *bots* se evalúan mediante enfrentamientos directos

Algoritmo 6: Algoritmo MuCCCo(α, λ)

```

1   $nCoev \leftarrow 0$ ;  $activePop \leftarrow Bots$ ;  $nonActivePop \leftarrow Maps$ ;  $\phi \leftarrow thresholdvalue$ ;
2   $HoF_{Bots} \leftarrow \emptyset$ ;  $HoF_{Maps} \leftarrow PRELOADMAPS()$ ;  $frb \leftarrow FIXEDRIVALBOT()$ ;
3  while  $nCoev < MaxFailCoevolutions \wedge \neg timeout$  do
4      if  $nCoev \bmod \lambda = 0$  then //HoF updating every  $\lambda$  Coevolutions
5          |  $PURGE_{\alpha}(HoF_{activePop})$ ; // HoF updating
6      end if
7       $pop_{activePop} \leftarrow RANDOMSOLUTIONS()$ ; // Active pop randomly
          initialized
8       $pop_{activePop} \leftarrow EVALUATEPOP_{frb}(HoF_{nonActivePop})$ ; // Evaluate
          candidates in Active population against rival HoF and FRB
9       $i \leftarrow 0$ ;
10      $foundWinner \leftarrow false$ ;
11     while  $(i < MaxGen) \wedge \neg foundWinner$  do
12         |  $parents \leftarrow SELECT(pop_{activePop})$ ;
13         |  $children' \leftarrow RECOMBINE(parents, p_X)$ ;
14         |  $children'' \leftarrow MUTATE(children', p_M)$ ;
15         |  $pop_{activePop} \leftarrow REPLACE(children'')$ ;
16         |  $pop_{activePop} \leftarrow EVALUATEPOP_{frb}(HoF_{nonActivePop})$ ;
17         | if  $(FITNESS_{activePop}(BEST(pop_{activePop})) \geq \phi_{activePop})$  then //winner
18             | found!
19             |  $foundWinner \leftarrow true$ ;
20             |  $HoF_{activePop} \leftarrow HoF_{activePop} \cup \{BEST(pop_{activePop})\}$ 
21             |  $temp \leftarrow activePop$ ;  $activePop \leftarrow nonActivePop$ ;
22             |  $nonActivePop \leftarrow temp$ ; // interchange populations'
23             | activity roles
24         | else
25             |  $i \leftarrow i + 1$ ;
26         | end if
27     end while
28     if  $(foundWinner = true)$  then
29         |  $nCoev \leftarrow 0$ ; // start new search
30     else
31         |  $nCoev \leftarrow nCoev + 1$ ; // continue search
32     end if
33 end while

```

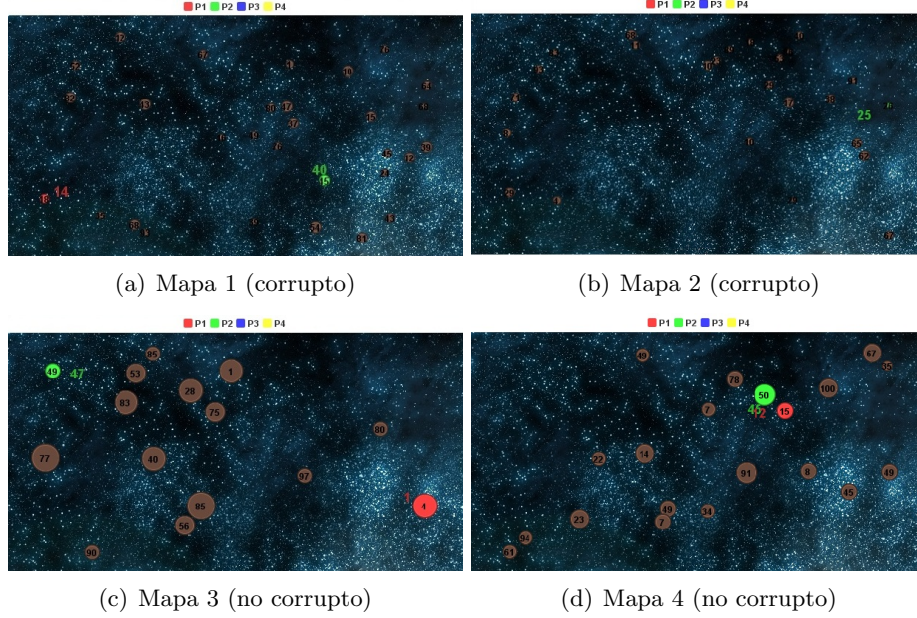


Figura 5.3: Ejemplos de mapas generados con el algoritmo $\text{MuCCCo}(\alpha, \lambda)$ para el juego *Planet Wars*: tanto los planetas como las flotas son identificados por un número que muestra la cantidad de naves que poseen. Sus colores identifican el jugador al que pertenecen, el *FRB* se denota por el color verde.

contra el *FRB* en todos los mapas del *HoF* correspondiente. Por lo tanto, dada una estrategia específica $s \in \text{Pop}_{\text{Bots}}$, su *fitness* (F) se calcula de la siguiente manera:

$$F_{\text{Bots}}(s) = \frac{1}{k} \sum_{m \in \text{HoF}_{\text{Maps}}} (r_m^{s, \text{FRB}} + (C_1 - n\text{Turn}_{s, \text{FRB}}(m)) + C_2 \cdot Q_{s, \text{FRB}}(m)) \quad (5.1)$$

- $k = \#\text{HoF}_{\text{Maps}} \in \mathbb{N}$ es la cardinalidad del *HoF* de mapas;
- La función $r_m^{a,b} \in \mathbb{R}$ devuelve ϕ puntos si la estrategia a derrota a la estrategia b en el mapa m , $\frac{\phi}{2}$ si la batalla termina en empate, y 0 si el *FRB* gana a la estrategia s ; entonces, por ejemplo, $r_m^{s, \text{FRB}}$ debe devolver ϕ puntos si la estrategia s derrota al *FBR* sobre el mapa m .

CAPÍTULO 5. GENERACIÓN SIMULTÁNEA DE CONTENIDOS Y ESTRATEGIAS PARA JUEGOS USANDO COEVOLUCIÓN

- La función $nTurn_{a,b}(m) \in \mathbb{N}$ devuelve el número de turnos que dura la partida, que jugada en el mapa m , termina con la victoria de a o con un empate, y devuelve cero en otro caso.
- and $Q_{a,b}(m) \in \mathbb{R}$ es el porcentaje de planetas conquistados (respecto al total de planetas que hay en el mapa m) por a cuando termina la batalla.
- C_1 y C_2 son dos valores constantes que hemos asignado en nuestros experimentos, $C_1 = 500$ y $C_2 = 10$, y que se han definido a partir de nuestra experiencia de juego. Su fin es el de hacer más significativas en la función del *fitness* las variables que representan el número de turnos y la cantidad de planetas conquistados.

Para la evaluación de los mapas se utiliza la Ecuación (5.2) donde $m \in Pop_{Maps}$ es el mapa a evaluar, la función $r_m^{a,b} \in \mathbb{R}$ se define como se mostró antes y $l = \#HoF_{Bots} \in \mathbb{N}$ es la cardinalidad del HoF de *bots*. En esta nueva ecuación de aptitud se emplea una modificación de la función $Q_{a,b}(m)$ usada en la Ecuación (5.1). La razón de esto, como se ha mencionado antes, es penalizar los *mapas corruptos*, porque en experimentos preliminares, basados en la función $Q_{a,b}(m)$ nuestro algoritmo generó (como campeones) mapas donde la mayoría de los planetas tenían una posición claramente beneficiosa con respecto al planeta de origen del FRB. Obsérvese que el número de turnos no se considera en esta función, inicialmente había sido incluido, pero los primeros experimentos mostraron que encontrar individuos que permitieran la victoria del FRB sin “aplastar” a los *bots* adversarios era muy difícil para la población de los mapas. Por lo tanto, tratamos de reducir la presión de selección durante el proceso de búsqueda centrándonos sólo en el objetivo de lograr mapas victoriosos.

$$F_{Maps}(m) = \frac{\sum_{s \in HoF_{Bots}} (r_m^{FRB,s} + C_2 \cdot Q'_{FRB,s}(m))}{l} \quad (5.2)$$



Donde

$$Q'_{a,b}(m) = \begin{cases} 0, & Q_{b,a}(m) < \rho \\ Q_{a,b}(m) + Plus_{b,a}(m), & Q_{b,a}(m) \geq \rho \end{cases}$$

y

$$Plus_{b,a}(m) = Q_{b,a}(m) - \rho$$

La función de *fitness* de los mapas valora positivamente aquellos mapas en los que los *bots* no son ferozmente derrotados por el FRB, lo que básicamente significa que el *bot* debe conquistar al menos el $\rho\%$ de los planetas en el mapa. Esta restricción evita la convergencia del algoritmo hacia mapas corruptos. Obsérvese también que se da una bonificación a aquellos mapas en los que el *bot* perdedor conquista más del $\rho\%$ del mapa. De esta manera se intenta intensificar la búsqueda de mapas que generen mayor actividad entre los juegos. Se han probado diferentes valores para $\rho \in [0\%, 50\%]$ pero encontramos que cuando ρ estaba cerca de 0% todos los planetas tienden a alinearse muy próximos a la posición inicial del FRB y lejos de la posición inicial del *bot* oponente. A su vez, cuando ρ tiende al 50% , el algoritmo tenía serios problemas para encontrar campeones de mapas (probablemente porque los mapas no pudieron por sí mismos encontrar el equilibrio del juego). Finalmente, basándonos en nuestra experiencia de juego y los experimentos preliminares, se establece que $\rho = 20\%$.

Es esta experimentación se usarán las variantes del HoFCC que fueron explicadas en el capítulo anterior pero adaptadas a este contexto multi-contenido. A continuación se explica la implementación de cada una para este nuevo dominio.

5.1.2. Variante del modelo MuCCCo que potencia la diversidad de las soluciones (MuCCCo Div)

Para la población de estrategias, la diversidad proporcionada por cada uno de los miembros del HoF se calcula a través de la distancia genotípica de la siguiente manera. Manejamos la memoria de campeones como una matriz en la que cada fila representa

CAPÍTULO 5. GENERACIÓN SIMULTÁNEA DE CONTENIDOS Y ESTRATEGIAS PARA JUEGOS USANDO COEVOLUCIÓN

una solución y cada columna un gen (es decir, una acción en la estrategia). Tenga en cuenta que hay 9 columnas según la codificación de las estrategias de IA definida para el *Planet Wars* (ver Capítulo 3, Sección 3.5.2). El valor de entropía para una columna específica j se calcula basado en la Ecuación (4.2) definida en la Sección 4.2.2 y sería:

$$H_j = - \sum_{i=1}^{\#HoF_{Bots}} (p_{ij} \log p_{ij}) \quad (5.3)$$

donde p_{ij} es la probabilidad de la acción i en la columna j , y $\#HoF_{Bots} \in \mathbb{N}$ es la longitud de la memoria de *bots* campeones. Finalmente, la entropía de todo el conjunto se calcula:

$$H = \sum_{j=1}^9 H_j \quad (5.4)$$

Cuanto mayor sea el valor de H mayor será la diversidad del conjunto. Para determinar la contribución de la diversidad a una solución específica, calculamos el valor de la entropía con esta solución dentro del conjunto y el valor con ella fuera del conjunto y finalmente, la diferencia entre estos dos valores representa la Contribución de diversidad de esa solución. Esta noción de distancia genotípica es similar al método *Novelty Search* (Lehman y Stanley, 2011), ya que ambos pretenden mantener la diversidad en un conjunto de soluciones. Sin embargo, nuestra propuesta es diferente porque la diversidad no es un objetivo en nuestro proceso de búsqueda sólo se utiliza como un indicador a tener en cuenta en la política seguida para mantener un tamaño manejable de los HoFs. Y con esta idea, se asume que los individuos eliminados no afectarán la calidad del proceso de búsqueda.

Para el caso de los mapas, el cálculo se realiza mediante el análisis de la distancia media entre los planetas del mapa con respecto a todos los otros mapas en el HoF:

$$D(m_j) = \frac{\sum_{i=1}^k (S_{ij})}{k} \quad (5.5)$$

donde $k \in \mathbb{N}$ es la cardinalidad del HoF de mapas del cual el individuo m_j es miembro, y $\forall i \in HoF\{m_{i=0}, m_{i=1}, \dots, m_{i=k}\}; i \neq j$ se calcula:

$$S_{ij} = \frac{(eS_{ij}) + (eS_{ji})}{2} \quad (5.6)$$

donde eS_{ij} es el promedio de la distancia de Euclidean de cada planeta contenido en el mapa $i - th$ (i.e. el mapa m_i del HoF de la población de mapas) a cada planeta del mapa $j - th$ (i.e el mapa m_j del HoF); y eS_{ji} representa el mismo concepto de distancia pero esta vez se toma como punto de inicio el planeta $j - th$ para calcular la distancia de Euclidean.

5.1.3. Variante del modelo MuCCCo que se enfoca en la calidad de las soluciones (MuCCCo-Quality)

La medida de calidad para el *Planet Wars* se definió sobre la base de nuestra experiencia de juego, en el caso de los *bots*, sería el número de derrotas que ese miembro del HoF sufrió en el paso coevolutivo anterior. Para los mapas, la calidad se define como el número de derrotas que cada FRB sufrió cuando tuvo que enfrentarse a enemigos en este mapa también en el paso coevolutivo anterior. En lo que sigue, y abusando del lenguaje, diremos que un mapa *sufre una derrota contra una estrategia de juego b* (es decir, un *bot*) si el FRB es derrotado por *b* en ese mapa.

5.1.4. Variante del modelo MuCCCo que combina las heurísticas de diversidad y calidad (MuCCCo-U)

Esta variante aplica modificaciones a las funciones de *fitness* definidas en 5.1 y 5.2. Estos cambios están inspirados en el enfoque del CFS visto en el capítulo anterior. La idea principal es que una victoria contra un oponente *b* se condidera más relevante si *b* ha vencido a un número significativo de oponentes. Entonces, dado un dominio $d \in \{Bots, Maps\}$, un valor de penalización N para cada individuo i (para $1 \leq i \leq n$)

en la población Pop_d es calculado como sigue:

$$N_i = 1 - \frac{1}{k} \sum_{j=1}^k \frac{v_{ij}}{V(j)} \quad (5.7)$$

donde $v_{ij} = 1$ si el individuo i -th de la población del dominio d derrota¹ al campeón j -th del HoF rival (cuya longitud es k , o sea $k = \#Pop_d$) y 0 en otro caso. Además:

$$V(j) = \sum_{i=1}^{\#Pop_d} v_{ij} \quad (5.8)$$

es el número de individuos en la población que logra vencer al oponente j en el HoF rival. Como consecuencia, $N_i \approx 0$ si el candidato i -ésimo derrota a todos los oponentes del HoF rival y es uno de los pocos candidatos en lograrlo; $N_i = 1$ si no derrota a ningún oponente; y $0 < N_i < 1$ dependiendo de cuántas veces gane y de lo usual que sea vencer a ciertos oponentes. El *fitness* de un candidato i es entonces calculado como sigue:

$$F_i = P_i - \omega N_i \quad (5.9)$$

donde P_i es el resultado obtenido en las batallas según las Ecuaciones (5.1) o (5.2) dependiendo de si se trata de *Bots* o de *Mapas*, y $\omega \in \mathbb{N}$ es un coeficiente que se usa para escalar el valor de N_i con el fin de hacerlo más significativo respecto a P_i .

5.2. Experimentos y resultados

Esta sección describe el análisis experimental realizado en el juego del *Planet Wars*. Consideramos dos instancias para cada algoritmo (MuCCCo-Div, MuCCCo-Qua y MuCCCo-U) que varían según el valor de $\alpha \in \{10\%, 50\%\}$. La notación *MuCCCo-Variante- α* (donde *Variante* $\in \{Div, Qua, U\}$) se utiliza para denotar cada una de las

¹Recuerde que, en el caso que $d = Maps$, se considera que un mapa m derrota a un *bot* b si el FRB es vencido por b en una batalla sobre este mapa m .



Tabla 5.1: Parámetros del ciclo coevolutivo

Parámetro	Valor
<i>maxCoevFallidas</i>	10
<i>maxCoevoluciones</i>	100
<i>maxGeneraciones</i>	50
<i>tamañoPoblación</i>	30
<i>maxEvaluaciones</i>	10000
<i>probCruce</i>	$p_X = 1,0$
<i>probMutación</i>	$p_M = 0,01$
ϕ <i>Bots</i>	1600
ϕ <i>Mapas</i>	1250
λ	3

instancias. En todos los casos se establece $\lambda = 3$. Se realizaron diez ejecuciones de cada instancia del algoritmo. La base evolutiva empleada en el ciclo coevolutivo es un algoritmo genético de estado estacionario que usa el torneo binario para la selección, el cruce uniforme, la mutación *bit-flip* y el reemplazo elitista. Al principio del ciclo coevolutivo el *mapa 10* (de la colección de mapas diseñada para el concurso original de Google AI Challenge 2010) se establece como el *mapa oponente* (es decir, este mapa es agregado al Hof de la población de mapas). También se eligió el *ProspectorBot*, del conjunto de *bots* que fueron originalmente proporcionados como *ejemplos de bots* en el concurso como el FRB, porque es una estrategia bastante ofensiva que puede ayudar a producir *bots* con un adecuado espíritu de lucha. A continuación, se dan detalles de la configuración de los experimentos y se analizan los resultados obtenidos.

5.2.1. Configuración de los experimentos

Todos los experimentos fueron ejecutados usando el *Planet Wars*. La Tabla 5.1 muestra los parámetros utilizados en los experimentos, note que estos parámetros coinciden con los explicados anteriormente en el Capítulo 3, Sección 4.2.4.

Hemos analizado los resultados obtenidos con el test de Kruskal-Wallis (ver los resultados en la Tabla 5.2). Además, en los casos en que esta prueba detectara diferencias

significativas en las distribuciones, se empleó el método Dunn–Sidak para determinar cuáles pares de algoritmos eran significativamente diferentes y cuáles no.

Tabla 5.2: Resultados del test de Kruskal-Wallis para los indicadores.

Indicador	p -value
Mejor fitness de <i>Bots</i>	9.0524e-007
Mejor fitness de Mapas	4.5878e-006

5.2.2. Análisis de los resultados

Las Figuras 5.4 y 5.5 muestran el *fitness* del mejor individuo encontrado en cada ejecución de las instancias del algoritmo en las poblaciones de mapas y *bots*, respectivamente. Recordemos que las instancias analizadas en esta sección son MuCCCo-Qua-10, MuCCCo-Qua-50, MuCCCo-Div-10, MuCCCo-Div-50, MuCCCo-U-10 y MuCCCo-U-50.

La prueba de Kruskal-Wallis confirma que las diferencias entre algunos valores son estadísticamente significativas y según los resultados de la población de *bots* (Figura 5.4), los algoritmos de la familia HoFCC-Qua obtienen los mejores resultados. Le siguen los que se basan en las métricas de diversidad, mientras que las versiones multiobjetivos (es decir, los HoFCC-U) muestran un comportamiento menos competitivo. Esta última afirmación puede deberse a que las versiones ‘U’ tienen más difícil obtener un alto valor de aptitud debido a la penalización que se aplica a la puntuación de acuerdo con la ecuación (5.9). El resultado de la prueba basada en el método Dunn–Sidak corrobora que las distribuciones del ‘Div-10’ y ‘U-50’, y las familias ‘Qua’ y ‘U’ son estadísticamente diferentes entre sí.



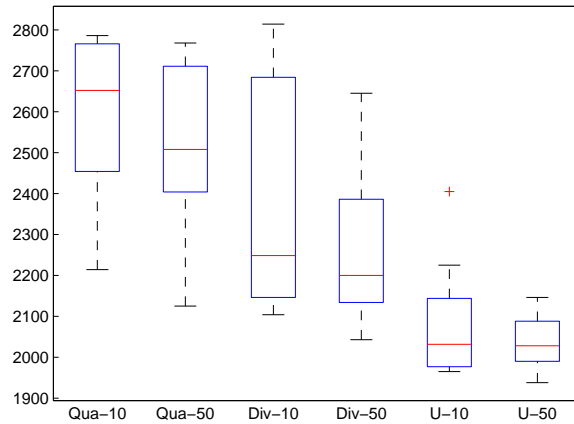


Figura 5.4: Análisis estadístico del mejor *fitness* de la población de *bots* encontrado por cada versión del algoritmo $\text{MuCCCo}(\alpha, \lambda)$.

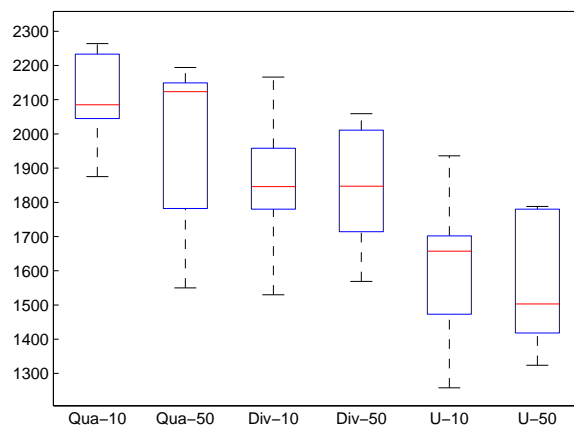


Figura 5.5: Análisis estadístico del mejor *fitness* de la población de mapas encontrado por cada versión del algoritmo $\text{MuCCCo}(\alpha, \lambda)$.

Respecto al comportamiento del *fitness* de los mapas mostrado en la Figura 5.5, los algoritmos ofrecen resultados muy similares a los de los *bots*. Una vez más, se observa una clara distinción entre las tres familias de algoritmos, el test de Dunn–Sidak nos confirmó que MuCCCo-Qua y MuCCCo-U son estadísticamente diferentes entre sí.

CAPÍTULO 5. GENERACIÓN SIMULTÁNEA DE CONTENIDOS Y ESTRATEGIAS PARA JUEGOS USANDO COEVOLUCIÓN

Para comparar los algoritmos desde otro punto de vista, la Figura 5.6 muestra la representación del mejor *fitness* de los *bots*, el mejor *fitness* de los mapas y el número de evaluaciones de cada ejecución (en los ejes X , Y , y Z respectivamente). Cada punto (x, y, z) corresponde a una de las 10 ejecuciones realizadas por cada instancia del algoritmo. Como se puede observar, los algoritmos se agrupan en familias, y cada familia comprende 20 puntos correspondientes a las 10 ejecuciones del algoritmo para los valores de $\alpha = 10\%$ y $\alpha = 50\%$. Esta decisión de agrupar los resultados por familias ha estado motivada pues en experimentos preliminares se pudo comprobar que no hay diferencias significativas entre los resultados de las distintas instancias de una misma familia.

Observe en la Figura 5.6 que hay sólo dos soluciones no dominadas en el frente de Pareto, una perteneciente a MuCCCo-Qua y la otra a MuCCCo-Div. Al analizarlos en profundidad, observamos que en ambos casos la dominancia se debe a un valor bajo en el eje Z (es decir, el número de evaluaciones) el cual se corresponde a ciclos coevolutivos muy cortos. En tal caso, este resultado puede ser interpretado como un signo de estancamiento temprano del proceso de búsqueda. Sin embargo, considerando que los valores de *fitness* alcanzados en estos ciclos cortos corresponden a los mejores valores, también podríamos pensar que existe una rápida convergencia hacia soluciones aparentemente robustas. Debemos prestar especial atención a si en ciclos grandes y cortos los algoritmos encuentren valores de *fitness* similares pues eso podría indicarnos que el proceso de búsqueda se ve afectado por una pérdida de requisitos y por lo tanto sólo encuentra soluciones mediocres. Para complementar este análisis, se tendrá en cuenta otro indicador de “robustez” para los *bots* que ayudará a llegar a conclusiones más precisas.

La Tabla 5.3 muestra los resultados de la prueba de robustez para los *bots*. Aquí, los diez campeones *bots* obtenidos en las ejecuciones independientes de las instancias del algoritmo, se enfrentaron entre sí en un torneo *Todos vs. Todos*, y los resultados



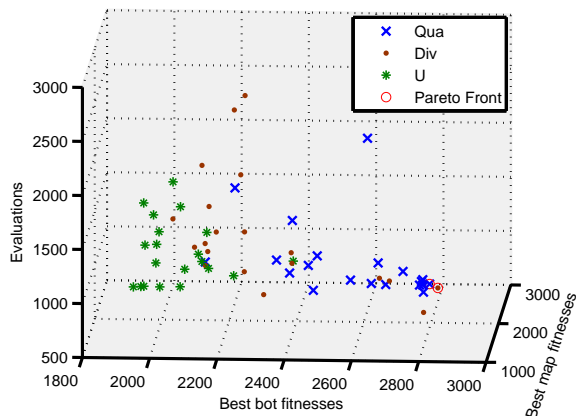


Figura 5.6: Representación en un frente de Pareto de la distribución de los mejores *fitness* de *bots* (eje *x*), los mejores *fitness* de los mapas (eje *y*), y el número de evaluaciones (eje *z*) logrado por cada algoritmo

fueron agrupados por familias (20 por cada familia, 10 para $\alpha = 10\%$ y otros 10 para $\alpha = 50\%$). Entonces, se tiene que, cada individuo compite contra los otros 20 campeones de las familias de algoritmos adversos y en cada confrontación, se ejecutan dos batallas (cada una en un escenario distinto). Los escenarios del juego escogidos son dos mapas, seleccionados al azar de la colección original del *Planet Wars*. La Tabla 5.3 muestra las familias de algoritmos que compiten entre sí, el número de victorias obtenidas por cada uno de ellos y el porcentaje que representa cada valor con respecto al número total de batallas. También muestra el número de empates con su porcentaje correspondiente. En total se ejecutaron 800 batallas (es decir, 20 individuos \times 20 opositores \times 2 mapas).

Téngase en cuenta también que hemos diferenciado las batallas según el orden de los jugadores (es decir, primer jugador como Jugador 1 (J1) y segundo jugador como Jugador 2 (J2)) esto es relevante porque cuando se ejecuta el juego *Planet Wars*, la posición de uno o otro jugador en el mapa puede resultar en una ventaja o desventaja en la partida. La Tabla 5.3 muestra una ligera variación (entre 1 y 2 por ciento) con

CAPÍTULO 5. GENERACIÓN SIMULTÁNEA DE CONTENIDOS Y
ESTRATEGIAS PARA JUEGOS USANDO COEVOLUCIÓN

Tabla 5.3: Resultados del *Todos vs. Todos* en los mapas ‘30’ y ‘70’ (del conjunto original de mapas del *Planet Wars*).

Jugador 1	Jugador 2	Victorias-J1	Victorias-J2	Empates
Qua	Div	203(25 %)	211(26 %)	386(49 %)
Qua	U	287(36 %)	231(29 %)	282(35 %)
Div	Qua	224(28 %)	217(27 %)	359(45 %)
Div	U	284(36 %)	193(24 %)	323(40 %)
U	Qua	218(27 %)	286(36 %)	296(37 %)
U	Div	199(25 %)	299(37 %)	302(38 %)

Tabla 5.4: Resultados del *Todos vs. Todos* en dos mapas elegidos de forma aleatoria de los mapas victoriosos generados por nuestros algoritmos

Jugador 1	Jugador 2	Victorias-J1	Victorias-J2	Empates
Qua	Div	306 (38 %)	466 (58 %)	28 (4 %)
Qua	U	307 (38 %)	458 (57 %)	35 (5 %)
Div	Qua	274 (34 %)	516 (65 %)	10 (1 %)
Div	U	324 (41 %)	427 (53 %)	49 (6 %)
U	Qua	231 (29 %)	544 (68 %)	25 (3 %)
U	Div	265 (33 %)	477 (60 %)	58 (7 %)

respecto al número de victorias y derrotas cuando enfrentamos a los mismos *bots* en los mismos mapas y sólo se intercambian sus posiciones (resp. J1 o J2). En ambos casos (es decir, para las dos posibles posiciones del jugador) MuCCCo-Div parece tener el conjunto de soluciones más difícil de superar y MuCCCo-U es el más vulnerable en las competiciones.

A continuación se repite la prueba anterior, utilizando los mismos *bots* que se han evaluado antes, pero esta vez las batallas tienen lugar en dos mapas que han sido generados por los algoritmos implementados en este trabajo, y que han sido elegidos al azar del conjunto de mapas victoriosos encontrados. Por lo tanto, observando la Tabla 5.4 se ve un comportamiento muy similar en los resultados de los *bots* que se posicionan como Jugador 2. Obsérvese que en todos los casos los mapas claramente favorecen al Jugador 2, y esta vez hay una notable variación en los resultados cuando las posiciones



CAPÍTULO 5. GENERACIÓN SIMULTÁNEA DE CONTENIDOS Y ESTRATEGIAS PARA JUEGOS USANDO COEVOLUCIÓN

de los *bots* en las batallas se intercambian, el número de victorias/derrotas varía entre un 20 y un 30 por ciento (véanse las diferencias entre las celdas con el mismo color).

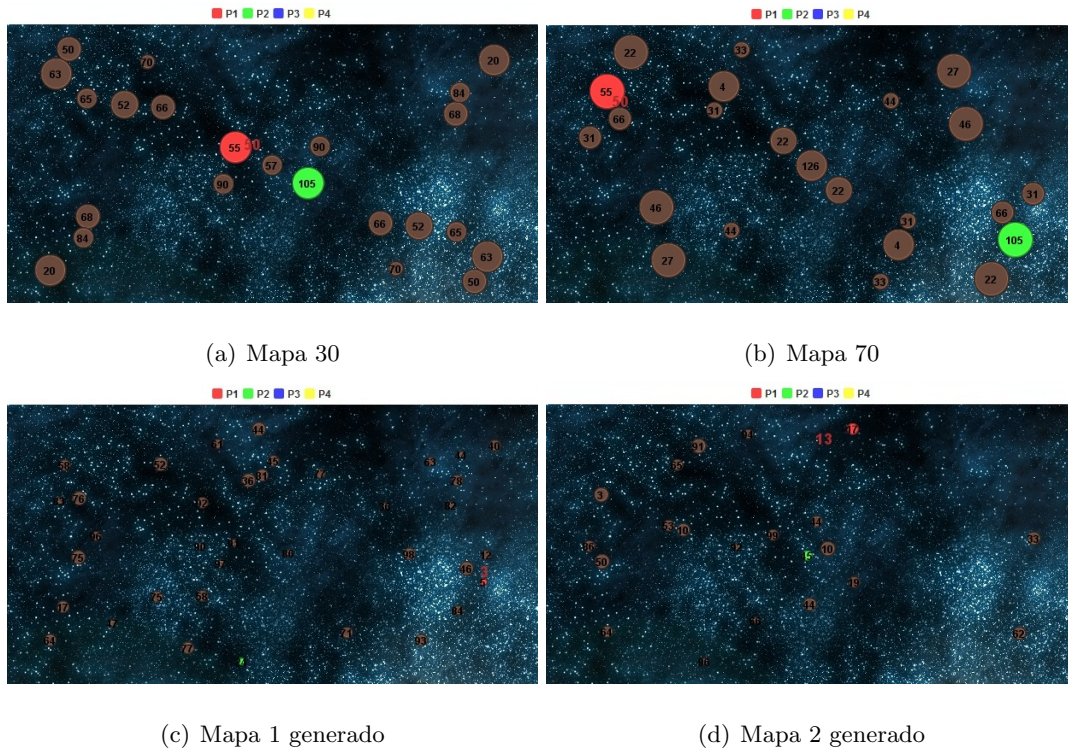


Figura 5.7: Los mapas utilizados en el torneo de *Todos vs. Todos*, los dos primeros son mapas balanceados del conjunto de mapas originales del *PlanetWars* y los otros dos son mapas generados por los algoritmos aquí presentados.

Cabe destacar que nuestros algoritmos coevolutivos intentan encontrar mapas que aseguren la victoria del FRB contra los *bots* del HoF (como se mencionó en secciones anteriores), y se ha dispuesto que este FRB siempre ocupe la posición del Jugador 2 en las batallas que se ejecutan durante el proceso de evaluación. Por lo tanto, este interesante comportamiento sugiere cómo el proceso de aprendizaje que emerge en la población de mapa converge hacia la creación de estructuras más favorables para el Jugador 2. A continuación se aplican algunas pruebas a nuestros mapas para analizarlos con más detalle.

En primer lugar, se muestran los resultados de una evaluación geométrica de los

CAPÍTULO 5. GENERACIÓN SIMULTÁNEA DE CONTENIDOS Y ESTRATEGIAS PARA JUEGOS USANDO COEVOLUCIÓN

mapas, para ello se utilizan los mismos mapas que se analizaron en la Figura 5.5 (que fueron los mejores individuos obtenidos por cada algoritmo en los experimentos descritos anteriormente). Pero esta vez en lugar de tener seis algoritmos tenemos sólo tres porque están agrupados por familias y se agrega otro conjunto que contiene 20 mapas (que fueron elegidos al azar) del conjunto de mapas oficiales del *Planet Wars*. Algunas de las mediciones utilizadas para esta prueba se basan en el análisis estético de mapas presentado en (Lara-Cabrera et al., 2013a). Estos son indicadores relacionados con la distribución espacial de los planetas y otras características como el número de naves y la tasa de crecimiento. La primera prueba explora las diferencias entre los conjuntos de mapas con respecto a la distancia media de los planetas al planeta inicial del Jugador 2 en cada mapa, esta distancia se calcula mediante la fórmula de la distancia euclídea. La Figura 5.8 muestra que en nuestros mapas la distancia media

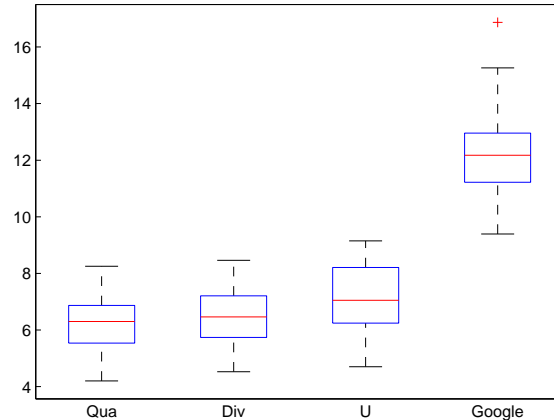


Figura 5.8: Análisis estadístico de la distancia promedio de los planetas al planeta inicial del Jugador 2 para cada uno de los mapas del HoF.

es significativamente menor que el conjunto de mapas de Google. Las distribuciones de valores de nuestros mapas se comportan de manera similar entre ellas, ninguna se destaca por encontrar soluciones especialmente radicales, pero la prueba muestra que hay diferencias significativas entre nuestros mapas y el conjunto de mapas de Google.

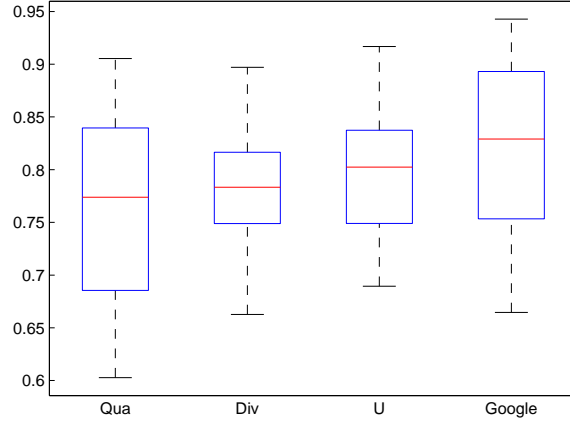


Figura 5.9: Análisis estadístico de la Correlación de Pearson entre el “número de naves” y el “ratio de crecimiento”.

Una segunda prueba fue aplicada a los mapas, la cual se centró en el “número de buques” y la “tasa de crecimiento” y chequeó si había una relación entre los valores asignados a estas características cuando los mapas eran creados por nuestro algoritmo. Sea s_i y w_i , respectivamente, el número de naves y la tasa de crecimiento del planeta i en un mapa (N es el número de planetas en este mapa). Entonces, se especifica el promedio y la desviación estándar de esta característica (μ_s y σ_s , respectivamente) y la Correlación de Pearson entre el número de buques y la tasa de crecimiento ρ como sigue:

$$\mu_s = \sum_{i=1}^N s_i \quad (5.10)$$

$$\sigma_s = \sqrt{\frac{\sum_{i=1}^N (s_i - \mu_s)^2}{N}} \quad (5.11)$$

$$\rho = \frac{\sum_{i=1}^N s_i w_i - N \mu_s \mu_w}{N \sigma_s \sigma_w} \quad (5.12)$$

De acuerdo con la Figura 5.9 en todas las distribuciones analizadas las relaciones entre las variables son positivas, para los casos de los conjuntos “Div” y “U” muestran

menos dispersión en los valores pero no es una diferencia significativa con respecto a los otros conjuntos. Para resumir, se encontró que existe una relación que puede ser clasificada como “alta” porque en todos los casos el valor de la mediana es mayor que 0,5 lo cual es un resultado esperado porque la correlación entre el tamaño y el número de naves en un planeta es razonable.

Finalmente, se muestra el análisis de la diversidad en los conjuntos de mapas utilizando la métrica de diversidad de mapas definida en la Ecuación (5.5) que calcula la distancia media entre los planetas de un mapa con respecto a todos los otros mapas del conjunto. Por lo tanto, se calcula la métrica de diversidad D_i para cada mapa del conjunto de acuerdo con la Ecuación (5.5), entonces el promedio de la diversidad para cada conjunto se denomina μ_D , y se obtiene la desviación estándar σ_d de estos valores de diversidad, como sigue:

$$\sigma_d = \sqrt{\frac{\sum_{i=1}^N (D_i - \mu_d)^2}{N}} \quad (5.13)$$

La Figura 5.10 muestra una clara distinción entre el conjunto de mapas de Google y nuestros mapas, el primero tiene menos dispersión en su distribución de valores y produce diferencias significativas con respecto al conjunto de mapas de nuestros algoritmos que son más dispersos. Este resultado puede ser un indicador de que nuestros algoritmos son capaces de mantener un nivel de diversidad adecuado en el HoF, de hecho, observe cómo la distribución de la familia “Div” tiene la mayor dispersión debido a que esta variante de algoritmo presta especial atención al control de la diversidad en el HoF durante el proceso coevolutivo.

A continuación, se muestran tres experimentos adicionales que hemos realizado, uno para mostrar la capacidad del sistema para desarrollar *bots* competitivos, otro para evaluar la adecuación de nuestro método para hacer frente a un conjunto de FRBs (por ejemplo, para especializar contenido/*bots* con respecto a varios adversarios o perfiles de jugadores) y el tercero para demostrar la adaptación de nuestros mapas



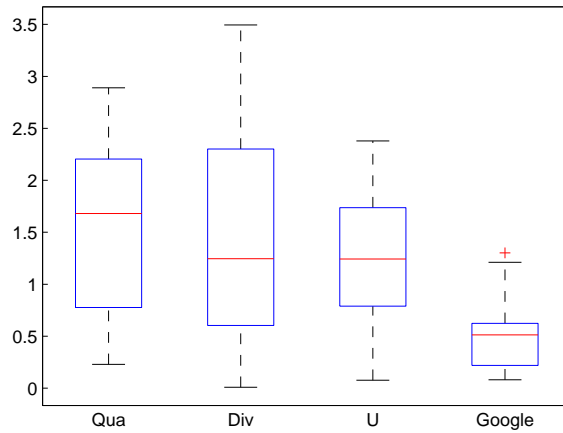


Figura 5.10: Análisis estadístico de la diversidad en el conjunto de mapas.

para favorecer a un FRB cuando se enfrenta a uno de los mejores *bot* expertos conocidos para el juego *Planet Wars*. En el primer caso, se ha seleccionado para hacer de FRB a un *bot* muy especializado, el *GeneBot* que ha sido descrito en la literatura científica (Fernández-Ares et al., 2011; Mora et al., 2012). La estrategia de juego de este *bot* se obtuvo de un proceso de optimización especializado basado en AEs.

En el segundo experimento se incluyeron tres FRBs: *GeneBot*, *ProspectorBot* y *RandomBot* (es decir, otro *bot* del conjunto original de Google para *PlanetWars*). Para el análisis de estas tres pruebas extras se consideraron todas las variantes de ‘MuCCCo’, y cada experimento se ejecutó 10 veces.

La Tabla 5.5 muestra los resultados del torneo *Todos vs. Todos* entre 15 individuos victoriosos (tomados de las ejecuciones de nuestros algoritmos) y donde se usaron dos mapas balanceados (es decir, que no favorecen a un jugador específico) del conjunto de mapas de Google que no estuvieron involucrados en el proceso coevolutivo. El prefijo ‘Many’(resp.‘Unique’) indica que la cardinalidad del conjunto de FRB es 3 (resp. 1). Tenga en cuenta que la versión original de *GeneBot* también se incluye en los torneos.

En todos los casos, los individuos obtenidos de las instancias ‘fijas’ son más robustos

CAPÍTULO 5. GENERACIÓN SIMULTÁNEA DE CONTENIDOS Y
ESTRATEGIAS PARA JUEGOS USANDO COEVOLUCIÓN

Tabla 5.5: Resultados del torneo de *Todos vs. Todos* distinguiendo entre los *bots* evolucionados con un FRB (*GeneBot*) y los de tres FRBs.

Player 1	Player2	Victories-P1	Victories-P2	Draws
ManyDiv	UniqueDiv	129	321	0
ManyDiv	UniqueQua	139	311	0
ManyDiv	UniqueU	83	367	0
ManyQua	ManyDiv	256	194	0
ManyQua	ManyU	210	240	0
ManyQua	UniqueDiv	170	280	0
ManyQua	UniqueQua	146	304	0
ManyQua	UniquedU	102	348	0
ManyU	ManyDiv	276	174	0
ManyU	ManyQua	240	210	0
ManyU	UniqueDiv	182	268	0
ManyU	UniqueQua	169	281	0
ManyU	UniqueU	111	339	0
ManyDiv	<i>GeneBot</i>	5	25	0
ManyQua	<i>GeneBot</i>	9	21	0
ManyU	<i>GeneBot</i>	15	15	0
FixedDiv	<i>GeneBot</i>	15	15	0
FixedQua	<i>GeneBot</i>	19	11	0
FixedU	<i>GeneBot</i>	25	5	0
ZerlingRush	<i>GeneBot</i>	29	1	0
ZerlingRush	FixedU	30	0	0
ZerlingRush	FixedU	0	30	0

en las batallas y muestran las mejores actuaciones contra sus adversarios. La razón podría ser que un único FRB conduce hacia una especialización más profunda de las soluciones durante el proceso de búsqueda, mientras que el uso de muchos FRB genera soluciones menos especializadas (a expensas de obtener un perfil más amplio). Observe también que los *bots* evolucionados a través de las versiones ‘Unique’ fueron capaces de superar al *GeneBot* (en especial, los de la versión ‘Unique U’). Este es un resultado prometedor, si consideramos que nuestros *bots* no fueron optimizados en los dos mapas donde se efectuaron estos enfrentamientos.

En el tercer experimento, se consideró el *bot ZerlingRush* (desarrollado por *Green-Tea*) que se clasificó en la octava posición en el concurso *Google AI Challenge 2010*



(donde compitieron más de 4600 propuestas de *bots*). Véanse en las tres últimas filas de la Tabla 5.5 los resultados que se describen a continuación.

En una primera prueba el *ZerlingRush* ganó a todos nuestros *bots* (incluyendo el *GeneBot* original) en el conjunto de mapas proporcionados por Google. Luego, fijamos *ZerlingRush* como el FRB en el algoritmo *FixedU*, y dejamos que *Genebot* co-evolucionara para vencerlo al mismo tiempo que los mapas del juego co-evolucionaron para desfavorecerlo. Y el resultado fue que el *GeneBot* co-evolucionado venció al *ZerlingRush* en todos nuestros mapas generados (no corruptos). Esta es una prueba de que los mapas se adaptan de forma exitosa para satisfacer los objetivos establecidos y a su vez, las estrategias de IA también optimizan su desempeño.

5.3. Conclusiones del capítulo

En este capítulo se ha descrito un modelo evolutivo que permite la coevolución de estrategias de IA y contenidos de juego a través del enfoque competitivo. Al tratarse de dos dominios intrínsecamente diferentes, resulta complicado establecer una relación de competición directa tal y como exige un sistema competitivo, es por eso que se decidió incorporar un tercer participante al modelo. Se trata de un conjunto de oponentes externos que son quienes desempeñan el rol de adversarios en las interacciones entre las poblaciones. Estos adversarios permitirán introducir al proceso de optimización, objetivos específicos que pueden estar basados en decisiones de diseño del juego que fomentarán la adaptación de las partidas en función de dichas decisiones.

Todos los experimentos del modelo MuCCCo se realizaron con el juego *Planet Wars*. Se pudo comprobar que además de generar mapas que favorecen a ciertos perfiles de jugadores, también se logró optimizar las estrategias de IA para que fueran capaces de vencer a oponentes expertos, incluso hubo pruebas que demostraron que las IAs generadas estaban al nivel del mejor *bot* descrito en la literatura científica para este juego.

CAPÍTULO 5. GENERACIÓN SIMULTÁNEA DE CONTENIDOS Y ESTRATEGIAS PARA JUEGOS USANDO COEVOLUCIÓN

Los experimentos aquí explicados permitieron analizar el comportamiento de las “carreras armamentistas” generadas por los algoritmos implementados, tanto sus logros como sus vulnerabilidades. En las poblaciones de *bots*, hemos visto que el algoritmo MuCCCo-Div fue el más exitoso respecto a los valores de *fitness*, seguido por MuCCCo-Qua y, contra todo pronóstico, la versión ‘U’ obtuvo los resultados más bajos. Este mismo comportamiento se obtuvo en el torneo *Todos vs. Todos*, donde una vez más, los algoritmos MuCCCo-Div y MuCCCo-Qua continuaron mostrando los mejores resultados, muy similares entre sí. También se comprobó que los algoritmos basados en el enfoque multiobjetivo (es decir, MuCCCo-U) tienen un desempeño pobre en combates directos. Creemos que este último punto puede verse influenciado por la tendencia de nuestros algoritmos a realizar iteraciones coevolutivas cortas, en los experimentos rara vez se superaban las 30 coevoluciones y el tamaño del HoF nunca alcanzó los 10 individuos. Esta situación, unida al pequeño tamaño de las poblaciones, puede haber limitado el potencial del enfoque CFS para ayudar a centrar la búsqueda hacia individuos verdaderamente robustos.

Un aspecto interesante sobre la evolución de los *bots* es que los algoritmos siempre se estancaron en un ciclo coevolutivo de la población de *bots*, es decir, en una iteración donde los *bots* deberían haber superado los resultados de la población de mapas. Recordemos que los *bots* no se enfrentan a los mapas directamente, sino que lo hacen contra un *bot* experto sobre los mapas de la población oponente. Por tanto, podemos deducir que llegado a un punto de la carrera armamentista, el *bot* fijo (con la ayuda de los mapas generados) se convierte en invencible para la población de estrategias y la búsqueda se estanca.

Para el caso de la evolución de los mapas, se observó en los experimentos que los mapas generados favorecían al Jugador 2 debido a que esa era la posición (segundo jugador) que le correspondía al *bot* experto en las partidas que se realizaban durante la fase de evaluación del ciclo coevolutivo. En realidad, este era un resultado esperado y



CAPÍTULO 5. GENERACIÓN SIMULTÁNEA DE CONTENIDOS Y ESTRATEGIAS PARA JUEGOS USANDO COEVOLUCIÓN

requiere un control por nuestra parte. Creemos que nuestros mapas especializados han propiciado partidas más interesantes y escenarios más diversos y adaptados en función de objetivos específicos. Este es precisamente uno de los objetivos que se persiguen con la Generación Procedural de Contenidos (PCG): proporcionar los medios para poder adaptar la complejidad de la partida a cada jugador, lo cual es un requisito para lograr juegos dinámicos y personalizados.



Capítulo 6

Conclusiones

Como se ha explicado en esta memoria, la Coevolución además de las muchas ventajas que ofrece para explorar espacios de búsquedas donde se generen relaciones competitivas, tiene una larga lista de patologías que afectan la calidad de los resultados finales. Los remedios para combatir estos males son objeto de investigaciones en la actualidad. Se han definido algunas pautas a seguir para evadir estos comportamientos indeseables de los modelos coevolutivos, pero aún no son suficientes. Porque cada modelo es muy específico del entorno y las condiciones donde se aplica, y además, algunas de las pautas son realmente difíciles de aplicar.

Encontrar algoritmos que pueden reducir el efecto de las patologías inherentes de los modelos coevolutivos fue uno de los focos de esta tesis. En los experimentos iniciales mostrados en esta memoria se exploró el uso del *Hall of Fame* (HoF) como método de archivo para conservar a los campeones obtenidos en cada paso coevolutivo y utilizarlos en el proceso de evaluación para guiar la búsqueda. En todos los experimentos se comprobó que las métricas definidas para optimizar el uso del HoF eran eficaces y lograban mejorar el enfoque del proceso de búsqueda, sobre todo en el caso de la métrica que potenciaba la diversidad. El estudio empírico realizado sobre el juego *Robot Wars* demostró que el conocido problema de los ciclos siguió apareciendo en el

proceso coevolutivo debido a la inexistencia de una relación de transitividad entre los campeones de la memoria.

Posteriormente, se empleó el juego *Planet Wars*, para extender el trabajo previo introduciendo cambios en el proceso coevolutivo. Este juego permitió una experimentación más profunda y por tanto el logro de resultados más consistentes, debido fundamentalmente a que es un juego RTS más fácil de usar en los experimentos pero no menos complejo. Los esfuerzos en esa segunda etapa estuvieron enfocados en modificar el mecanismo de evaluación para poder identificar mejor las soluciones que eran realmente robustas. El enfoque propuesto mantiene el uso del HoF, pero también incorpora una memoria adicional (es decir, otro archivo denominado salón de celebridades, HoC) para contener otros *bots* especializados que formarán parte del conjunto de evaluación de los individuos. Este concepto de HoC permite aumentar la presión selectiva en la búsqueda coevolutiva y propiciar la diversidad, pues los bots expertos han sido especializados en otros contextos independientes al suyo, además no tienen ninguna relación evolutiva con la población.

El siguiente paso consistió en proponer un modelo que permite la generación automática y simultánea de IA y contenidos para juegos. Este mecanismo puede ayudar a los diseñadores a crear contenidos que se adapten para alcanzar objetivos específicos de juego o para guiar la partida hacia escenarios deseados. Imaginemos, por ejemplo, un mapa/nivel que se adapta para favorecer a los jugadores novatos frente a los jugadores más experimentados de manera que todos ellos puedan disfrutar de la partida (es decir, que la experiencia de juego no resulte ni demasiado fácil para el jugador experto ni muy difícil para los jugadores experimentados).

Desde la perspectiva del diseñador, nuestro enfoque puede ser visto como un modelo para la adaptación dinámica del juego de acuerdo con el perfil de un jugador específico a través de la Generación Procedural de Contenidos (PCG). Según nuestro conocimiento, esta es la primera vez que un algoritmo de Inteligencia Artificial (en concreto un

algoritmo Coevolutivo Competitivo) se utiliza para generar de forma simultánea contenido del juego y estrategias que guíen, sin intervención de jugadores humanos y de forma automática, las decisiones a tomar por los jugadores virtuales de un videojuego. Es importante destacar que ese modelo coevolutivo ha sido diseñado con un enfoque general que lo hace aplicable a cualquier género de juego que promueva la competición entre los jugadores. Esta tesis ha mostrado la aplicación de este enfoque así como su viabilidad, usando como instancia el juego *Planet Wars*. Experimentalmente, se ha probado también que es posible generar contenido que se adapta a los objetivos específicos de diseño del juego. Respecto a las estrategias de juego generadas, se hicieron varias pruebas para constatar que eran capaces de vencer a algunos de los mejores bots descritos en la literatura científica para el juego empleado.

Como conclusión general se confirmó que la evaluación de los individuos, uno de los puntos más vulnerables del proceso coevolutivo, es la base para generar la presión competitiva de la carrera armamentista y fijar los umbrales de calidad que deben alcanzar y superar las poblaciones que compiten. Por tanto, una evaluación poco robusta, limita la capacidad del modelo de hallar soluciones verdaderamente óptimas o casi óptimas. Y es aquí cuando toca preguntarse. ¿Cómo se hace una evaluación robusta? ¿Cómo evitar que las competiciones inherentes al modelo no hagan que la búsqueda se incline hacia soluciones que por circunstancias puntuales puedan obtener un buen resultado contra un oponente, pero que en realidad no tengan una alta calidad? ¿Basta con una evaluación objetiva de un individuo en un momento puntual de la evolución o hay que aplicar un criterio también subjetivo que tenga en cuenta otros factores del entorno? Esas son algunas de las cuestiones que han motivado la investigación y han sido objeto de análisis en el transcurso de esta tesis.

Los resultados que se han obtenido pretenden aportar un granito de arena a la búsqueda de las respuestas de las preguntas que continúan vigentes en el quehacer científico de esta área. Esta tesis es un primer paso en nuestra investigación sobre la

CAPÍTULO 6. CONCLUSIONES

aplicación de la Coevolución en los videojuegos. El estudio general que se ha desarrollado es un punto de partida para un trabajo futuro que seguirá investigando en el tema de los mecanismos de evaluación y la generación de múltiples contenidos e IAs para videojuegos. Esto último encaminado hacia el objetivo de la generación (mediante PCG) de juegos completos, que incluyan incluso las reglas del juego. Y relacionado con esto se hará referencia en el Apéndice A a un juego que fue desarrollado dentro del marco de esta tesis para ser utilizado como plataforma de experimentación y facilitar la investigación de técnicas de PCG de más amplio alcance.

Bibliografía

- Adamu, K. y Phelps, S. (2010). An empirical study of collaboration methods for coevolving technical trading rules. En *Computational Intelligence (UKCI), 2010 UK Workshop on*, pp. 1–6. IEEE.
- Angeline, P. J. y Pollack, J. B. (1993). Competitive Environments Evolve Better Solutions for Complex Tasks. En Forrest, S. (ed.), *Proc. 5th International Conference on Genetic Algorithms (ICGA93)*, pp. 264–270, Urbana-Champaign, IL, USA. Morgan Kaufmann.
- Ashlock, D., Ashlock, W., Samothrakis, S., Lucas, S. M., y Lee, C. (2012). From competition to cooperation: Co-evolution in a rewards continuum. En *IEEE Conference on Computational Intelligence and Games (CIG 2012)*, pp. 33–40. IEEE.
- Ashlock, D. y McGuinness, C. (2013). Landscape automata for search based procedural content generation. En *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*, pp. 1–8.
- Au, C. y Leung, H. (2008). On the behavior of cooperative coevolution in dynamic environments. En *Evolutionary Computation. CEC 2008. IEEE Congress on*, pp. 2827–2836. IEEE.
- Avery, P. (2008). *Coevolving a computer player for resource allocation games: using*



BIBLIOGRAFÍA

- the game of Tempo as a test space*. Tesis Doctoral, School of Computer Science, University of Adelaide, Adelaide, South Australia.
- Avery, P. y Louis, S. J. (2010). Coevolving team tactics for a real-time strategy game. En *Proc. IEEE Congress on Evolutionary Computation*, pp. 1–8, Barcelona, Spain. IEEE.
- Avery, P. M. y Michalewicz, Z. (2007). Static experts and dynamic enemies in coevolutionary games. En *IEEE Congress on Evolutionary Computation*, pp. 4035–4042. IEEE.
- Bäck, T. (1996). *Evolutionary algorithms in theory and practice - evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press.
- Bäck, T., Hammel, U., y Schwefel, H. (1997). Evolutionary computation: comments on the history and current state. *IEEE Transactions on Evolutionary Computation*, 1(1):3–17.
- Bäck, T. y Hoffmeister, F. (1991). Adaptive search by evolutionary algorithms. *Models of Selforganization in Complex Systems (MOSES)*, 64:156–163.
- Barricelli, N. A. (1962). Numerical Testing of Evolution Theories. Part I: Theoretical Introduction and Basic Tests. *Acta Biotheoretica (parts I/II)*, 16.
- Bucci, A. y Pollack, J. (2003). A mathematical framework for the study of coevolution. En *Foundations of Genetic Algorithms (FOGA)*, volumen 7, pp. 221–235, United Kingdom. Morgan Kaufmann.
- Burke, E. K., Gustafson, S. M., y Kendall, G. (2004). Diversity in genetic programming: an analysis of measures and correlation with fitness. *IEEE Trans. Evolutionary Computation*, 8(1):47–62.

- Buro, M. (2002). ORTS: A Hack-Free RTS Game Environment. En Schaeffer, J. et al. (eds.), *Computers and Games*, volumen 2883 de *Lecture Notes in Computer Science*, pp. 280–291. Springer.
- Cardona, A. B., Togelius, J., y Nelson, M. J. (2013). Competitive coevolution in Ms. Pac-Man. En *Proc.IEEE Congress on Evolutionary Computation*, pp. 1403–1410, Cancun, Mexico.
- Cliff, D. y Miller, G. (1995). Tracking the red queen: Measurements of adaptive progress in coevolutionary simulations. *Advances In Artificial Life*, pp. 200–218.
- Collins, K. (2009). An Introduction to Procedural Music in Video Games. *Contemporary Music Review*, 28(1):5–15.
- Cook, M., Colton, S., y Gow, J. (2012). Initial Results from Co-operative Co-evolution for Automated Platformer Design. En c, C. D. C. (ed.), *Applications of Evolutionary Computation - EvoApplications (EvoGAMES)*, volumen 7248 de *Lecture Notes in Computer Science*, pp. 194–203. Springer.
- Corruble, V., Madeira, C. A. G., y Ramalho, G. (2002). Steps toward Building of a Good AI for Complex Wargame-Type Simulation Games. En Mehdi, Q. H. y Gough, N. E. (eds.), *3rd International Conference on Intelligent Games and Simulation (GAME-ON 2002)*, pp. 155–159, London, UK.
- Csikszentmihalyi, M. (1991). *Flow: The psychology of optimal experience*, volumen 41. HarperPerennial New York.
- Das, S., Panigrahi, B., y Pattnaik, S. (2009). Nature-inspired algorithms for multi-objective optimization. *Handbook of Research on Machine Learning Applications and Trends: Algorithms Methods and Techniques*, Hershey, New York, 1:95–108.
- Dawkins, R. y Krebs, J. R. (1979). Arms races between and within species. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 205(1161):489–511.



BIBLIOGRAFÍA

- Dayhoff, J. E. (1990). *Neural network architectures: an introduction*. Van Nostrand Reinhold Co.
- de Jong, E. (2004). Towards a bounded Pareto-Coevolution archive. En *Proc. IEEE Congress on Evolutionary Computation*, volumen 2, pp. 2341–2348, Portland, Oregon, USA.
- de Jong, E. D., Stanley, K. O., y Wiegand, R. P. (2007). Introductory tutorial on coevolution. En *Proceedings of the 2007 Conference on Genetic and Evolutionary Computation, GECCO'07*, pp. 3133–3157, New York, USA. ACM.
- DEV (2017). Libro Blanco del Desarrollo Español del Videojuego. <http://www.dev.org.es/images/stories/docs/libro%20blanco%20dev%202016.pdf>. Accedido: 2017-04-05.
- Diaz-Furlong, H. y Solis-González Cosío, A. (2013). An approach to level design using procedural content generation and difficulty curves. En *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*, pp. 1–8.
- Dorigo, M. (1992). *Optimization, learning and natural algorithms*. Tesis Doctoral, Politecnico di Milano, Italy.
- Dziuk, A. y Miikkulainen, R. (2011). Creating intelligent agents through shaping of coevolution. En *IEEE Congress on Evolutionary Computation (CEC 2011)*, pp. 1077–1083.
- Ebner, M., Watson, R. A., y Alexander, J. (2010). Coevolutionary Dynamics of Interacting Species. En Chio, C. D. et al. (eds.), *Applications of Evolutionary Computation, EvoApplicatons 2010 (EvoApplications (1))*, volumen 6024 de *Lecture Notes in Computer Science*, pp. 1–10. Springer.
- Eiben, A. E., Hinterding, R., y Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Trans. Evolutionary Computation*, 3(2):124–141.



- Eiben, A. E., Marchiori, E., y Valko, V. (2004). Evolutionary algorithms with on-the-fly population size adjustment. En *International Conference on Parallel Problem Solving from Nature*, pp. 41–50. Springer.
- Fernández-Ares, A., García-Sánchez, P., Mora, A. M., y Merelo, J. (2012). Adaptive bots for real-time strategy games via map characterization. En *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*, pp. 417–721. IEEE.
- Fernández-Ares, A., Mora, A. M., Guervós, J. J. M., García-Sánchez, P., y Fernandes, C. (2011). Optimizing player behavior in a real-time strategy game using evolutionary algorithms. En *IEEE Congress on Evolutionary Computation (CEC 2011)*, pp. 2017–2024.
- Ficici, S. (2004). *Solution concepts in coevolutionary algorithms*. Tesis Doctoral, Brandeis University, Waltham, Massachusetts.
- Ficici, S. G. y Pollack, J. B. (2003). A game theoretic memory mechanism for coevolution. En *Proceedings of the 2003 international conference on Genetic and evolutionary computation (GECCO'03): Part I*, pp. 286–297, Berlin, Heidelberg. Springer-Verlag.
- Fister Jr., I., Yang, X., Fister, I., Brest, J., y Fister, D. (2013). A Brief Review of Nature-Inspired Algorithms for Optimization. *CoRR*, abs/1307.4186.
- Floreano, D. y Nolfi, S. (1997). God save the red queen! Competition in coevolutionary robotics. En *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pp. 398–406. Morgan Kaufmann.
- Floreano, D., Nolfi, S., y Mondada, F. (1998). Competitive Co-evolutionary Robotics: From Theory to Practice. En *Proc. of the Fifth International Conference on Simulation of Adaptive Behavior on From Animals to Animats 5*, pp. 515–524, Zurich, Switzerland.



BIBLIOGRAFÍA

- Fogel, D. B. y Michalewicz, Z. (2001). Why Evolutionary Algorithms? En *Current Trends in Theoretical Computer Science*, pp. 579–602.
- Fogel, L. J., Owens, A. J., y Walsh, M. J. (1966). *Artificial Intelligence Through Simulated Evolution*. John Wiley & Sons, New York.
- Font, J., Mahlmann, T., Manrique, D., y Togelius, J. (2013). A Card Game Description Language. En Esparcia-Alcázar, A. (ed.), *Applications of Evolutionary Computation*, volumen 7835 de *Lecture Notes in Computer Science*, pp. 254–263. Springer Berlin Heidelberg.
- Frade, M., de Vega, F. F., y Cotta, C. (2008). Modelling Video Games' Landscapes by Means of Genetic Terrain Programming - A New Approach for Improving Users' Experience. En Giacobini, M. et al. (eds.), *Applications of Evolutionary Computing*, volumen 4974 de *Lecture Notes in Computer Science*, pp. 485–490, Berlin Heidelberg. Springer-Verlag.
- Frade, M., de Vega, F. F., y Cotta, C. (2009). Breeding Terrains with Genetic Terrain Programming: The Evolution of Terrain Generators. *International Journal of Computer Games Technology*, 2009.
- Frade, M., de Vega, F. F., y Cotta, C. (2010). Evolution of artificial terrains for video games based on obstacles edge length. En *IEEE Congress on Evolutionary Computation*, pp. 1–8. IEEE.
- Freeman, D. (2004). Creating Emotion in Games: The Craft and Art of Emotioneering&Trade;. *Computer Entertainment*, 2(3):15–15.
- Friedberg, R. M. (1958). A learning machine: Part I. *IBM Journal of Research and Development*, 2(1):2–13.
- García Ortega, R. H., García Sánchez, P., Mora, A., y Merelo, J. J. (2014). My life as a sim: evolving unique and engaging life stories using virtual worlds. En Sayama, H.,

-
- Rieffel, J., Risi, S., Doursat, R., y Lipson, H. (eds.), *Proceedings of the fourteenth International Conference on the Synthesis and Simulation of Living Systems*, pp. 580–587.
- Halim, Z., Baig, A. R., y Mujtaba, H. (2010). Measuring entertainment and automatic generation of entertaining games. *International Journal of Information Technology, Communications and Convergence*, 1(1):92–107.
- Hendriks, M., Meijer, S., Van Der Velden, J., y Iosup, A. (2013). Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications and Applications*, 9(1):1:1–1:22.
- Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor: Institute for Social Research, The University of Michigan.
- Jaskowski, W. y Krawiec, K. (2010). Coordinate System Archive for coevolution. En *IEEE Congress on Evolutionary Computation CEC 2010*, pp. 1–10. IEEE.
- Johansson, A. y Dell'Acqua, P. (2012). Emotional behavior trees. En *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*, pp. 355–362. IEEE.
- Johnson, R., Melich, M., Michalewicz, Z., y Schmidt, M. (2004). Coevolutionary Tempo game. En *Proc. Congress on Evolutionary Computation*, volumen 2, pp. 1610–1617, Portland, OR, USA. IEEE.
- Jong, E. y Pollack, J. (2004). Ideal evaluation from coevolution. *Evolutionary Computation*, 12(2):159–192.
- Jousselin, E., Desdevises, Y., y Coeur, A. (2009). Fine scale cospeciation between *Brachycaudus* and *Buchnera aphidicola*: bacterial genome helps define species and evolutionary relationships in aphids. *Proceedings of the Royal Society B: Biological Sciences*, 276(1654):187.



BIBLIOGRAFÍA

- Juile, H. y Pollack, J. B. (1996). Dynamics of coevolutionary learning. En *Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, pp. 526–534. MIT Press.
- Kennedy, J. y Eberhart, R. (1995). Particle swarm optimization. En *IEEE International Conference on Neural Networks*, volumen 4, pp. 1942–1948. IEEE.
- Koza, J. R. (1993). *Genetic programming - on the programming of computers by means of natural selection*. Complex adaptive systems. MIT Press.
- Kruskal, W. y Wallis, W. (1952). Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621.
- Lara-Cabrera, R., Cotta, C., y Fernández-Leiva, A. J. (2013a). Evolving Aesthetic Maps for a Real Time Strategy Game. En *Proc. of the First Spanish Symposium on Entertainment Computing*, Madrid, Spain.
- Lara-Cabrera, R., Cotta, C., y Fernández-Leiva, A. J. (2013b). A Procedural Balanced Map Generator with Self-adaptive Complexity for the Real-Time Strategy Game Planet Wars. En Esparcia-Alcázar, A. I. (ed.), *EvoApplications*, volumen 7835 de *Lecture Notes in Computer Science*, pp. 274–283. Springer.
- Lara-Cabrera, R., Cotta, C., y Fernández-Leiva, A. J. (2013c). A review of computational intelligence in RTS games. En *IEEE Symposium on Foundations of Computational Intelligence, FOCI 2013*, pp. 114–121.
- Lara-Cabrera, R., Cotta, C., y Fernández-Leiva, A. (2014). An analysis of the structure and evolution of the scientific collaboration network of computer intelligence in games. *Physica A: Statistical Mechanics and its Applications*, 395:523 – 536.
- Lehman, J. y Stanley, K. O. (2011). Abandoning Objectives: Evolution Through the Search for Novelty Alone. *Evol. Comput.*, 19(2):189–223.



- Leung, Y., Gao, Y., y Xu, Z. (1997). Degree of population diversity - a perspective on premature convergence in genetic algorithms and its Markov chain analysis. *IEEE Trans. Neural Networks*, 8(5):1165–1176.
- Liapis, A., Martinez, H., Togelius, J., y Yannakakis, G. (2013a). Adaptive game level creation through rank-based interactive evolution. En *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*, pp. 1–8.
- Liapis, A., Yannakakis, G., y Togelius, J. (2013b). Enhancements to constrained novelty search: two-population novelty search for generating game content. En Blum, C. y Alba, E. (eds.), *Genetic and Evolutionary Computation Conference*, pp. 343–350, Amsterdam, The Netherlands. ACM.
- Lichocki, P. (2008). Evolving players for a real-time strategy game using gene expression programming. Tesis de licenciatura, Poznan University of Technology, Poznan, Poland.
- Lidén, L. (2003). Artificial stupidity: The art of intentional mistakes. *AI Game Programming Wisdom*, 2:41–48.
- Livingstone, D. (2005). Coevolution in Hierarchical AI for Strategy Games. En *IEEE Conference on Computational Intelligence and Games (CIG 2005)*. IEEE.
- Lobo, F. G. y Goldberg, D. E. (2004). The parameter-less genetic algorithm in practice. *Inf. Sci.*, 167(1-4):217–232.
- Lobo, F. G. y Lima, C. F. (2005). A review of adaptive population sizing schemes in genetic algorithms. En *Genetic and Evolutionary Computation Conference, GECCO 2005, Workshop Proceedings, Washington DC, USA, June 25-26, 2005*, pp. 228–234. ACM.
- Lozano, M., Herrera, F., y Cano, J. R. (2008). Replacement strategies to preserve useful



BIBLIOGRAFÍA

- diversity in steady-state genetic algorithms. *Information Sciences*, 178(23):4421–4433.
- Lucas, S. M. (2009). Computational Intelligence and AI in Games: A New IEEE Transactions. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(1):1–3.
- Lucas, S. M., Mateas, M., Preuss, M., Spronck, P., y Togelius, J. (2015). Artificial and Computational Intelligence in Games: Integration (Dagstuhl Seminar 15051). *Dagstuhl Reports*, 5(1):207–242.
- Mahlmann, T., Togelius, J., y Yannakakis, G. N. (2012). Spicing Up Map Generation. En Chio, C. D. et al. (eds.), *Applications of Evolutionary Computation*, volumen 7248 de *Lecture Notes in Computer Science*, pp. 224–233, Málaga, Spain. Springer-Verlag.
- Malone, T. W. (1980). What makes things fun to learn? Heuristics for designing instructional computer games. En *Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small systems*, pp. 162–169. ACM.
- Marquez, L., Redman, R., Rodriguez, R., y Roossinck, M. (2007). A virus in a fungus in a plant: three way symbiosis required for thermal tolerance. *Science*, 315(5811):513.
- Michalewicz, Z. y Fogel, D. B. (2004). *How to solve it - modern heuristics: second, revised and extended edition (2. ed.)*. Springer.
- Miconi, T. (2009). Why Coevolution Doesn't "Work": Superiority and Progress in Coevolution. En Vanneschi, L. et al. (eds.), *Proc. Genetic Programming, 12th European Conference (EuroGP 2009)*, LNCS, pp. 49–60, Tübingen, Germany. Springer.



- Miconi, T. y Channon, A. (2006). Analysing Coevolution Among Artificial 3D Creatures. En *Artificial Evolution*, volumen 3871, pp. 167–178, Birmingham UK. Springer Berlin/Heidelberg.
- Miles, C. y Louis, S. J. (2006). Co-evolving real-time strategy game playing influence map trees with genetic algorithms. En *Proc. International Congress on Evolutionary Computation*, Vancouver. IEEE.
- Mora, A. M., Fernández-Ares, A., Guervós, J. J. M., García-Sánchez, P., y Fernandes, C. M. (2012). Effect of Noisy Fitness in Real-Time Strategy Games Player Behaviour Optimisation Using Evolutionary Algorithms. *Journal of Computer Science and Technology*, 27(5):1007–1023.
- Moran, N., Nakabachi, A., y McCutcheon, J. (2008). Genomics and evolution of heritable bacterial symbionts. *Annual Review of Genetics*, 42:165–190.
- Nash, J. (1951). Non-cooperative games. *Annals of mathematics*, pp. 286–295.
- Nerome, M., Yamada, K., Endo, S., y Miyagi, H. (1998). Competitive co-evolution based game-strategy acquisition with the packaging. En Jain, L. C. y Jain, R. K. (eds.), *Knowledge-Based Intelligent Electronic Systems, 2nd International Conference (KES (3))*, pp. 184–189. IEEE.
- Nogueira, M., Cotta, C., y Fernández-Leiva, A. J. (2012a). On Modeling, Evaluating and Increasing Players' Satisfaction Quantitatively: Steps towards a Taxonomy. En Chio, C. D. et al. (eds.), *Applications of Evolutionary Computation - EvoGAMES*, volumen 7248 de *Lecture Notes in Computer Science*, pp. 245–254. Springer.
- Nogueira, M., Cotta, C., y Fernández-Leiva, A. J. (2013). An Analysis of Hall-of-Fame Strategies in Competitive Coevolutionary Algorithms for Self-Learning in RTS Games. En Nicosia, G. y Pardalos, P. M. (eds.), *Learning and Intelligent*



BIBLIOGRAFÍA

- Optimization - 7th International Conference, LION 7, Catania, Italy, January 7-11, 2013, Revised Selected Papers*, volumen 7997 de *Lecture Notes in Computer Science*, pp. 174–188. Springer.
- Nogueira, M., Cotta, C., y Fernández-Leiva, A. J. (2014a). Virtual player design using self-learning via competitive coevolutionary algorithms. *Natural Computing*, 13(2):131–144.
- Nogueira, M., Cotta, C., y Fernández-Leiva, A. J. (2016). Competitive Algorithms for Coevolving Both Game Content and AI. A Case Study: Planet Wars. *IEEE Transactions on Computational Intelligence and AI in Games*, 8(4):325–337.
- Nogueira, M., Fernández-Leiva, A. J., y Cotta, C. (2014b). Eryna: una herramienta de apoyo a la revolución de los videojuegos. En *Proceedings 1st Congreso de la Sociedad Española para las Ciencias del Videojuego, CoSECivi 2014, Barcelona, Spain, June 24, 2014.*, pp. 173–184.
- Nogueira, M., Gálvez, J., Cotta, C., y Fernández-Leiva, A. J. (2012b). Hall of Fame based competitive coevolutionary algorithms for optimizing opponent strategies in a new RTS game. En Fernández-Leiva and others, A. (ed.), *13th annual European conference on simulation and AI in computer games (GAME-ON 2012)*, pp. 71–78, Málaga, Spain. Eurosis,.
- Nolfi, S. y Floreano, D. (1998). Coevolving Predator and Prey Robots: Do “Arms Races” Arise in Artificial Evolution? *Artificial Life*, 4(4):311–335.
- Onuczko, C., Szafron, D., Schaeffer, J., Cutumisu, M., Siegel, J., Waugh, K., y Schumacher, A. (2006). Automatic Story Generation for Computer Role-Playing Games. En Laird, J. E. y Schaeffer, J. (eds.), *Proceedings of the Second Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE2006)*, pp. 147–148.

-
- Oyama, K. (1986). La Coevolución. *Ciencias, Facultad de Ciencias, Universidad Nacional Autónoma de México*, 2(001):64–73.
- Pelikan, M., Goldberg, D. E., y Cantú-Paz, E. (2000). Bayesian Optimization Algorithm, Population Sizing, and Time to Convergence. En *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '00), Las Vegas, Nevada, USA, July 8-12, 2000*, pp. 275–282. Morgan Kaufmann.
- Peña, L., Ossowski, S., Peña, J. M., y Sánchez, J. Á. (2011a). EEP - A lightweight emotional model: Application to RPG video game characters. En Cho, S., Lucas, S. M., y Hingston, P. (eds.), *2011 IEEE Conference on Computational Intelligence and Games, CIG 2011, Seoul, South Korea, August 31 - September 3, 2011*, pp. 142–149. IEEE.
- Peña, L., Peña, J. M., y Ossowski, S. (2011b). Representing Emotion and Mood States for Virtual Agents. En Klügl, F. y Ossowski, S. (eds.), *Multiagent System Technologies - 9th German Conference, MATES 2011, Berlin, Germany, October 6-7, 2011. Proceedings*, volumen 6973 de *Lecture Notes in Computer Science*, pp. 181–188. Springer.
- Picard, R. W. (1997). *Affective computing*. MIT Press, MIT Media Laboratory, Cambridge, MA.
- Polceanu, M., García, A. M., Jiménez, J. L., Buche, C., y Leiva, A. J. F. (2016). The Believability Gene in Virtual Bots. En *Proceedings of the Twenty-Ninth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2016, Key Largo, Florida*, pp. 346–349. AAAI Press.
- Preuss, M., Beume, N., Danielsiek, H., Hein, T., Naujoks, B., Piatkowski, N., Stür, R., Thom, A., y Wessing, S. (2010). Towards Intelligent Team Composition and



BIBLIOGRAFÍA

- Maneuvering in Real-Time Strategy Games. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(2):82–98.
- Quadflieg, J., Preuss, M., Kramer, O., y Rudolph, G. (2010). Learning the track and planning ahead in a car racing controller. En *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, pp. 395–402. IEEE.
- Rechenberg, I. (1984). The evolution strategy, a mathematical model of darwinian evolution. En *Synergetics from microscopic to macroscopic order*, pp. 122–132. Springer.
- Reynolds, C. (1994). Competition, coevolution and the game of tag. En Brooks y Maes, P. (eds.), *Proceedings of Artificial Life IV*, pp. 59–69, Cambridge, Massachusetts. MIT Press.
- Rose, K. D. (1990). Arguments on Evolution. A Paleontologist’s Perspective. *BioScience*, 40(4):312–314.
- Rosin, C. y Belew, R. (1997a). New methods for competitive coevolution. *Evolutionary Computation*, 5(1):1–29.
- Rosin, C. D. y Belew, R. K. (1997b). *Coevolutionary search among adversaries*. Tesis Doctoral, University of California at San Diego, La Jolla, CA, USA.
- Ruiz-Moyano, A., Nogueira, M., y Fernández-Leiva, A. J. (2016). Hacia la Generación Automática de Mecánicas de Juego: un Editor de Reglas para Eryna. En *Proceedings of the 3rd Congreso de la Sociedad Española para las Ciencias del Videojuego (CoSeCiVi 2016), Barcelona, Spain, June 29, 2016.*, pp. 125–134.
- Samothrakis, S., Lucas, S. M., Runarsson, T. P., y Robles, D. (2013). Coevolving Game-Playing Agents: Measuring Performance and Intransitivities. *IEEE Transactions on Evolutionary Computation*, 17(2):213–226.

- Sarma, J. y Jong, K. A. D. (1997). An Analysis of Local Selection Algorithms in a Spatially Structured Evolutionary Algorithm. En Bäck, T. (ed.), *Proceedings of the 7th International Conference on Genetic Algorithms, East Lansing, MI, USA, July 19-23, 1997*, pp. 181–187. Morgan Kaufmann.
- Scales, D. y Thompson, T. (2014). SpelunkBots API-An AI Toolset for Spelunky. En *Computational Intelligence and Games (CIG), 2014 IEEE Conference on*, pp. 1–8. IEEE.
- Sims, K. (1994). Evolving 3D Morphology and Behavior by Competition. *Artificial Life*, 1(4):353–372.
- Smith, G., Avery, P., Houmanfar, R., y Louis, S. J. (2010). Using co-evolved RTS opponents to teach spatial tactics. En Yannakakis, G. N. y Togelius, J. (eds.), *IEEE Conference on Computational Intelligence and Games (CIG 2010)*, pp. 146–153. IEEE.
- Sokal, R. y Rohlf, J. (1995). *Biometry: the principles and practice of statistics in biological reseach*. W.H. Freeman and Company, New York, USA.
- Spronck, P. (2005). *Adaptive Game AI*. Tesis Doctoral, Universiteit Maastricht (IKAT), The Netherlands.
- Stanley, K. O., Bryant, B. D., y Miikkulainen, R. (2005). Real-time neuroevolution in the NERO video game. *IEEE Transactions on Evolutionary Computation*, 9(6):653–668.
- Stanley, K. O. y Miikkulainen, R. (2002). The Dominance Tournament Method of Monitoring Progress in Coevolution. En *Genetic and Evolutionary Computation Conference, GECCO'02*. Morgan Kaufmann.
- Sweetser, P. y Wyeth, P. (2005). GameFlow: a model for evaluating player enjoyment in games. *Computers in Entertainment (CIE)*, 3(3):3–3.



BIBLIOGRAFÍA

- Sykes, J. y Brown, S. (2003). Affective Gaming: Measuring Emotion Through the Gamepad. En *CHI '03 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '03, pp. 732–733, New York, NY, USA. ACM.
- Thompson, J. (2005). *The geographic mosaic of coevolution*. University of Chicago Press, Chicago.
- Thompson, J. (2010). Four central points about coevolution. *Evolution: Education and Outreach*, 3(1):7–13.
- Togelius, J., Burrow, P., y Lucas, S. M. (2007a). Multi-population competitive coevolution of car racing controllers. En *Proc. IEEE Congress on Evolutionary Computation*, pp. 4043–4050, Singapore. IEEE.
- Togelius, J., Champanand, A. J., Lanzi, P. L., Mateas, M., Paiva, A., P., M., y Stanley, K. O. (2013). Procedural Content Generation: Goals, Challenges and Actionable Steps. En *Artificial and Computational Intelligence in Games*, volumen 6 de *Dagstuhl Follow-Ups*, pp. 61–75. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.
- Togelius, J., De Nardi, R., y Lucas, S. (2007b). Towards automatic personalised content creation for racing games. En *Computational Intelligence and Games, 2007. CIG 2007. IEEE Symposium on*, pp. 252–259.
- Togelius, J., Yannakakis, G. N., Stanley, K. O., y Browne, C. (2011). Search-Based Procedural Content Generation: A Taxonomy and Survey. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):172–186.
- Torres, E. (2009). *Ciento Cincuenta años de pensamiento coevolutivo: La vida es una maraña de interacciones*. Tesis Doctoral, Universidad Nacional de Colombia, Bogotá, Colombia.
- Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, 59(236):433–460.

- Ueda, S., Quek, S., Itioka, T., Inamori, K., Sato, Y., Murase, K., y Itino, T. (2008). An ancient tripartite symbiosis of plants, ants and scale insects. *Proceedings of the Royal Society B: Biological Sciences*, 275(1649):2319.
- Warman, P. (2016). 2016 Global Games Market Report. Informe técnico núm., Newzoo. Accedido el 20 Marzo 2016.
- Watson, R., Pollack, J., et al. (2001). Coevolutionary dynamics in a minimal substrate. En *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO'01*, pp. 702–709, United Kingdom. Morgan Kaufmann.
- Weber, B. G. y Mateas, M. (2009). A data mining approach to strategy prediction. En Lanzi, P. L. (ed.), *IEEE Symposium on Computational Intelligence and Games*, pp. 140–147. IEEE.
- Wiegand, R., Liles, W., y De Jong, K. (2001). An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. En *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO'01*, pp. 1235–1242. Morgan Kaufmann.
- Yang, X.-S. y Deb, S. (2009). Cuckoo search via Lévy flights. En *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pp. 210–214. IEEE.
- Yannakakis, G. (2012). Game AI revisited. En *Proc. Computing Frontiers Conference*, pp. 285–292, Caligari, Italy. ACM.
- Yannakakis, G. y Hallam, J. (2009). Real-time game adaptation for optimizing player satisfaction. *Computational Intelligence and AI in Games, IEEE Transactions on*, 1(2):121–133.
- Yannakakis, G. N. (2008). How to model and augment player satisfaction: a review. En Berkling, K., Giuliani, D., y Potamianos, A. (eds.), *Proceedings of the 1st Workshop on Child, Computer and Interaction, WOCCI'08*, pp. 21.



BIBLIOGRAFÍA

- Yannakakis, G. N. y Hallam, J. (2006). Towards capturing and enhancing entertainment in computer games. En *Advances in artificial intelligence*, pp. 432–442. Springer.
- Yannakakis, G. N. y Hallam, J. (2007). Towards optimizing entertainment in computer games. *Applied Artificial Intelligence*, 21(10):933–971.
- Yannakakis, G. N. y Togelius, J. (2015). A Panorama of Artificial and Computational Intelligence in Games. *IEEE Trans. Comput. Intellig. and AI in Games*, 7(4):317–335.
- Zitzler, E. y Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Trans. Evolutionary Computation*, 3(4):257–271.



Apéndices



UNIVERSIDAD
DE MÁLAGA

Apéndice A

El juego *Eryna*

Varias plataformas de prueba han sido implementadas para apoyar el trabajo de los investigadores en el campo de los videojuegos, la mayoría están enfocadas a la evaluación de mecanismos de IA para optimizar el comportamientos de los *bots*. *Eryna* (Nogueira et al., 2014b) es una herramienta de apoyo a la investigación, similar a las que se han citado en el Capítulo 3, con la distinción de que se han integrado en ella módulos que facilitan la exploración de algoritmos relacionados con las nuevas tendencias de la IA aplicada a videojuegos.

Eryna ha sido desarrollado en el marco de esta tesis, con la colaboración de la empresa *A Bonfire of Souls*¹ que se ha encargado de la visualización gráfica. Hasta la fecha, este juego ha servido de marco experimental para el trabajo publicado en (Ruiz-Moyano et al., 2016), enfocado en la generación y optimización de las reglas de *Eryna* mediante un Algoritmo Genético (AG). En el futuro próximo será usado como marco experimental en varias de nuestras futuras líneas de trabajo encaminadas hacia la aplicación de técnicas de PCG que permitan acercarnos al objetivo de generar videojuegos completos.

Eryna pretende facilitar la tarea a los investigadores para que no tengan que lidiar

¹<http://abonfireofsouls.com/>

con problemas comunes como por ejemplo la ausencia de un mecanismo que guarde los *logs* para que sean fáciles de procesar durante la fase de análisis de los resultados. También ofrece módulos ya predefinidos que permiten incorporar algoritmos bioinspirados para aplicarlos al juego. En lo que sigue se explicará la arquitectura de esta herramienta y se darán detalles de su funcionamiento.

Eryna es un juego RTS multijugador donde cada jugador virtual compite por ser el único ganador. Está inspirado en el juego *Planet Wars*, podemos considerarlo una versión extendida de ese juego, a la cual se le han incorporado cambios en el entorno del juego y además *Eryna* permite partidas *multijugador*. La mecánica del juego es la de un clásico juego en un entorno espacial en la que los usuarios conquistan mundos. Pero en este caso la lógica del juego se complica un poco (respecto al *Planet Wars*) pues habrá que manejar un conjunto de recursos como sucede típicamente en los juegos RTS. Cada jugador contará con un grupo de unidades y recursos que deberá ir repartiendo por diferentes mundos para así ir aumentando sus conquistas y poderío. El motor lógico del juego desencadena automáticamente un grupo de “reglas evolutivas” que se encargan de controlar la producción/variación de los recursos naturales en el entorno. Las posibles acciones que puede ejecutar un jugador durante la partida son: enviar unidades entre mundos; y asignar misiones en un mundo para apoyar determinado objetivo que desee el jugador (por ejemplo: construir armas, producir alimentos).

La parametrización es una palabra de orden en este juego, permitirá al investigador ajustar varias opciones para crear diferentes niveles de dificultad en las partidas. Las reglas del juego han sido definidas en función de varios parámetros que también podrán ser configurados por el usuario para generar partidas muy distintas incluso podrían ser generadas mediante un mecanismo de PCG. Los mapas se han definido como ficheros de texto con una nomenclatura simple y fácil de generar, para facilitar el empleo de técnicas de PCG.

Descripción lógica del sistema

A continuación se presenta la vista lógica de la arquitectura de *Eryna*, véanse los componentes principales del sistema y su interacción. Se muestran dos diagramas que han sido distinguidos entre sí por las funcionalidades que le brindan al usuario, el primero de ellos (Figura A.1) agrupa los principales módulos de *Eryna* como *plataforma interactiva de prueba y optimización*, y el segundo (Figura A.2) muestra la arquitectura de *Eryna* como *entorno de competición de IA online*.

La Figura A.1 ofrece la primera vista del sistema, entre sus componentes se puede ver una versión no gráfica del juego (en la figura se nombra “Game_Engine_Local”) que se usará para lanzar partidas mono o multijugador entre los *Bots* que han sido implementados por un usuario (investigador) usando para ello un módulo de IA básico (“IA_Engine”) que proporcionará la herramienta. Esta versión no jugable, es la base de este módulo, y será usada como plataforma de prueba para evaluar el desempeño de dichos *bots* y de otros recursos suministrados. También se integrará un subsistema de Optimización Interactiva cuyo objetivo será el de optimizar el funcionamiento de modelos bioinspirados que sean suministrados por el usuario y que tendrán como objetivo la generación de soluciones subjetivamente interesantes para el juego base. Los modelos afectivos también tendrán su espacio, pues habrá un módulo que se encargará de evaluar (basado en alguno de los modelos definidos para esto en la literatura) el comportamiento emocional que genera en el juego determinado recurso proporcionado por el usuario.

Por último, es importante mencionar que siempre que el investigador emplee la herramienta como plataforma de prueba y evaluación le serán emitidos ficheros de logs con la información sobre el funcionamiento de su recurso que le permitirán hacer un análisis serio de los resultados a posteriori. El objetivo general de este módulo es permitir a los investigadores que prueben u optimicen recursos del juego que hayan sido creados por algoritmos especializados para eso.

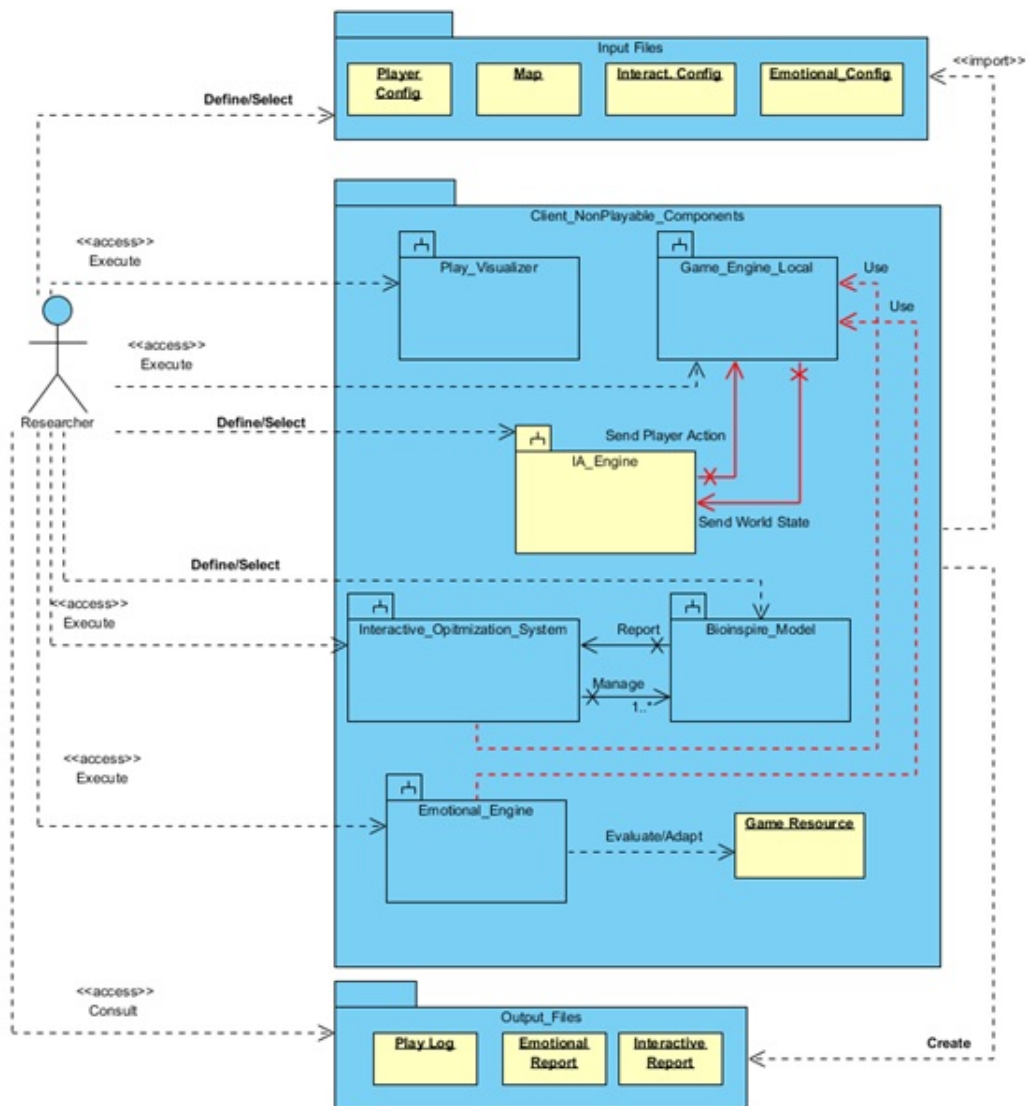


Figura A.1: *Eryna*: Vista lógica de la plataforma interactiva de prueba y optimización

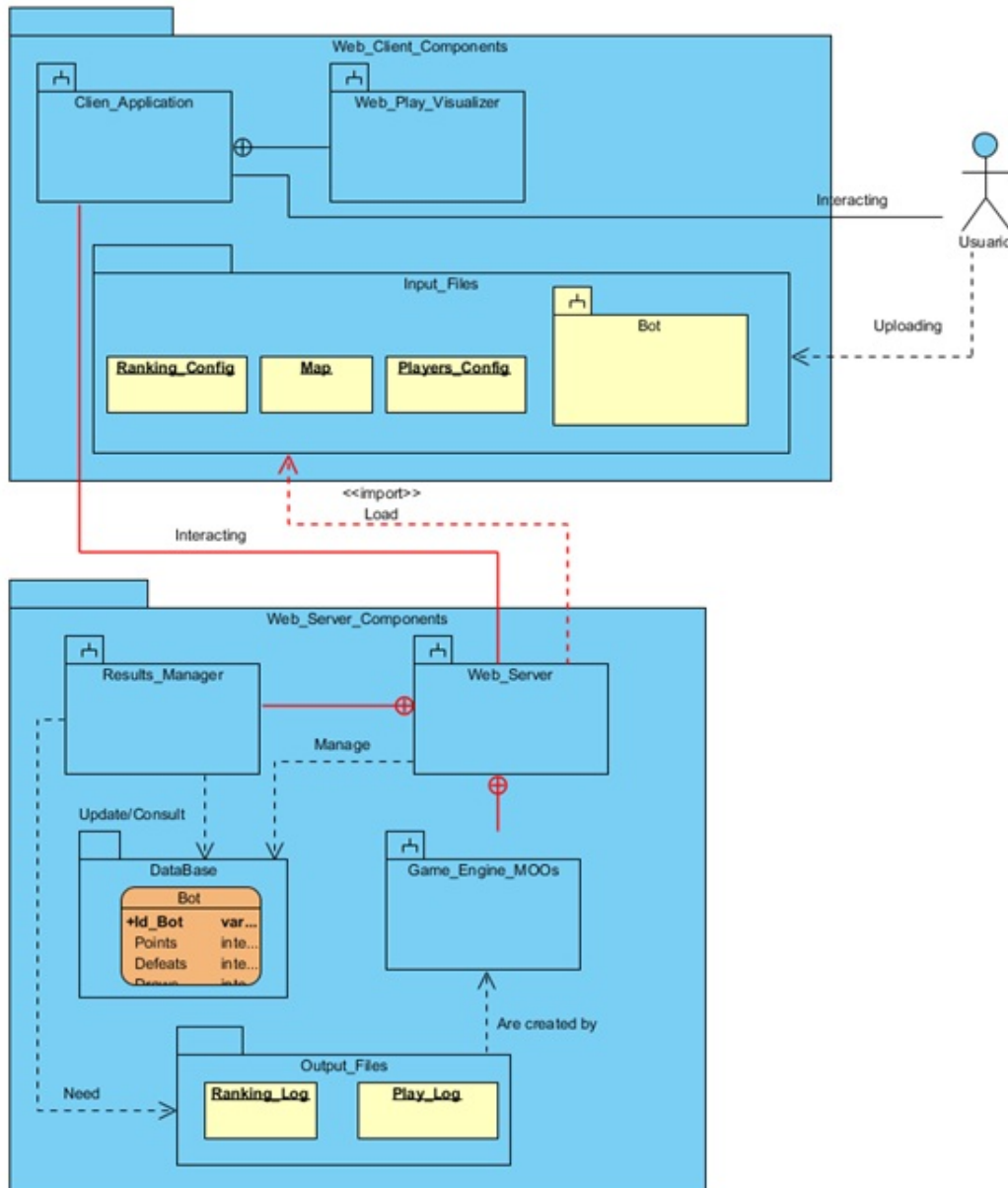


Figura A.2: *Eryna*: Vista lógica del entorno de competición online

En la Figura A.2 se muestra la segunda vista, que consta de una aplicación web donde estará accesible toda la información y los archivos necesarios para el uso de la herramienta. La web también dará acceso directo al juego RTS, los usuarios podrán subir *bots* al servidor y lanzar una partida donde estos jugadores virtuales compitan

APÉNDICE A. EL JUEGO *ERYNA*

entre ellos. También estará disponible un ranking general que contenga la posición de los *bots* que son evaluados usando este servicio.

