Aprendizaje de estrategias de decisión utilizando redes neuronales artificiales en juegos repetitivos no cooperativos en el ámbito de la economía evolucionista

Fabián Andrés Giraldo Giraldo
Ingeniería de Sistemas y Computación, MSc.(c)
Código: 02300383

Universidad Nacional de Colombia
Facultad de Ingeniería
Departamento de Sistemas e Industrial
Bogotá, D.C.
Junio de 2013

Aprendizaje de estrategias de decisión utilizando redes neuronales artificiales en juegos repetitivos no cooperativos en el ámbito de la economía evolucionista

Fabián Andrés Giraldo Giraldo
Ingeniería de Sistemas y Computación, MSc.(c)
Código: 02300383

Disertación presentada para optar al título de
Msc En sistemas y computación

Director
Jonatan Gómez Perdomo, Ph.D.
Matemáticas con concentración en Computer Science

Línea de investigación
Ecología, Sociedad y Cultura Artificial

Grupo de investigación
Grupo de Investigación en Vida Artificial - Alife

Universidad Nacional de Colombia
Facultad de Ingeniería
Departamento de Sistemas e Industrial
Bogotá, D.C.
Junio de 2013

Título en español

Aprendizaje de estrategias de decisión utilizando redes neuronales artificiales en juegos repetitivos no cooperativos en el ámbito de la economía evolucionista.

Title in English

Learning for Decision Making Strategies using Artificial Neural Networks in Repetitive, Non-Cooperative Games in the Field of Evolutionary Economics.

Resumen: El presente proyecto de investigación muestra el aprendizaje de estrategias de decisión utilizando redes neuronales artificiales en juegos repetitivos no cooperativos, específicamente, se modelaron los juegos no cooperativos: dilema del prisionero, juego de la gallina y caza del ciervo.
En la configuración de los juegos se presentan varios escenarios a saber: competencias entre agentes cuyos programas corresponde con estrategias de juegos usadas en competencias de juego no cooperativos, competencia entre agentes cuyo programa corresponde con una red neuronal obtenida a través de procesos de neuroevolución, y por último, competencia entre agentes cuyo programa corresponde con redes neuronales que se adaptan en línea.
Con fin de tener un esquema de especificación unificado, adicionalmente, se planteó el desarrollo de un laboratorio computacional en el ámbito de la economía computacional basado en agentes, dicho laboratorio permite la especificación de modelos de simulación usando un lenguaje desarrollado denominado UNALCOL. El lenguaje tiene una serie de características entre las cuales se encuentran: un entorno integrado de desarrollo que facilita las tareas de programación y una plataforma de simulación para los modelos especificados.
Un elemento importante de dicho lenguaje es que permite la integración con librerías externas para soportar el proceso de toma de decisiones.
Los resultados del proceso de investigación indican que pueden ser especificados juegos no cooperativos en UNALCOL, lo anterior, dado el correcto funcionamiento de las simulaciones realizadas con los juegos dilema del prisionero, juego de la gallina y caza del ciervo.
Adicionalmente, el proceso de evolución de las redes neuronales (perceptron multicapa, red de base radial) desarrollado con el fin de adaptar estrategias de aprendizaje en los agentes cuando compiten en los juegos no cooperativos, son comparables a los resultados obtenidos en la literatura, usando algoritmos genéticos y enjambres de partículas.
Por último, el proceso de evolución de estrategias en línea, basado en redes neuronales, integrado a los agentes cuando compite con otros contrincantes garantiza el cambio de la estrategia de juego con el fin de maximizar el puntaje obtenido.

Abstract: This research project studies the learning of decision making strategies using artificial neural networks in Repetitive, Non-Cooperative games. In this particular case, the following non-cooperative games were modeled: Prisoner's Dilemma, Chicken Game and Stag Hunt.
In each game setup the following scenarios can be seen: competition between agents whose programming corresponds to a Neural Network obtained through Neuroevolution procedures and also, competition between agents whose programming corresponds to Neural Networks which adapt online.
In order to obtain a unified specification diagram, development of a computational labo-

ratory dealing with agent based computational economy was proposed. The experiments performed through this laboratory will allow specification of simulation models using a previously developed language called UNALCOL. This language has the following characteristics: an integrated development environment which facilitates programming tasks, and a simulation platform for specified models. An important characteristic of this language is that it allows integration with external libraries to support the decision making process.
The research process' results indicate that Non-cooperative games can be specified in UNALCOL as long as the simulations made with Prisoner's Dilemma, Chicken Game and Stag Hunt are functioning properly. Additionally, the neural network evolutionary process (Multilayered perceptron, radial basis network) developed in order to adapt learning strategies in the agents when they compete in Non-cooperative games is compatible with the results obtained in textbooks using genetic algorithms and particle swarms. Finally, the evolutionary process of online strategies based on Neural Networks, integrated to agents when they compete against each other guarantees game strategy changes in order to maximize the final score.

Palabras clave: algoritmos genéticos, caza del ciervo, dilema prisionero, juego de la gallina, teoría de juegos, neuroevolución, perceptron, red base radial, red neuronal.

Keywords: genetic algorithms, stag hunt, prisoner dilemma, chicken game, game theory, neuroevolution, perceptron, radial basis network, neural network.

Nota de aceptación

Trabajo de tesis
Aprobada

"Mención Por asignar"

Jurado
ph.D Ivan Dario Hernandez

Jurado
ph.D Yoan Pinzón

Director
ph.D Jonatan Gómez Perdomo

Bogotá, D.C., Diciembre de 2013

Dedicado a

A mis padres por su constante apoyo, su dedicación y compromiso buscando mi bienestar.

A mi Hermano que es un Bacán, que lo admiro mucho.

A mi compañera de viaje Isabel Londoño, la cual me ha demostrado su amor y apoyo durante este largo proceso.

A Sebastián Leyva, que a pesar que los paisas no le caen bien, demuestra el compañerismo y el cariño de las personas de Bogotá.

A Diana Cely, por su ayuda en el desarrollo de este trabajo y amistad.

Agradecimientos

A los docentes de la Maestría en Ingeniería en Sistemas y Computación, especialmente a los Profesores Jonatan Gómez y Elizabeth León, los cuales se convierten en un ejemplo por seguir, dada su motivación por compartir toda su experiencia y conocimiento a la Universidad Nacional y por ende a los estudiantes del Programa de Ingeniería de Sistemas.

También agradecer a la Facultad de Ingeniería de la Fundación Universitaria San Martín sede Bogotá, por la colaboración en tiempo para finalizar con mis estudios; especialmente al director del Programa Andrés Felipe Solarte.

Al grupo de Investigación ALIFE por sus comentarios en el seminario, con el fin de fortalecer el trabajo de investigación.

A Bogotá, por ser una ciudad que me acogió y me brindó tantas oportunidades en mi vida.

Índice general

Índice de tablas

Índice de figuras

Introducción

La ciencia de la complejidad ha sido reconocida por diferentes académicos en importantes disciplinas científicas como la física, la economía, las ciencias de la computación y las ciencias sociales, por su capacidad para modelar sistemas no lineales. Dicha ciencia, estudia los sistemas como un conjunto de componentes interconectados cuyo comportamiento no es explicable exclusivamente por las propiedades de los mismos, más bien el comportamiento emerge de los componentes interconectados. Entre los temas de investigación de la ciencia de la complejidad, se encuentra la economía evolucionista.

La economía evolucionista es un área del pensamiento económico que rechaza los supuestos tradicionales, encargados de indicar que la economía es un sistema cerrado que eventualmente logra un estado de equilibrio, en donde supuestos de racionalidad de perfecta, inversiones homogéneas, entre otros, deben hacerse para permitir el análisis matemático [31]. En su lugar, esta área del pensamiento, ve la economía como un sistema complejo adaptativo y dinámico que trata de comprender el comportamiento emergente del sistema macroscópico, a partir de la dinámica microscópica de cada agente (humanos, firmas, mercado, etc.) [32], [63]. real.
A partir de lo anterior, Kurt Dopfer, John Foster y Jason Potts en [23], [24], [25] , [46] desarrollan un entorno de trabajo analítico en el ámbito de la economía evolutiva con una arquitectura Micro, Meso y Macro.
El resultado final que proponen, es un marco de trabajo para el análisis ontológico de la evolución económica, en el cual los cambios en el meso dominio conllevan a entender los procesos micro y las macro consecuencia involucradas.

Con el objeto de modelar los sistemas complejos y aplicar los conceptos involucrados en la economía evolucionista, se han venido desarrollando una serie de paradigmas de modelamiento y simulación, entre los que se encuentran, la dinámica de sistemas y la simulación basada en agentes [72].

La simulación basada en agentes, es un paradigma de modelamiento que se caracteriza por comprender cómo varios agentes (autónomos, heterogéneos e independientes), con sus propias metas y objetivos son capaces de realizar interacciones entre sí y con su entorno. Para el caso particular de este trabajo, el objetivo de la simulación basada en agentes, es tratar de inferir las propiedades globales de todo el sistema o identificar los patrones de comportamiento emergentes a partir de las reglas que determinan el comportamiento individual de los agentes. El comportamiento global del sistema no es abstraído, sino que emerge durante el proceso de inferencia al ejecutar el modelo [72]. La aplicación de

la simulación basada en agentes en el contexto económico, es conocido como economía computacional basada en agentes.

Uno de los temas de investigación de la economía computacional basada en agentes es la evolución de normas de comportamiento, en el cual, la idea es estudiar qué papel juegan la reputación, la confianza, la reciprocidad, la venganza, el rencor, y el castigo en juegos. Tradicionalmente, el estudio de las normas de comportamiento es realizado usando teoría de Juegos Clásica.

La teoría de juegos es una herramienta matemática con aplicaciones en economía que permite analizar comportamientos estratégicos de jugadores cuando existe conflicto de intereses. Hay varios tipos de juegos en dicha teoría, entre ellos se encuentran: los juegos no cooperativos, en los cuales los jugadores no se pueden comunicar, es decir, no se puede llegar a acuerdos previos, por lo tanto cada jugador debe tener una estrategia de toma de decisiones con el fin de maximizar la recompensa esperada [70].

La teoría de juegos clásica trata de analizar la dinámica de la toma de decisiones haciendo uso de supuestos poco realistas como: homogeneidad (todos los jugadores son idénticos y aprenden siguiendo el mismo proceso) y racionalidad perfecta (jugadores ejecutan estrategias sin errores, jugadores suponen que cambiar su estrategia no provoca desviaciones en las estrategias de los otros contrincantes, existe un conocimiento común tanto de las reglas como de los supuestos de racionalidad).

En conclusión, se puede indicar que un juego en el enfoque clásico se interpreta como una descripción literal de una interacción idealizada en los que supuestos de racionalidad perfecta parecen bastante naturales, sin embargo, los modelos basados en racionalidad han sido desplazados en las últimas décadas por modelos evolutivos en los cuales se trata de estudiar las condiciones de cómo y porqué ciertos comportamientos en un entorno complejo pueden ser aprendidos por agentes con racionalidad limitada, a través de un proceso de adaptación guiado por planes estratégicos en juegos repetitivos [71].

Con el fin de comprender los comportamientos emergentes en los juegos no cooperativos, muchos trabajos de investigación han abordado el problema, entre las estrategias propuestas está la utilización de la economía computacional basada en agentes [79], permitiendo configurar a agentes económicos diferentes estados, reglas, estrategias o conductas, con el fin de experimentar variados escenarios del sistema y tratar de comprender comportamientos globales que surgen a nivel macro a través de interacciones locales repetidas de agentes egoístas. Sin embargo, estas aproximaciones requieren que las reglas y/o estrategias de los agentes sean preconfiguradas con anterioridad al proceso de simulación, elemento que en muchos sistemas no se conocen y que en algunas circunstancias no se pueden modelar dadas las características de complejidad, no linealidad e incertidumbre.

Otras aproximaciones, utilizan agentes integrados con redes neuronales, con el fin de modelar el comportamiento racional limitado de los agentes y evolucionar sus comportamientos de tal forma que se adapten mejor al ambiente, a través de la evolución de estrategias. La idea general es buscar mecanismos que permitan a los agentes originar, adaptar y retener estrategias que garanticen su adaptabilidad al entorno a partir de variables económicas subyacentes a un sistema económico específico [24]. Sin embargo, tradicionalmente, diseñar una red neuronal en caso de juegos no cooperativos involucra seleccionar la arquitectura de la red, la cual depende del problema a ser resuelto, por lo tanto, es un proceso manual en el cual un experto humano tiene que experimentar sobre diferentes

arquitecturas hasta encontrar una que retorne buenos resultados después de un proceso de entrenamiento basado en técnicas de aprendizaje reforzado.

Como solución a esta problemática surge la neuroevolución, mecanismo que permite diseñar redes neuronales artificiales usando técnicas evolutivas, con las cuales se puede evolucionar la topología, los pesos y las reglas de aprendizaje [74]. Diferentes estrategias de evolución han sido utilizadas, entre estas se encuentran: algoritmos evolutivos [35], técnicas de co-evolución [19] [88], programación genética[22] y programación genética gramatical [84]. Sin embargo, no es posible comparar dichos mecanismos dado que no existe una metodología común sobre la cual fueran diseñados los experimentos; no existe una plataforma de computación basada en agentes común sobre los cuales fueran modelados los diferentes casos de prueba especificados en cada proceso de investigación; adicionalmente, no hay métricas de comparación comunes con el fin de determinar medidas de rendimiento de los algoritmos [55].

Por otra parte, las plataformas de computación basadas en agentes en las cuales se realizan los procesos de simulación de los sistemas económicos requieren un proceso adicional, comprender o construir una plataforma de modelamiento con el fin de configurar los diferentes sistemas que se quieren simular. Actualmente existe una variedad de plataformas (StartLogo [12], NetLogo [83], RePast [21], etc.), sin embargo, para la implementación de modelos se deben conocer elementos sintácticos (en algunos casos complejos) del lenguaje, complicando un poco la tarea. Adicionalmente, proveer a los agentes mecanismos de adaptabilidad a partir de proceso de aprendizaje no está en el núcleo central de dichas plataformas y si lo están, requiere de cierta experticia en programación de computadores y las técnicas de aprendizaje particulares que se deseen aplicar.

El presente proyecto, realiza una comparación entre modelos de red neuronal en procesos de toma de decisiones realizadas por agentes en juegos repetitivos no cooperativos en el ámbito de la economía evolucionista usando como métrica de comparación la recompensa final obtenida, en el cual, se especifiquen modelos bajo un esquema de simulación basado en agentes para la toma de decisiones soportadas en redes Neuronales obtenidas a partir de técnicas de computación evolutiva (neuroevolución), en el cual se realicen los experimentos siguiendo lineamientos y métricas preestablecidas con el fin de analizar las diferentes técnicas de adaptación de las estrategias.

Como parte del desarrollo del trabajo, se realizaron las siguientes contribuciones en revistas indexadas y congresos.

Economía Computacional Basada en Agentes. El artículo muestra diferentes trabajos de investigación que se han venido desarrollando sobre un enfoque de simulación denominado, economía computacional basada en agentes, que rechaza las asunciones de los enfoques de estudio tradicionales que indican que la economía es un sistema cerrado que eventualmente logra un estado de equilibrio, en el cual supuestos de racionalidad perfecta, inversiones homogéneas, entre otros, deben realizarse para permitir que los modelos sean tratables analíticamente, en su lugar, ve a la economía como un sistema complejo, adaptativo y dinámico.
Este nuevo enfoque permite utilizar la simulación basada en agentes con el fin de comprender cómo varios agentes económicos (firmas, grupos económicos) con sus propias reglas y objetivos son capaces de interactuar entre sí y con su entorno, obteniendo como resultado final comportamientos emergentes que no son explicables directamente de las propiedades de los agentes individuales.

El anterior artículo fue aceptado y está en proceso de publicación en la revista LAMPSAKOS indexada en el índice Bibliográfico Nacional Publindex (IBN) de Colciencias (Colombia) en la categoría C

Aprendizaje de estrategias de decisión en juegos repetitivos no cooperativos en el ámbito de la Economía Evolutiva. Fue publicado en la revista Tecnura, indexada en el índice Bibliográfico Nacional Publindex (IBN) de Colciencias (Colombia) en la categoría B. El artículo presenta el diseño e implementación de diferentes mecanismos para realizar procesos de evolución de estrategias en juegos no cooperativos, específicamente en el dilema del prisionero iterado, ampliamente usado como modelo a estudiar en el ámbito de la economía evolutiva.
Las estrategias desarrolladas para los mecanismos de evolución de estrategias de juego fueron Algoritmos Genéticos (GA) y Particle Swarm Optimization (PSO). El resultado final es un ambiente de simulación en el cual se puede verificar la emergencia de estrategias que pueden vencer a otras estrategias a través de un proceso de entrenamiento en el cual se pueden especificar los juegos utilizando un enfoque de programación por bloques o a través de un lenguaje especifico de dominio textual, facilitando notablemente las tareas de programación involucradas[36].

Evolución de redes neuronales para la toma de decisiones en juego repetitivos no cooperativos.

El trabajo presenta el proceso de evolución de redes neuronales (perceptron, red de base radial) usando algoritmos genéticos para el aprendizaje de estrategias de decisión en juegos repetitivos no cooperativos, en los cuales los parámetros para configurar la topología de las redes son obtenidos usando un diseño experimental factorial mixto.
Los resultados obtenidos a través del proceso de evolución de las redes neuronales son comparables a los resultados obtenidos en la literatura, usando algoritmos genéticos y enjambres de partículas para los juegos: dilema del prisionero, juegos de la gallina y caza del ciervo. El artículo fue aceptado para ser presentado en el Octavo Congreso Colombiano de Computación (8CCC 2013), la indexación de las memorias saldrá en IEEE Xplore.

Laboratorio Computacional en el ámbito de la Economía Computacional basada en Agentes. Fue aceptado para ser presentado en el CICOM 2013, Tercer Congreso Internacional de Computación México-Colombia y XIV Jornada AcadÃ©mica en Inteligencia Artificial.

El artículo presenta un laboratorio computacional en el ambiente de la economía computacional basada en agentes. Dicho laboratorio permite la especificación de modelos de simulación usando un lenguaje desarrollado denominado UNALCOL. El lenguaje tiene una serie de características entre las cuales se encuentran: Un entorno integrado de desarrollo que facilita las tareas de programación y una plataforma de simulación para los modelos especificados.
Una característica importante de dicho lenguaje es que permite la integración con librería externas para soportar el proceso de soporte a decisiones. Con el fin de validar el funcionamiento del lenguaje se presenta como caso de prueba un modelo de juegos no cooperativos repetitivo, específicamente el dilema del prisionero iterado. Se configuran como estrategias de juegos ALLC (Cooperación), TFT (Tit for Tac) y por último un perceptrón multicapa. Los resultados obtenidos en el proceso de simulación evidencias la cooperación mutua.

El trabajo está estructurado de la siguiente forma: en el capítulo I, se muestran los objetivos del proyecto de investigación, en el capítulo II se presenta el estado del arte donde se muestran los fundamentos conceptuales y trabajos previos realizados en el área de estudio; el capítulo III presenta el proceso de evolución de redes neuronales para la toma de decisiones en juegos repetitivos no cooperativos, en el cual se entrenan redes neuronales (Perceptrón Multicapa, Red de base Radial) para competir con estrategias clásicas utilizadas en juegos No cooperativos tales como: el dilema del prisionero, el juego de la gallina y la caza del ciervo; en el capítulo IV, se presenta el diseño e implementación de un laboratorio computacional, en el cual se pueden especificar modelos de simulación basados en agentes en los cuales se modelan los juegos no cooperativos mencionados anteriormente, adicionalmente, a los agentes involucrados se les integra estrategias de aprendizaje basadas en redes neuronales.;se presentan tres escenarios a saber: agentes que compiten usando estrategias de juego comúnmente utilizadas en la literatura para competencias de juegos no cooperativos repetitivos, agentes con redes neuronales especificadas por el modelador; específicamente, se integraron los resultados obtenidos en el capítulo II, por último, a los agentes se les adapta un mecanismo de aprendizaje basado en redes neuronales y algoritmos genéticos en los cuales, el agente se adapta a la estrategia en línea, basado en el comportamiento del oponente.

Como parte final del trabajo, se presentan las conclusiones y trabajos futuros.

CAPÍTULO 1

Objetivos

1.1. Objetivo General

Comparar modelos de redes Neuronales en procesos de toma de decisiones realizadas por agentes en juegos repetitivos no cooperativos en el ámbito de la economía evolucionista.

1.2. Objetivos Específicos

1. Determinar un esquema de simulación basado en agentes para la toma de decisiones soportada en redes neuronales en juegos repetitivos no cooperativos.
2. Implementar modelos de redes neuronales en esquemas de simulación basados en agentes para la toma de decisiones en juegos repetitivos no cooperativos.
3. Formular un lenguaje específico de dominio para la especificación de modelos de simulación basados en agentes en juegos repetitivos no cooperativos con estrategias de aprendizaje basadas en redes neuronales obtenidas mediante técnicas de neuro-evolución.
4. Definir métrica de comparación de modelos de redes neuronales en modelos basados en agentes para la toma de decisiones en juegos repetitivos no cooperativos.
5. Realizar una comparación experimental de los modelos de redes neuronales aplicados en la toma de decisiones en juegos repetitivos no cooperativos como: Dilema del prisionero iterado, caza del ciervo (StagHunt) y el juego de la gallina (Chicken ó Snowdrif), basados en métricas de comparación definida.

CAPÍTULO 2

Estado del Arte

2.1. Introducción

La economía evolucionista es un área del pensamiento económico que rechaza las asunciones tradicionales, que indican que la economía es un sistema cerrado que eventualmente logra un estado de equilibrio, en la cual supuestos de racionalidad perfecta, inversiones homogéneas, entre otras, deben hacerse para permitir la tratabilidad analítica. En su lugar, ve a la economía como un sistema complejo adaptativo y dinámico que trata de comprender el comportamiento emergente del sistema macroscópico, a partir de la dinámica microscópica de cada agente (humanos, firmas, mercado, etc.). El interés principal es encontrar las condiciones o reglas en las cuales el comportamiento del sistema se asemeje al comportamiento real [49].

Con el fin de comprender los comportamientos emergentes en los sistemas económicos, se han desarrollado una serie de trabajos de investigación. Entre los enfoques propuestos, se encuentra, la utilización de la simulación basada en agentes, permitiendo configurar a los agentes diferentes estados, reglas y conductas, con el fin de experimentar diversos escenarios del sistema y tratar de entender los comportamientos globales emergentes que surgen a nivel macro a través de las interacciones locales repetidas de agentes que buscan su beneficio particular.

Otras aproximaciones, utilizan agentes integrados con algoritmos heurísticos y aprendizaje de máquina, con el fin de evolucionar el comportamiento de los agentes en los sistemas económicos, de tal forma que se adapten al ambiente, a través de la evolución de reglas y estrategias.

El presente capítulo tiene como objetivo exponer los fundamentos básicos de la economía computacional basada en agentes y ver cómo los sistemas económicos pueden ser vistos como procesos de conocimiento basado en reglas que evolucionan en el tiempo y que pueden ser simulados en ambientes computacionales, en los cuales los agentes tienen la capacidad de adaptarse al ambiente a través de procesos de aprendizaje y evolución de reglas de comportamiento.

El capítulo está organizado de la siguiente forma: la sección 2 provee los elementos principales de la simulación de sistemas complejos; la sección 3 presenta los elementos fundamentales de la economía computacional basada en agentes. La perspectiva de un

sistema económico como un proceso de conocimiento desde un enfoque macro, micro, y el enfoque maso presentado en la sección 4; la sección 5 presenta diferentes trabajos de investigación relacionados con proceso de evolución de reglas y estrategias; en la sección 6 se muestran diferentes plataformas de simulación basadas en agentes; por último, se presentan las conclusiones.

2.2. Economía Evolucionista

La economía evolucionista es un área del pensamiento económico que rechaza las supuestos tradicionales, que indican que la economía es un sistema cerrado que eventualmente logra un estado de equilibrio, en la cual premisas de racionalidad perfecta, inversiones homogéneas, entre otras, deben hacerse para permitir la tratabilidad analítica.
En su lugar, la economía evolucionista ve a la economía como un sistema complejo adaptativo y dinámico que trata de comprender el comportamiento emergente del sistema macroscópico, a partir de la dinámica microscópica de cada agente (humanos, firmas, mercado, etc.). El interés principal es encontrar las condiciones o reglas en las cuales el comportamiento del sistema se asemeje al comportamiento real.

Kurt Dopfer, John Foster y Jason Potts [23], [24], [25] , [46] desarrollan un entorno de trabajo analítico en el ámbito de la economía evolutiva con una arquitectura Micro, Meso y Macro. La idea central es definir un sistema económico como una población de reglas, una estructura de reglas y un proceso de reglas.

El nivel Meso, es entendido como el conjunto de reglas que gobiernan los sistemas. El nivel Micro, se refiere a los individuos (agentes) portadores de reglas y el sistema que ellos organizan (individuos, firmas, etc.). Cada agente está conformado por una estructura interna y una estructura externa. La estructura interna tiene relación con los procesos cognitivos (mente) y la estructura externa tiene relación directa con la interacción con otros agentes (Sociedad). El conocimiento aparece como un proceso interdependiente del intercambio de información continua entre agentes en la sociedad. [49]
El nivel Macro, consiste en la población de estructuras de los sistemas de Meso, en otras palabras, indica el orden o la coordinación entre las poblaciones de reglas o unidades meso. La Auto-Organización que se deriva del Meso determina la estructura del Macro.

Para explicar el entorno de trabajo parten de la discusión que los métodos tradicionales que exponen los procesos económicos realizan supuestos para permitir la tratabilidad analítica e indican que dicha aproximación está basada en tratamientos algebraicos que suponen que una actitud verdaderamente científica y filosófica de la economía debe estar basado en la lógica matemática. La expresión natural de dicho tratamiento algebraico en economía es la sobreposición de proposiciones formales microeconómicas, en la cual es necesario idealizar estados estáticos con el fin de deducir consecuencias macroscópicas.

A partir de lo anterior, proponen una manera de pensar diferente sobre las cuestiones fundamentales de la coordinación, el cambio de la economía y adaptan la perspectiva que la evolución económica es un crecimiento de procesos de conocimiento. Como consecuencia directa, indican que un sistema económico puede ser visto como una estructura compleja de reglas que evolucionan sobre un largo periodo de tiempo. El proceso por el cual nuevas reglas son originadas, adaptadas y difundidas en un sistema económico constituye el núcleo fundamental de la economía evolucionista.
El punto esencial para comprender este nuevo enfoque, es que el nivel macro no es un

comportamiento de agregación de los micro, elemento considerado en las interpretaciones tradicionales, sino más bien ofrece una perspectiva del sistema meso visto como un conjunto, del mismo modo, micro no es la esencia reducida de un sistema económico.

Una regla es un esquema deductivo que permite operaciones. El resultado de dichas operaciones son los efectos o consecuencias de las reglas.
En biología una nueva regla es creada por un error en el proceso de replicación genética de un organismo y la evolución ocurre cuando el error es selectivamente retenido por una especie por generaciones. En economía por ejemplo, las empresas pueden ser portadoras de genes económicos. Sin embargo, dichos genes son rara vez replicados con totalidad.

El crecimiento de conocimiento tiene aspectos relacionados con la micro organización de reglas y los aspectos macro asociados con la estructura de la población de reglas, pero la evolución económica puede ser vista como un proceso en el cambio(actualización) de reglas y del sistema de reglas. Es decir, el meso cambio es el núcleo de la evolución económica y la dinámica meso es llamada la meso trayectoria.

Una meso trayectoria puede ser vista como un proceso de tres fases: creación (Emergencia), difusión (adaptación y adopción) y retención (mantenimiento) de una regla de un sistema económico.
El proceso empieza cuando se introduce un nuevo conocimiento en un sistema económico. A partir de dicho conocimiento, en la fase de creación, cada agente desarrolla una idea/regla que conduce al diseño de una nueva organización de personas, energía, materiales. Un ejemplo de lo anterior, puede ser un empresario actuando con imaginación en un contexto con incertidumbre (uso de nueva tecnología). En este caso, el primer adoptante de la norma (conocimiento) realiza una nueva regla y por lo tanto tiene el potencial de desarrollar nuevas capacidades y participar en nuevas interacciones.

En la fase de creación hay muchas ideas y reglas, pero muchas de ellas nos son viables. La fase de difusión, involucra la adopción y adaptación de las reglas viables en el mundo de los contextos organizacionales. Así se encuentra una nueva organización micro y un meso orden en la economía evidenciada en el mercado y estructura organizacional industrial.

Por lo tanto, un proceso micro-meso puede implicar nuevos productos, nuevos consumidores, nuevas expectivas, nuevas organizaciones, nuevas interacciones, nuevos mercados, nuevas leyes, etc.

Basado en la fundamentación teórica que propone la economía evolucionista, es posible estudiar los comportamientos emergentes producto de nuevas políticas económicas, nuevas normas legislativas, comportamientos de agentes egoístas actuando en ambientes competitivos.
Ver un sistema económico como un sistema complejo adaptativo y dinámico que evoluciona a partir meso trayectorias, permite el uso de paradigmas de modelamiento usados tradicionalmente en la literatura (simulación basada en agentes), en los cuales la mente de los agentes y las interacciones son modelados de acuerdo a los principios fundamentales de la economía evolucionista.
Es decir, el ambiente puede ser artífice de nuevos ordenes económicos, a través de la aplicación de nuevas normas. Producto de dichas normas/reglas, los agentes deben interpretar dicha regla y crear comportamientos innovadores producto de la interpretación. A partir de la aplicación de la interpretación de los portadores de reglas, se pueden producir comportamientos emergentes que explicar comportamientos económicos existentes.

En el caso del proyecto de tesis. El ambiente definirá las reglas y pagos de los juegos no cooperativos. A partir de lo anterior, los agentes definirán su interpretación y crearán redes neuronales que cumplen con las características del juego.
Producto de la interacción entre los agentes en el ambiente, comportamientos emergentes (cooperación, no-cooperación) aparecen. Los agentes retendrán aquellas redes (reglas) que maximicen su ganancia. A partir de lo anterior, los otros agentes con los cuales interactúan definirán también su actualización.

2.3. Simulación de Sistemas Complejos

En años recientes, la ciencia de la complejidad ha sido reconocida por diferentes académicos en importantes disciplinas científicas como: la física, la economía, las ciencias de la computación y las ciencias sociales, por su capacidad para modelar sistemas no lineales [60]. Dicha ciencia, estudia los sistemas como un conjunto de componentes interconectados cuyo comportamiento no es explicable exclusivamente por las propiedades de los mismos, más bien el comportamiento emerge de los componentes interconectados. Con el objeto de modelar los sistemas complejos, se han venido desarrollando una serie de paradigmas de modelamiento y simulación entre los que se encuentran: la dinámica de sistemas [57] y la simulación basada en agentes [72].

La simulación basada en agentes, es un paradigma de modelamiento que se caracteriza por comprender cómo varios agentes (autónomos, heterogéneos e independientes), con sus propias metas y objetivos son capaces de realizar interacciones entre sí y con su entorno. Para el caso particular de esta revisión, el objetivo de la simulación basada en agentes, es tratar de inferir las propiedades globales de todo el sistema o identificar los patrones de comportamiento emergentes a partir de las reglas que determinan el comportamiento individual de los agentes. El comportamiento global del sistema no es abstraído, sino que emerge durante el proceso de inferencia al ejecutar el modelo [72].

Un escenario de interés particular en donde es utilizada la simulación basada en agentes, es la economía computacional, siendo un área del pensamiento económico que rechaza los supuestos tradicionales, encargados de indicar que la economía es un sistema cerrado que eventualmente logra un estado de equilibrio, en donde supuestos de racionalidad de perfecta, inversiones homogéneas, entre otros, deben hacerse para permitir el análisis matemático [31]. En su lugar, esta área del pensamiento, ve la economía como un sistema complejo adaptativo y dinámico que trata de comprender el comportamiento emergente del sistema macroscópico, a partir de la dinámica microscópica de cada agente (humanos, firmas, mercado, etc.) [32], [63].

Existen una serie de características que sustentan por qué la economía puede ser vista como un sistema complejo adaptativo; entre las que se encuentran: [18]: Las interacciones entre agentes heterogéneos son dispersas, no hay controlador global activo; la coordinación emerge por mecanismos de competencia y coordinación entre agentes; existe una organización jerárquica que permite a unidades económicas de diferente nivel comunicarse; los agentes computacionales tal como sucede en el mundo real pueden adaptarse continuamente al producto de la experiencia acumulada; los agentes pueden, continuamente, innovar basado en las regularidades que logran abstraer del mundo en el que actúan, por último, los agente actúan fuera del equilibro dinámico, es decir, los

agentes, dado su interacción, pueden mejorar constantemente las acciones en un ambiente económico.

2.4. Economía Computacional Basada en Agentes

Las economías son sistemas dinámicos complejos en los cuales un conjunto de agentes interaccionan a nivel local, dando lugar a regularidades globales, tales como: tasa de empleo y crecimiento, distribución de ingreso, tasas de desempleo, tasas de inflación, etc. Estas regularidades globales, a su vez, retroalimentan la determinación de las interacciones locales [8], [79].

La economía computacional basada en agentes (ACE) estudia los procesos económicos modelados como sistemas dinámicos de agentes interactuando. Ejemplos de posibles agentes incluye: individuos (consumidores, trabajadores), grupos sociales (familias, firmas, agencias gubernamentales), instituciones (sistemas reguladores del mercado), entidades biológicas (cultivos, ganados, bosques) y entidades físicas (infraestructura, clima y regiones geográficas) [80].

La investigación actual sobre ACE está encaminada en los siguientes aspectos:

Comprensión empírica Porque particularidades globales evolucionan y persisten a pesar de la ausencia de control y planificación centralizada.

Comprensión normativa Encargado de determinar cómo los modelos basados en agentes pueden utilizarse como laboratorios para el descubrimiento de buenos diseños económicos. El objetivo es evaluar propuestas de diseños de políticas económicas, instituciones, procesos y cómo son deseables socialmente en el tiempo.

Visión cualitativa y generación de teorías Analiza cómo pueden los sistemas económicos comprenderse más plenamente a través de un examen sistemático de los comportamientos dinámicos potenciales bajo condiciones iniciales especificadas [80].

Sin embargo, existe una serie de líneas de investigación bien definidas en el campo de la economía computacional basada en agentes, entre las que se pueden mencionar [79], [81]:

Aprendizaje La idea principal es buscar cómo modelar la mente de los agentes computacionales. Los investigadores en ACE han usado un amplio rango de algoritmos para los procesos de aprendizaje en los agentes. Entre los principales algoritmos se encuentran: aprendizaje por refuerzo [9], redes neuronales [29], algoritmos genéticos [38], programación genética [40] y otros algoritmos de computación evolutiva que permiten capturar aspectos de aprendizaje inductivo.

Evolución de normas de comportamiento Se trata de estudiar la evolución de la cooperación, a pesar que el engaño produce ganancias inmediatas. La idea es estudiar qué papel juegan la reputación, la confianza, la reciprocidad, la venganza, el rencor, y el castigo en juegos.

Formación de redes económicas Estudiar la manera en que las interacciones en las redes económicas están determinadas para escoger de forma deliberada los socios. Una de las preocupaciones claves en esta tarea, es la aparición de redes de comercio de compradores y vendedores que determinan sus socios de forma adaptativa, sobre la base de experiencias pasadas.

Modelamiento y organizaciones Estudia la manera de determinar la forma óptima de una organización para alcanzar unas metas específicas.

Laboratorio computacional Un laboratorio computacional es un entorno de trabajo preconstruido, con una interfaz gráfica, que permite realizar procesos de experimentación en un dominio específico permitiendo estudiar sistemas de múltiples agentes, interactuando por medio de experimentos computacionales controlados y replicables [75]. Entre los temas de investigación para la construcción de laboratorios computacionales se encuentran: construcción de laboratorios computacionales para cada tipo de aplicación, construcción de plataformas computacionales multifacéticas, mecanismos para realizar validaciones de resultados con datos obtenidos por otras fuentes.

Adicionalmente, un aspecto importante que se debe entrar a considerar en los laboratorios computacionales es conocer con qué grado de flexibilidad pueden ser especificados esquemas de aprendizaje para los agentes en laboratorios computacionales; a pesar que muchos estudios tienden a realizar sus algoritmos de aprendizaje en forma de simples ecuaciones de actualización con parámetros fijos, la evidencia acumulada indica que esos algoritmos no funcionan bien en todas las situaciones. Un mejor camino para proceder es permitir que los agentes en los laboratorios computaciones aprendan a aprender.

2.5. Evolución de Reglas y Comportamiento

La tarea del descubrimiento de reglas ha sido abordada desde diversos paradigmas: construcción de árboles de decisión, aprendizaje inductivo, y recientemente redes neuronales, algoritmos evolutivos [35], técnicas de co-evolución [19] [88], programación genética[22] y programación genética gramatical [84].

A continuación se presentan una serie de trabajos de investigación en el campo de la simulación basada en agentes que permite visualizar diferentes enfoques de evolución de reglas de comportamiento de agentes económicos.

Clemens Van Dinther en [20], expone modelos de aprendizaje que pueden ser integrados en simulaciones económicas basadas en agentes con el fin de modificar el conjunto de reglas que poseen los agentes dinámicamente. Entre los métodos mencionados están: los algoritmos genéticos, técnicas de aprendizaje reforzado y redes neuronales vistas como sistemas clasificadores.
Un algoritmo genético puede ser comprendido como un modelo de interacción estratégica, es decir, que el agente debe tratar de buscar la mejor estrategia en un ambiente determinando. Por lo tanto, los algoritmos genéticos pueden ser vistos como modelos de aprendizaje social (aprendizaje por interacción). Otras de las técnicas utilizadas es el aprendizaje reforzado, cuyo objetivo es que los agentes aprenden de los pagos recibidos por acciones pasadas.

Bell A en [9], propone un sistema de aprendizaje reforzado de reglas en un juego repetitivo, la idea principal es ver cómo los agentes aprenden de la interacción con su ambiente. Los agentes en este enfoque, asocian premios con acciones sobre la base de acciones pasadas y luego trasladan esos premios en probabilidades de tomar esas acciones en un futuro. Dicha aproximación es adaptada en aprendizaje basado en agentes en el contexto de los juegos, se presenta el problema del Farol o problema del bar Santafé, en el cual los agentes (no existe proceso de comunicación) deben decidir si ir o no ir al bar y dependiendo de la cantidad de agentes, se abre o no el bar, por lo tanto reciben un pago por la acción tomada en un instante de tiempo. Los resultados presentados evidencian la rápida convergencia del aprendizaje reforzado al comportamiento promedio agregado de los agentes, cuando los contrincantes son adaptativos.

Shu-Heng Chen en [40], presenta varios casos de aplicación del uso de la programación genética y programación genética gramatical en economía computacional basada en agentes. Entre los trabajos mencionados por el autor se encuentran: el desarrollado por Duffy y Engle-Warnick, en el cual se usa programación genética para inferir estrategias en experimentos de decisión económica específicamente, fue aplicado para la evolución de estrategias en el juego económico Ultimátum; los resultados obtenidos muestran la capacidad del algoritmo para generar reglas de decisión que permiten vencer al contrincante.

Robert Axelrod en [59], propone el uso de algoritmos genéticos para evolucionar estrategias en el dilema del prisionero iterado. Planteó la utilización de esquemas de codificación binaria tomando en consideración juegos previos y a partir de esta información obtener reglas que especifican la acción a tomar. Las estrategias obtenidas como producto de la evolución son similares a TIC FOR TAC; es decir, que existe cooperación mutua y castigo por deserción.
El esquema propuesto por Axelrod fue utilizado posteriormente por Errity A en [2], sin embargo, este último realizó una modificación en la codificación de los tres primeros movimientos con el fin de utilizar movimientos previos de las jugadas realizadas, en vez de jugadas aleatorias pre-configuradas.

Nelis F et al en [33], [62] presentan la aplicación de técnicas de coevolución basado en Particle Swarm Optimization (PSO) para evolucionar estrategias en el dilema del prisionero iterado, tres diferentes técnicas de coevolución son utilizadas: una red neuronal entrenada con PSO, la cual es usada para predecir la próxima acción a ser ejecutada, un enfoque binario del PSO en el que la partícula representa un estrategia de juego completa y finalmente un enfoque que explora las estructuras simétricas de las estrategias creadas por los humanos. Los resultados experimentales indican que el enfoque de redes neuronales es capaz de inducir altos niveles de cooperación, sin embargo en algunas ocasiones podrían conducir a colapsos catastróficos debido al desarrollo de estrategias débiles.

Floortje Alkemade en [5], presenta varios modelos entre los cuales se encuentran: ell oligopolio de Cournot, el cual tiene como objetivo modelar la búsqueda y aprendizaje del comportamiento de agentes económicos. También presenta un ambiente de simulación de duopolio de Cournot, donde los agentes toman decisiones haciendo suposiciones de las estrategias del otro agente en el mercado, en este caso hacen uso de un modelo de coevolución, en el cual cada firma tiene su población de estrategias y un algoritmo genético es aplicado en cada una de las firmas por separado. Los resultados en este caso, indican que existe una convergencia hacia el equilibrio competitivo.
Por último, presentan el uso de algoritmos genéticos para la difusión de información sobre una red social, con el fin de que las firmas puedan aprender acerca de la estructura de la red

y las características del consumidor, cuando la información habilitada es limitada, y usa esta información para evolucionar estrategias de marketing satisfactorias. Presentan como caso en la aplicación, el problema de decisión de una firma, de seleccionar la estrategia publicitaria para introducir con éxito un nuevo producto en una red de consumidores [28].

Thomas Riechmann en [78], presenta la aplicación de los algoritmos genéticos en el modelamiento y análisis de procesos de aprendizaje económico por medio de simulaciones por computador. En su estudio presenta la aplicación de la economía computacional para el modelamiento de problemas económicos, específicamente presenta un modelo de monopolios regionales cuyo objetivo es determinar la cantidad de bien a producir en una región con el fin de maximizar los beneficios. Los resultados obtenidos por los algoritmos genéticos permiten conocer a las firmas cuánto producir y ofertar para la venta. Como conclusión final luego del experimento, indica que los algoritmos genéticos efectivamente pueden ser utilizados como mecanismos de aprendizaje de estrategias en problemas económicos.

Ralf Herbrich et al en [43], en su trabajo realizan una acercamiento a la aplicación de las redes neuronales en la economía. En el campo de la simulación basada en agentes, las redes neuronales pueden ser consideradas como agentes que aprenden dependencias de su ambiente y por lo tanto infieren estrategias de comportamientos basado en un número limitado de observaciones.
Entre las aplicaciones de las redes neuronales en la economía mencionan: la clasificación de agentes económicos (clientes, compañias), la predicción de series de tiempo y el modelamiento de procesos de aprendizaje de agentes adaptativos artificiales con racionalidad limitada.

En el caso del modelamiento de procesos de aprendizaje, Thomas Sargen en [41], propone un trabajo, en el que muestra cómo las neuronas de la red son interpretadas como agentes que actualizan su percepción del ambiente de acuerdo a la información que ellos reciben, sus decisiones (salidas de las neuronas) ejercen una influencia sobre el ambiente, encargado de retroalimentar al agente.

Francesco Luna en [29], se concentra en el problema del rol de las instituciones en un proceso de aprendizaje en agentes económicos. En este trabajo se utiliza una red neuronal para modelar el proceso de aprendizaje, y un algoritmo genético se usa para desarrollar y obtener los pesos en esta red de la mejor forma posible para optimizar el aprendizaje.

Harrald y Fogel en [25] evolucionaron estrategias en el dilema del prisionero iterado utilizando perceptores multicapa feedforward. Todas las redes juegan con otras en una competición todos contra todos, los juegos duraban 151 movimientos y la medida de rendimiento utilizada fue el promedio de los pagos obtenidos en los movimientos. Luego las redes fueron ordenadas de acuerdo a la medida de rendimiento y las mejores (mitad), fueron seleccionadas para ser padres en la siguiente generación. Los resultados obtenidos indican que para converger a las estrategias buscadas se requieren muchas generaciones para conseguir estrategias con tendencia a cooperar.

Richart Lum et al en [67], hace uso de NEAT (Neuroevolution of Augmenting Topologies) para entrenar un jugador para el dilema del prisionero iterado.
NEAT es una técnica de evolución de redes neuronales basada en algoritmos genéticos, en la cual se puede evolucionar la topología de la red y los pesos de las conexiones.
Para realizar el modelamiento, indica que el proceso de adaptación a las estrategias variadas del oponente debe usar la historia de movimientos anteriores, en el caso puntual del trabajo desarrollado se tuvieron en cuenta los últimos 5 movimientos; tanto del contrin-

cante como propios.
Para la obtención de la medida de rendimiento (fitness), de cada una de las redes neuronales se planteó un torneo igual al desarrollado por Axelrod [84] cuando evolucionó estrategias utilizando algoritmos genéticos, pero en este caso se consideraron las siguientes estrategias: TIC FOR TAC, TIC FOR TWO TAC, ALWAY COOPERATE, ALWAY DEFECT. Los resultados obtenidos indican que las redes neuronales obtenidas se adaptaban a las estrategias de los contrincantes dado que se obtenían buenos pagos en cada uno de los enfrentamientos realizados.

Existen adicionalmente otros juegos no cooperativos que han sido objeto de estudio, Browning et al en [13], presentan un algoritmo genético para estudiar la evolución del comportamiento social en juegos repetitivos como juego de la gallina, la batalla de los sexos, caza del ciervo. Los resultados muestran la emergencia de la cooperación cuando existe reciprocidad en los juegos dilema del prisionero y el juego de la gallina.

2.6. Modelamiento del oponente

El modelamiento del oponente es usado en la literatura para referirse a un proceso de entrenar un jugador artificial para enfrentarse contra oponentes específicos en juego determinado.

Bakkes S. et al [7], definen el modelo del oponente como una representación abstracta de un jugador o el comportamiento de un jugador en un juego. El objetivo del modelo del oponente es mejorar las capacidades de un agente artificial, permitiendo adaptarse a sus oponente y de esta forma explotar sus debilidades.

Lockett A. et al [56], indican que el modelo del oponente puede ser descrito de dos maneras: de forma explícita o implícita.
En un modelo del oponente implícito, el algoritmo de aprendizaje codifica el conocimiento de los oponentes como estados internos y toma decisiones como producto del proceso de entrenamiento. Generalmente, es utilizado para defenderse de oponentes específicos; sin embargo, realizar el proceso de generalización del conocimiento específico para enfrentar otros oponentes desconocidos, no es posible sin realizar un proceso de entrenamiento.
Por su parte, el modelamiento explícito, entrena una forma funcional tomando información del oponente como entrada y produce una representación del oponente como salida, en términos de las acciones que el oponente puede tomar. La información a incluir en los modelos depende del modelo a generalizar, sin embargo, pueden ser utilizadas las acciones pasadas del oponente con el fin de predecir acciones futuras en información más directa o modelar oponentes desconocidos tomando como referencia un conjunto de oponentes conocidos.

Hingston P. et al [42], presentan un modelo de co-evolución para el dilema del prisionero iterado en el cual interactúan dos tipos de agentes: agentes adaptativos y agentes con estrategias fijas. En los agentes no adaptativos, las estrategias fijas son representadas como funciones de probabilidad, que indica la probabilidad de cooperar dado los n movimientos anteriores. Por su parte, los agente adaptativos mantiene un modelo de cada oponente con el fin determinar la contra-estrategia. El modelo de cada oponente es un conjunto de funciones M0, M1,.., Mn, donde cada M es un par ordenado (X, Y), X refleja cómo el oponente ha cooperado dado la secuencia de movimientos realizados por el oponente y Y refleja con qué frecuencia el oponente no ha cooperado. Estos valores pueden ser usados

para estimar la estrategia de juego del oponente y a partir de esta determinar una contra-estrategia. El proceso de actualización de los modelos del agente se realiza a través de una función de relevancia, aumentando el valor en X en uno, si el contrincante cooperó o Y en caso contrario. Posteriormente se aplica una función de decadencia.
Se realizaron dos tipos de experimentos, uno con solo agentes no adaptativos, los resultados al cabo de varias iteraciones evidencia la cooperación mutual, obteniendo valores de 2.783, lo cual indica que los integrantes de la población cooperaron en un 87 %. En otro experimento, pusieron a interactuar los dos tipos de agentes, los resultados muestran una disminución en el valor promedio del fitness a 2.52, sin embargo, la emergencia de la cooperación sigue vigente. La conclusión del trabajo como el aprendizaje y la evolución pueden ser combinadas en juegos estratégicos y evidencian el proceso de adaptación de estrategias usando la co-evolución.

Mark A et al [4], presentan un modelo de aprendizaje de estrategias online en juegos estratégicos (dilema del prisionero, piedra-papel y tijera). En dicho modelo las estrategias de juego son representadas por redes neuronales y se emplea un algoritmo genético para optimizar la estructura de la red (tamaño de la historia a utilizar). Adicionalmente, un mecanismo de aprendizaje es aplicado a red con el fin de realizar un proceso de adaptación online al modelo del oponente. El funcionamiento del algoritmo es: se genera un conjunto de posibles oponentes (imitadores), posteriormente, cuando se ha realizado un conjunto de juegos se busca en el conjunto de imitadores cual produce jugadas similares a las producidas por el contrincante hasta el momento actual del juego. Posteriormente, el mejor imitador es utilizado para predecir los próximos movimientos. Cada vez que el imitador genera un resultado correcto para los próximos movimientos se da una recompensa. El proceso de aprendizaje se da cada cierto número de rondas, realizando un proceso de entrenamiento con el algoritmo RPRO usando como datos la historia de juegos de los oponentes.
Para el proceso de experimentación, se hace uso de una red neuronal totalmente conectada, con una capa oculta y pesos entre [-1,1], los resultados muestran cómo se puede aprender la estrategia de juegos. Si el contrincante juega con RAND, ALLD, ALLC, and GRIM, la contra-estrategia encontrada es No cooperar, cuando se juega TFT y STFT al final del proceso se termina Cooperando, maximizando de esta forma el puntaje obtenido; para el caso de PAVLOV el comportamiento es irregular.

Lockett A. et al [56], proponen realizar un modelamiento explícito del oponente en el juego Guess It, usando para tal fin como información para el modelo, un conjunto de oponentes prototipo (Always Ask, Always Bluff, Call Then Ask, Call Then Bluff). Se hace uso de NEAT (NeuroEvolution ofAugmenting Topologies) con el fin de encontrar una red neuronal que represente al respetivo oponente.
En la experimentación, en cada generación, el jugador juega con los cuatro oponentes prototipo, hasta obtener la red neuronal que mejor representa la estrategia de juego. Como conclusión del trabajo indican que realizar el modelamiento del oponente, usando oponentes prototipo garantiza una maximización de la recompensa, dado que se obtienen tasas de ganancia de hasta el 90 %.

Bergsma M [10], presenta un estudio para realizar el modelamiento del oponente en el juego de cartas Machiavelli. Para tal fin, hace uso de redes neuronales totalmente conectadas entrenadas con dos algoritmos: backpropagación; usando como datos de entrenamiento juegos simulados, y algoritmos genéticos. Para el proceso de simulación crearon varios bot (agentes artificiales) representando diversas estrategias de juegos de oponentes. Los resultados obtenidos indican que cuando se usa backpropagación se obtienen representación de los oponentes hasta en un 70 %, sin embargo, con modelos de

algoritmos genéticos solo del 50 %. Como conclusión del trabajo indica que a pesar que va backpropagación da buenos resultados no es utilizable en aplicaciones reales, por lo tanto, se debe profundizar en el estudio de neuroevolución.

2.7. Plataformas de Modelamiento para Simulación Basada en Agentes

El desarrollo de simulaciones en el ámbito de la economía computacional basada en agentes, requiere comprender o construir una plataforma de modelamiento con el fin de configurar los diferentes sistemas que se quieren simular. Actualmente existen una variedad de plataformas entre las cuales se pueden encontrar:

Chris Langton et al en [61], desarrollan el sistema de simulación Swarm, un conjunto de herramientas para la simulación de sistemas multiagente.
Entre los trabajos desarrollados en el ámbito de la economía computacional se encuentran: el modelamiento basado en agentes para la evolución de sistemas eco-industriales, la meta del estudio es analizar las condiciones de sostenibilidad, es decir, conocer en qué momento los recursos se agotan.

K. Cao et al en [16], presentan como caso de estudio un modelo de un parque eco-industrial para la industria química implementado en el entorno de simulación basado en agentes Swarm. Otro de los trabajos desarrollados en [76], es un modelo que ilustra la emergencia y difusión de los estándares de software. La pregunta práctica a responder bajo este esquema consiste en cuantificar los beneficios de la estandarización.

El departamento de inteligencia artificial de la universidad de Wurzburg desarrolla SeSam (Shell foro Simulated Agent Systems), término definido como un ambiente genérico para modelar y experimentar con sistemas basados en agentes [15]. SeSam ha sido utilizado para la evolución de sociedades, para los cuales integraron en los agentes la capacidad de evolucionar sus reglas de comportamiento utilizando como base algoritmos genéticos con el fin de buscar los mejores parámetros en gráficos de actividad parametrizados [64].

Uri Wilensky en [83], [12], toma las características de StartLogoT (Evolución de StartLogo) y diseña NetLogo, caracterizado como un ambiente de programación multiagente multiplataforma y utilizado para la simulación de fenómenos complejos que evolucionan con el tiempo. Fue diseñada como plataforma educativa e investigativa y es ampliamente usada en diferentes disciplinas.

El grupo de ciencias sociales de la universidad de chicago desarrollan RePast (Recursive Porous Agent Simulation Toolkit) [21], entorno de trabajo extensible para la simulación de agentes. Entre los trabajos desarrollados bajo este entorno se encuentra: la construcción de un laboratorio computacional basado en agentes para ensayar la solvencia económica de la venta al por mayor en Mercados Eléctricos diseñados. La idea de este laboratorio, es probar en qué medida el mayorista (Wholesale Power Market Platform) es capaz de mantener el mercado ordenado y eficiente en el tiempo, a pesar de los intentos de los participantes en el mercado a obtener una ventaja individual a través de precios estratégicos [50], [86].

Andrew Begel en [48] propone una extensión de StartLogo, que denomina StartLogo TNG (The Next Generation), la cual provee dos signicativos avances con respecto a su versión previa. Primero, la programación es ahora realizada con bloques de programación

gráfica en lugar de comandos basados en texto y segundo habilita la representación 3D de los agentes en el mundo. Como se puede observar en los trabajos anteriores, para la implementación de modelos se deben conocer elementos sintácticos del lenguaje complicando un poco la tarea para la implementación. Adicionalmente, proveer a los agentes mecanismos de adaptabilidad a partir de proceso de aprendizaje no están en el núcleo central de dichas plataformas y si lo están, requieren de cierta experticia en programación de computadores y las técnicas de aprendizaje particulares que se deseen aplicar, habilidades de las cuales los modeladores carecen [36].

Con el fin de dar una solución a lo anterior, varios trabajos de investigación se han planteado, entre los cuales se encuentra el desarrollado por Okuyama et al, en el cual plantean el entorno de trabajo MAS-SOC (Multi-Agent Simulations for the SOCial Sciences) con el fin de permitir la creación de simulaciones sociales basadas en agentes a personas que no tengan mucha experiencia en programación. Por tal motivo, introdujeron el lenguaje ELMS (Enviroment Description Language for Multiagent Simulation) para la especificación de sistemas multiagentes, dicho lenguaje permite la especificación de; Agentes, reglas de comportamiento, percepciones, acciones, ambiente, entre otras, usando un lenguaje de alto nivel basado en XML [30].

A pesar que es un lenguaje más simple que los provistos por las plataformas mencionadas anteriormente, tiene elementos que puede dificultar la especificación de modelos: No existe un entorno de desarrollo integrado que facilite las tares de programación, no permite la integración con librerías de aprendizaje externas.

2.8. Resumen

El modelamiento de un sistema económico como un sistema complejo adaptativo, permite descubrir nuevos patrones emergentes de la interacción de agentes económicos individuales actuando en un ambiente, debido a que los supuestos tradicionales para permitir tratabilidad analítica son omitidos. En muchos procesos cotidianos tener un conocimiento de las relaciones globales entre conceptos limita los procesos de simulación, dado que generalmente se conocen los comportamientos individuales de los agentes involucrados, siendo este un elemento clave de la simulación basado en agentes, se pueden buscar explicaciones de procesos globales en sistemas económicos con poca información.

La integración de paradigmas de simulación basada en agentes, economía y técnicas como algoritmos evolutivos y aprendizaje reforzado, permiten que los agentes en un proceso de simulación, puedan descubrir las reglas individuales y de interacción que llevan a un sistema económico a una situación particular. Es decir, que se podría en un momento determinado inferir fallas, políticas y estrategias que se deberían tener en los agentes, con el fin de lograr un comportamiento global buscado. Lo anterior, permitiría visualizar la economía evolutiva desde una perspectiva Micro, Meso y Macro.

Se requieren plataformas de modelamiento de sistemas basados en agentes, que tengan como núcleo fundamental mecanismos de evolución (adaptación) de comportamiento de los agentes, utilizando estrategias evolutivas y técnicas de aprendizaje de máquina, en los cuales modeladores (economistas) con un lenguaje simple puedan simular diferentes procesos, con el fin de explicar por qué ciertos comportamientos económicos permanecen en el tiempo, a pesar que no exista control y planificación centralizada, adicionalmente

que les permita simular nuevos escenarios económicos con el fin de evaluar propuestas de políticas socialmente aceptables, entre otros.

El presente proyecto de investigación se centrará en el estudio de aprendizaje de estrategías de decisión utilizando redes neuronales artíficiales en juegos repetitivos no cooperativos en el ámbito de la economía evolucionista. Lo anterior, enmarcado en las líneas de investigación de la economía computacional basada en agentes, a saber:

- *Normas de Comportamiento:* Se centra el estudio en el área de teoría de juegos, específicamente la teoría de juegos No cooperativos. El objetivo es analizar las interacciones entre individuos que bajo un conflicto de intereses y reglas establecidas seleccionan racionalmente una estrategia con el fin de maximizar la recompensa esperada.

- *Laboratorio Computacional:* Se plantea desarrollar un laboratorio computacional para la especificación de simulaciones en el ámbito de la economía computacional basada en agentes, se tendrán en cuenta las ideas introducidas por Dopfer et al [24] Adicionalmente, buscando un esquema unificado para la especificación de simulaciones económicas, se ha planteado el desarrollo de un entorno de simulación bajo la propuesta de agentes inteligentes planteado por Russell et al [69]. Para lo cual es necesario definir la arquitectura y el programa de cada uno de los agentes involucrados en un proceso de simulación. La arquitectura debe permitir definir los sensores, actuadores, entorno y la medida de rendimiento.

  Por su parte, el programa debe implementar la función del agente, es decir, definir las acciones a ejecutar basado en las percepciones, el estado interno, la función de utilidad esperada en cada juego y los mecanismos de aprendizaje.

  En el caso de la economía computacional basada en agentes, cada agente (Arquitectura Micro) participante debe tener la capacidad de percibir: las decisiones del contrincante basado en las reglas de decisión utilizadas en cada una de las iteraciones, la recompensa entregada por el entorno en función de la tabla de recompensas alterando de esta forma el comportamiento (Arquitectura Macro).

  Dicha información debe ser almacenada temporalmente en un estado interno con capacidad limitada que posee cada agente, con el fin que el módulo de aprendizaje, pueda evaluar el resultado obtenido y basado en criterios de calidad y experiencia modificar los elementos de actuación definidos, es decir, modificar las reglas de tal manera que se pueda mejorar la selección de acciones buscando optimizar las medidas de rendimiento.

- *Aprendizaje:* Se plantea el desarrollo de evolución de estrategias de decisión usando redes neuronales, especificamente: Perceptrón multicapa y redes de base radial. Tal como se indicó anteriormente, el proceso de evolución de estrategias para el caso del presente proyecto de investigación consiste en entrenar a un agente para enfrentarse contra otros agentes en Juegos repetitivos no cooperativos.

  Se presentarán dos esquemas, un modelo ímplicito y un modelo explícito. Para el modelo ímplicito, se realiza un proceso de entrenamiento fuera de línea (offiline) con el fin tener redes neuronales obtenidas por procesos de Neuroevolución, que tengan la capacidad de defenderse cuando se enfrenta con oponentes específicos. Para el caso del modelo explícito, se adapta en línea el comportamiento del agente basado en el

contrincante con el cual se enfrenta, para tal fin, el agente basado en la historia de juego, busca obtener un contrincante prototipo usando esquemas de neuroevolución con el fin de adaptar su estrategia de juego dinamicamente.

CAPÍTULO 3

Evolución de redes Neuronales para la toma de decisiones en Juegos repetitivos no Cooperativos

3.1. Introducción

La teoría de juegos clásica es una herramienta matemática con aplicaciones en economía que permite analizar las interacciones entre individuos que bajo un conflicto de intereses y reglas establecidas buscan seleccionar racionalmente una estrategia con el fin de maximizar la recompensa esperada. En otras palabras, la teoría de juegos provee un lenguaje para formular, analizar y comprender escenarios estratégicos [70].

Entre los tipos de juegos que estudia la teoría de juegos se encuentran los no cooperativos, en los cuales los jugadores no pueden llegar a acuerdos previos y deciden su estrategia en función de los resultados esperados teniendo en cuenta las decisiones de los otros jugadores.

Los juegos no cooperativos o estratégicos pueden ser representados de dos formas: la forma normal, para juegos estáticos (las acciones de los jugadores son tomadas simultáneamente) y la forma extensiva para juegos dinámicos (un jugador tiene la posibilidad de observar los movimientos de los contrincantes antes de tomar una decisión).
La solución de los juegos estratégicos desde el punto de vista de la teoría de juegos clásica consiste en buscar los conceptos de solución, es decir, la teoría de equilibro que predice la asignación de pagos a los jugadores (métrica basada en recompensa).

Existe una importante variación de los juegos expresados en forma normal, denominados juegos repetitivos, en donde, un juego simultáneo es jugado múltiples veces (finita o infinitamente) por el mismo conjunto de agentes. Los agentes en cada ronda juegan con información perfecta de la historia, es decir, los agentes conocen la secuencia de acciones de las rondas previas, lo anterior habilita a los jugadores a desarrollar estrategias de juego con el fin de maximizar los pagos recibidos. Teóricamente, el movimiento de un jugador puede influir en el comportamiento de su oponente en el futuro, afectando de esta manera futuros pagos.

La importancia de estudiar los juegos repetitivos es que pueden ser utilizados para modelar relaciones económicas, sociales, políticas, entre otras, que presentan escenarios que se repiten constantemente en el tiempo. La idea general de este tipo de juegos, es

entender cómo cada jugador debe comportarse con el fin de evitar castigos o buscar una recompensa en el futuro. Ejemplos de este tipo de juegos son: negociaciones comerciales, Duopolios, Oligopolios, etc.

A pesar de los aportes de la teorías de juegos clásica existen diversas críticas, dado que los juegos son interpretados como descripciones literales de interacciones idealizadas (la aplicabilidad a modelos del mundo real puede verse limitada), en los cuales los supuestos de racionalidad perfecta y homogeneidad parecen bastante naturales.

La teoría clásica tiene en consideración una serie de supuestos para analizar las situaciones estratégicas entre los jugadores, entre los cuales se encuentran [39]:

1. Las acciones individuales son instrumentalmente racionales, las preferencias están representadas por funciones de utilidad, por lo tanto su objetivo es maximizar la utilidad esperada.

2. Debe existir un conocimiento común de racionalidad, es decir, cada jugador es instrumentalmente racional, cada jugador conoce que es instrumentalmente racional, cada jugador conoce que los demás jugadores son instrumentalmente racionales y así sucesivamente.

3. Existen prioridades comunes, cuando existen jugadores racionales con la misma información, deben obtener las mismas consecuencias y llegar de maneja independiente a las mismas conclusiones. Es decir, que se asume que los jugadores realizan inferencias y aprenden siguiendo el mismo proceso.

4. Las acciones son ejecutadas dentro de las reglas de juego, los jugadores conocen las reglas y las combinaciones de acciones a realizar para obtener una recompensa particular.

5. Los jugadores ejecutan sus estrategias sin errores. Es decir, la teoría de juegos asume que todos los jugadores son racionales. Esto significa que cada jugador podría escoger la acción que maximiza su utilidad esperada dada su creencia acerca de las acciones que los otros jugadores van a escoger.

El resultado de dichas críticas, dio origen a nuevas teorías como la teoría de juegos evolutivos, que no se basa en supuestos de racionalidad perfecta, sino en procesos de selección natural Darwiniana [71].

Los juegos evolutivos tratan de estudiar las condiciones de cómo y porqué ciertos comportamientos en un entorno complejo puede ser aprendido por agentes con racionalidad limitada, a través de un proceso de adaptación guiado por planes estratégicos.

Diferentes esquemas de evolución han sido utilizados para el perfeccionamiento de estrategias en juegos no cooperativos, tales como los desarrollados en [59], [25],[67], etc.

El presente capítulo tiene como objetivo presentar la capacidad de generalización y adaptación de las redes neuronales artificiales obtenidas por procesos de evolución para el aprendizaje de estrategias de decisión en juegos no cooperativos usando como métrica de comparación la recompensa final obtenida. Así como también, validar experimentalmente los resultados, realizando comparación con esquemas de solución existentes en la literatura. En este caso, el ambiente define las normas/ reglas de los juegos y los agentes a partir de la interpretación de la regla, originan redes neuronales que pueden ser retenidas a travÃ©s de

las iteraciones, si dichas redes dan como resultado, la maximización del beneficio cuando se compite con otros agentes. El planteamiento anteriormente mencionado, se modelará bajo los conceptos teóricos de la economía evolucionista.

El capítulo está estructurado de la siguiente forma: en la sección II se especifica los juegos no cooperativos usados en la investigación, la sección III desarrollan los métodos utilizados para la realización de procesos de evolución; en la sección IV se realiza el proceso de experimentación para juegos estándar y torneos para cada uno de los dilemas estudiados; por último, en la sección V se presentan las conclusiones de la investigación.

3.2. Juegos No Cooperativos

En un juego estratégico en forma normal, los jugadores son asumidos que escogen estrategias puras de forma simultanea e independientemente y reciben un pago que depende del perfil de estrategia utilizado.

En el presente trabajo de investigación se estudiarán tres juegos clásicos de la teoría de juegos denominados juego de suma no nula: el dilema del prisionero formulado por el matemático Tucker con base en las ideas de Flood y Dresher en 1950 [66], el juego de la gallina formulado por Freedman en los años 60 pero popularizado por Bertrand Russell en 1975 [17] y la caza del ciervo basado en el discurso sobre la desigualdad de Rousseau [85].

El dilema del prisionero (PD) es un modelo clásico de la teoría de juegos denominado juego de suma no nula, formulado por el matemático Tucker con base en las ideas de Flood y Dresher en 1950. Desde entonces ha sido discutido ampliamente por los teóricos de juegos, economistas, matemáticos, biólogos, filósofos, especialistas en política, especialistas en ética, sociólogos y científicos de la computación [66]. El juego es tradicionalmente descrito usando la metáfora de dos sospechosos que han sido arrestados y están siendo interrogados por separado. Sin ningún tipo de comunicación, cada jugador debe decidir si cooperar C (guardar silencio) o no cooperar D (traicionar a su pareja). Cada jugador recibe un pago individual dependiendo de las decisiones tomadas.

El juego de la gallina es un modelo clásico de la teoría de juegos denominado juego de suma no nula, formulado por Freedman en los años 60, pero popularizado por Bertrand Russell en 1975. El juego es tradicionalmente descrito usando la siguiente metáfora: una competencia de autos en la que dos participantes conducen un vehículo en dirección al del contrario; cuando los dos autos se acercan cada conductor tiene dos opciones: desviarse o no. Si un conductor se desvía, luego el conductor es una gallina y el otro es un valiente. Si ambos se desvían ambos son gallinas, y si ninguno se desvía, luego una colisión catastrófica tiene lugar [17].
Haciendo una analogía con el dilema del prisionero, se denomina cooperación a desviarse y no-cooperación a no desviarse. Pero mientras la cooperación mutua en el dilema del prisionero conduce a unos resultados inferiores, en el juego de la gallina conduce no solo a resultados que son inferiores, sino devastadores. El juego se basa en la idea de crear presión psicológica hasta que uno de los participantes se echa atrás.

El juego la caza del ciervo es un modelo clásico de la teoría de juegos denominado juego de suma no nula, basado en el discurso sobre la desigualdad de Rousseau, dicho modelo es descrito usando la siguiente metáfora: la caza del ciervo es un juego que describe un conflicto entre la seguridad y la cooperación social. Supongamos que los cazadores única-

mente cuentan con dos opciones: cazar ciervos o cazar liebres. Cada jugador debe elegir una acción sin conocer la del otro. La probabilidad de cazar una liebre es independiente de la decisión de los otros, sin embargo, si un individuo decide cazar un ciervo debe cooperar con su compañero para tener éxito.

Es imposible abatir un ciervo en solitario, pero la probabilidad de cazar un ciervo se incrementa notablemente con el número de participantes [85].
Un jugador individual puede cazar una liebre por sí mismo, pero una liebre vale menos que un ciervo. Esta situación se considera una analogía importante con la cooperación social.

En cada uno de los juegos existen cuatro posibles resultados: Ambos jugadores deciden cooperar (Reward), ambos jugadores deciden No cooperar (Punishment), el Jugador A decide cooperar mientras que el jugador B no coopera (Sucker), y el jugador A decide no cooperar mientras que el jugador B decide cooperar (Temptation).

Para el caso del dilema del prisionero el juego constituye un dilema si se cumplen las siguientes desigualdades, 3.1

$$T > R > P > S \, y \, 2R > S + T \tag{3.1}$$

Los valores típicos utilizados en la literatura para los procesos de experimentación son presentados en la tabla 3.1

Tabla 3.1. Valores del Dilema del Prisionero

Jugador A	Jugador B	
	Cooperar	No Cooperar
Cooperar	R=3,R=3	S=0,T=5
No Cooperar	T=5,S=0	P=1,P=1

Si existe una única oportunidad de jugar y basados en supuestos de racionalidad, la mejor decisión es siempre no-cooperar; por lo tanto, esta última es denominada una estrategia dominante en el juego (equilibrio de Nash).

En el juego de la gallina, si existe una única oportunidad de jugar y basados en supuestos de racionalidad, no existe una estrategia dominante, existen dos puntos de equilibrio (No-cooperar, Cooperar) y (Cooperar, No-cooperar).
La conformación de pagos es similar al dilema del prisionero, sin embargo, se debe cumplir la desigualdad 3.2:

$$T > R > S > P \tag{3.2}$$

La matriz de pago es presentada en la tabla 3.2

Por último, en la caza del ciervo la matriz de pago debe cumplir con la siguiente restricción, 3.3:

$$R > T >= P > S \tag{3.3}$$

Los valores típicos son presentados en la tabla 3.4

TABLA 3.2. Valores del Juego de la Gallina

Jugador A	Jugador B	
	Cooperar	No Cooperar
Cooperar	R=3,R=3	S=0,T=5
No Cooperar	T=5,S=0	P=0,P=0

TABLA 3.3. Valores de la Caza del Ciervo

Jugador A	**Jugador B**	
	Cooperar	No Cooperar
Cooperar	R=5,R=5	S=0,T=3
No Cooperar	T=3,S=0	P=1,P=1

Muchas variaciones de los juego han sido propuestas; una de ellas es el dilema del prisionero iterado (IPD) presentada por Axelrod en 1984 que puede ser extendida a los otros juegos, en el cual dos contrincantes juegan repetidamente el dilema del prisionero. La clave de los juegos repetitivos es que ambos jugadores pueden jugar nuevamente, lo cual habilita a desarrollar estrategias de juego basados en interacciones de juegos previos con el fin de maximizar los pagos recibidos. Teóricamente el movimiento de un jugador puede influir en el comportamiento de su oponente en el futuro, afectando de esta manera futuros pagos.

3.3. Método de Solución

Una red neuronal artificial consiste de una colección de elementos de procesamiento que están altamente interconectados y transforman un conjunto de entradas en un conjunto de salidas deseadas. El resultado de la transformación es determinada por las características de los elementos y los pesos asociados con las interconexiones entre ellos.
Entre las características principales de la red neuronal se incluyen la habilidad para aprender dependencias basadas en un número finito de observaciones.

El proceso de diseñar redes neuronales artificiales involucra dos partes: desarrollar la arquitectura y realizar un proceso de entrenamiento y validación. El desarrollo de la arquitectura involucra determinar cuántas neuronas va tener la red neuronal, cuántas capas y cómo van a ser interconectadas. Por su parte, el entrenamiento se refiere al cálculo de los valores de los pesos de las conexiones. Con el fin de automatizar el proceso de generación de redes neuronales, se han aplicado mecanismos de computación evolutiva (Neuroevolución) con el fin de evolucionar pesos de las redes neuronales, arquitectura, funciones de transferencia y las entradas (Buscar el conjunto de entradas óptimo).

3.3.1. Perceptrón Multicapa

Los perceptrones multicapa son redes totalmente conectadas hacia adelante con una o más capas de neuronas entre las capas de entrada y salida. Cada capa está compuesta de una o más neuronas artificiales en paralelo. Una neurona es presentada como en la Figura 3.1, tiene N pesos de entrada y una única salida.

Una neurona combina esos pesos de entrada para formar su suma y con referencia a un valor umbral y la función de activación, determinar la salida.

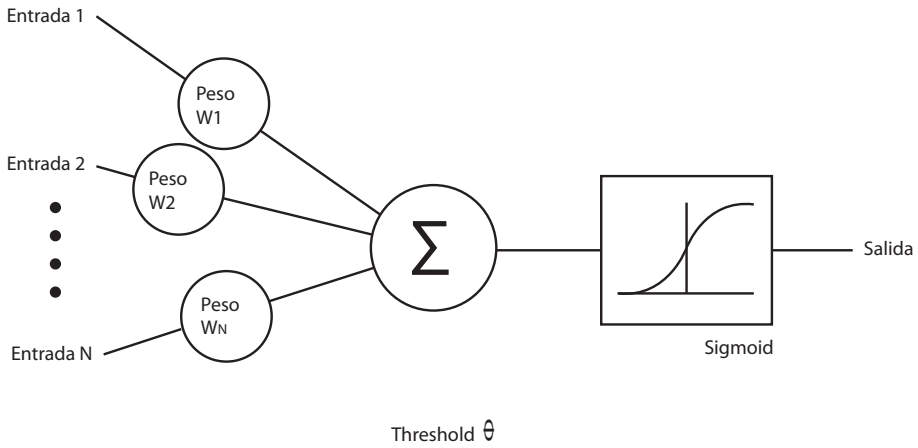


FIGURA 3.1. Funcionamiento de una neurona artificial.

Sean $x_1, x_2, ..., x_n$ las señales de entrada, $w_1, w_2, ..., w_n$ los pesos sinápticos, μ el potencial de activación, θ el umbral, y la señal de salida y f la función de activación [34].

$$u = \sum_{i=1}^{n} w_i * x_i \tag{3.4}$$

$$y = f(\mu - \theta) \tag{3.5}$$

Definiendo $w_o = \theta$ y $x_0 = -1$, la salida del sistema puede ser reformulado como:

$$y = f(\sum_{i=1}^{n} w_i * x_i) \tag{3.6}$$

La función de activación define la salida de la neuronas en términos del nivel de actividad en su entrada. La forma más común de función de activación usada es la función sigmoidal, sin embargo, existen otras funciones de activación tales como las especificadas a continuación.

1. Sigmoidal: $f(x) = \frac{1}{1+e^{-x}}$
2. Tangente Hiperbólica: $f(x) = \frac{e^{2x}-1}{e^{2x}+1}$
3. Senoidal:

La Fig 3.2, presenta un perceptrón con una capa de entrada, una capa oculta y una capa de salida. Un perceptrón de una única capa implementa un único hiperplano,un perceptrón de dos capas implementa regiones convexas arbitrarias consistentes de la intersección de hiperplanos, por su parte un perceptrón de tres capas implementa superficies de decisión de complejidad arbitraria, es por esta razón que los perceptrones de tres capas es la arquitectura más común [77].

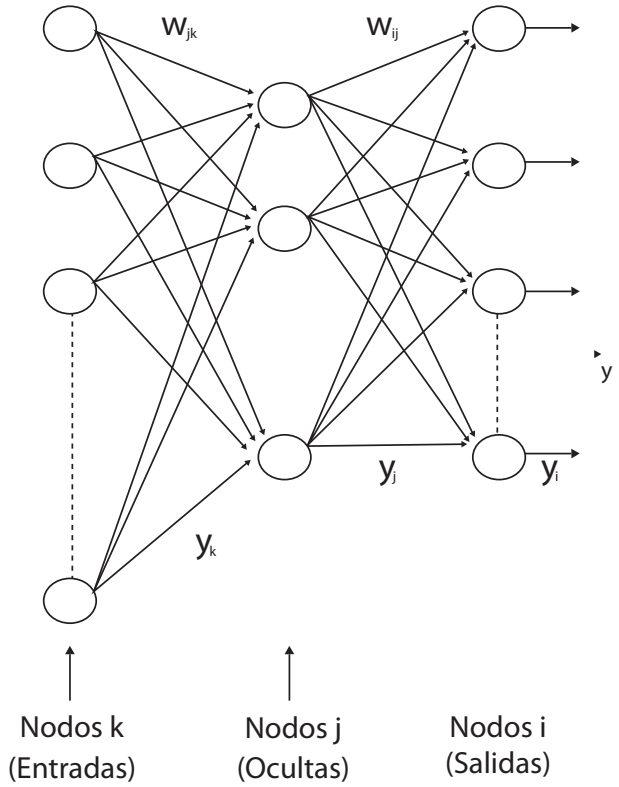


FIGURA 3.2. Funcionamiento de una neurona artificial.

El proceso de entrenamiento de un perceptrón multicapa es un proceso iterativo de ajustes aplicando sus parámetros libres. Los algoritmos de aprendizaje más comunes son back-propagation; este usa técnicas de búsqueda de gradiente para minimizar una función de costo igual al error cuadrático medio (MSE) entre la salida deseada y la salida actual de la red.

$$MSE = \frac{1}{l}\sum_{i=1}^{l}(y_i - \hat{y}_i)^2 \tag{3.7}$$

El desempeño del entrenamiento de una red neuronal depende del algoritmo de aprendizaje utilizado, del número de capas ocultas, del número de neuronas en cada capa oculta, de la conectividad o arquitectura de la red y también del tipo de función de activación que se elija para cada neurona [54].

Por otra parte, decidir el número de neuronas en la capa oculta es una parte importante, tener un número pequeño de neuronas en la capa oculta puede resultar en problemas de bajo ajuste, si existen muchas neuronas en la capa oculta puede suceder varios inconvenientes: Sobre-ajuste, tiempo de entrenamiento.

El número de neuronas por capa normalmente se determina por ensayo y error, aunque existe el criterio de considerar al promedio entre el número de entradas y salidas como un valor referencial del número de neuronas en las capas ocultas. Algunas reglas que pueden ser utilizadas para realizar el proceso de determinación son [77]:

1. El número de neuronas ocultas podría estar entre el tamaño de la capa de entrada y el tamaño de la capa de salida.

2. El número de nueronas ocultas podría ser 2/3 del tamaño de la capa de entrada, más el tamaño de la capa de salida.

3. El número de neuronas debe ser menos de dos veces el número de neuronas de la capa de entrada.

Otro de los criterios importantes son las funciones de activación; la función de activación cumple con el objetivo de limitar el rango de salida de la neurona y puede ser lineal o no lineal. Se selecciona de acuerdo con el problema y el criterio del investigador, en ocasiones por ensayo y error.

Una buena función de activación debería cumplir: primero, que ella misma y su derivada sean fáciles de computar y segundo, que la función debe tener una amplia parte lineal para lograr velocidad de entrenamiento y convergencia en pocos ciclos[54].

En el caso del perceptrón multicapa, se realizará el proceso de evolución de los pesos de las conexiones entre cada una de las neuronas, de cada una de las capas, usando como técnica de evolución algoritmos genéticos.
En la Fig 3.3, se visualiza un perceptrón multicapa totalmente conectado, específicamente con tres capas: una capa de entrada con dos neuronas y el bias, una capa oculta con dos neuronas y el bias y por último una capa de salida con una única neurona.

Por lo tanto, se deben evolucionar nueve pesos $H1, H2, ..., H9$, que corresponden con las diferentes interconexiones presentes entre las neuronas de las diferentes capas

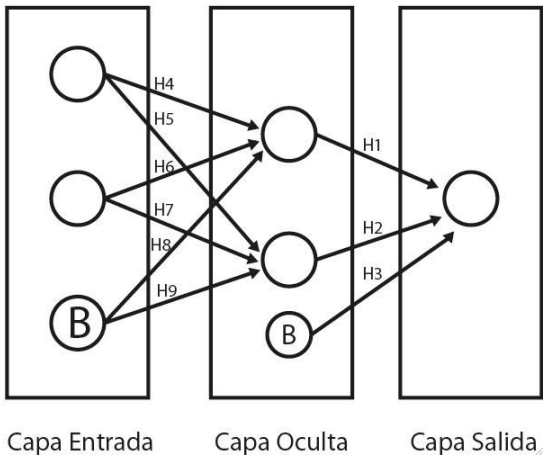


FIGURA 3.3. Evolución Perceptron Multicapal.

Con el fin de generalizar el esquema de evolución y permitir varias capas ocultas, así como un número determinado de neuronas en cada una de las capas, se puede determinar el número de pesos a evolucionar usando la Ec. 3.8.

$$W = ((I + 1) * H) + ((H + 1) * O) + ((L - 1) * (H + 1) * H) \tag{3.8}$$

Donde W es el número de pesos a evolucionar, I es el número de neuronas en la capa de entrada, H es el número de neuronas en la capa oculta, O el número de neuronas en la capa salida y L es el número de capas.
Aplicando la ecuación al caso presentado en la Fig. 1, obtenemos el número de pesos especificado en la Ec. 3.9:

$$W = ((2 + 1) * 2) + ((2 + 1) * 1) + (0 * (2 + 1) * 2) = 6 + 3 = 9 \tag{3.9}$$

Con el fin de evolucionar estrategias para los juegos no cooperativos planteados y basados en el esquema de representación propuesto por Axelrod [59], las redes neuronales tendrán seis neuronas en la capa de entrada y una neurona en la capa de salida. El número de capas ocultas, las funciones de activación y el número de neuronas en las capas ocultas más adecuado será determinado a través de procesos de experimentación.

El esquema de evolución a utilizar para realizar el proceso de evolución de los pesos es algoritmos genéticos; para el diseño del cromosoma del algoritmo genético, se planteó un cromosoma donde el tamaño corresponde al número de pesos a evolucionar y cada uno de los genes puede tomar valores en un intervalo entre *[-0.5, 0.5]*, dicho esquema está basado en los estudios previos realizados por Fogel et al [25].

3.3.2. Redes de Base Radial

Una red neuronal de base radial es un tipo especial de red neuronal que usa funciones de base radial como funciones de activación. Las redes de base radial son muy populares para la aproximación de funciones, predicción de series de tiempo, problemas de clasificación y control.

Una red de base radial es una red neuronal que está conformada por tres capas: una capa de entrada, una capa oculta y una capa de salida.

En la capa de entrada hay tantas neuronas como características de entrada. Las neuronas de entrada justo propongan las características de la entrada a próxima capa.
Cada neurona en la capa oculta es asociada a una función kernel $\phi_j(.)$ caracterizada por un centro μ_j y desviación estandar σ_j.

La salida está compuesta de tantas neuronas como clases deben ser reconocidas. Cada neurona de salida O_i calcula una suma ponderada sobre las respuestas de las neuronas ocultas para un patrón de entrada dado. Es decir, que para cada neurona de salida O_i se aplica la ecuación 3.10.

$$O_i(x_i) = \sum_{j=1}^{k} w_l, j\phi_j(x_i) + w_bias, l \tag{3.10}$$

donde k es el número de neuronas en la capa oculta, w_l, j representa el peso asociado entre la función kernel $\phi_j(.)$ y la neurona de salida O_l y w_bias, l es el sesgo (bias) a cada

una de las neuronas de salida. En la figura 3.4

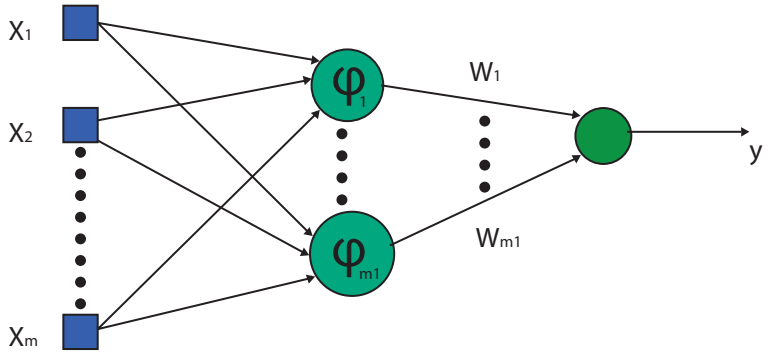


FIGURA 3.4. Evolución Red de base Radial

Es importante indicar que diferentes funciones kernel pueden ser utilizadas, por ejemplo:

1. Gaussian: $\phi(z) = exp(-z^2/2\sigma^2)$

2. Cuadrática: $\phi(z) = (z^2 + r^2)^{1/2}$

3. Cuadrática Inversa: $\phi(z) = 1/(z^2 + r^2)^{1/2}$

Donde $z = \|x - c_j\|$

El proceso de entrenamiento de las redes de base radial, es realizado en dos pasos:

1. La capa oculta es entrenada seleccionado los centros μ_j y los anchos σ_j de la función kernel asociada con las neuronas de la capa oculta.

2. Los pesos correspondientes a las conexiones entre las unidades ocultas y las salidas fijadas.

En las redes de base radial, la determinación del número de neuronas en la capa oculta es muy importante porque esta afecta la complejidad y la capacidad de generalización de la red. Si el número de neuronas en la capa oculta es insuficiente, la red RBF no puede aprender los datos adecuadamente. Por otro lado, si el número de neuronas es alto, pueden ocurrir problemas de pobre generalización o sobre-entrenamiento.

En la literatura, varios algoritmos han sido propuestos para entrenar las redes RBF, tales como: algoritmo de gradiente descendente, filtro de Kalman. Estos dos algoritmos están basadas en derivadas y tiene algunas deficiencias tales como convergencia a mínimos locales y consume mucho tiempo encontrar el gradiente óptimo. Dadas esas limitaciones, varios métodos de optimización global han sido usados para entrenar redes RBF; tales

como: algoritmos genéticos, particle swarm optimization (PSO) , sistemas inmune artificiales y algoritmos de evolución diferencial [52].

Como se indicó anteriormente, el entrenamiento de una red de base radial puede ser obtenido con la selección de los siguientes parámetros.

1. Los pesos de las conexiones entre la capa oculta y la capa de salida (w)
2. Parámetro de dispersión (desviación estadar) de las funciones kernel especificadas en las neuronas de la capa oculta (σ_j)
3. Los vectores Centro de las funciones kernel especificadas en las neuronas de la capa oculta (μ_j)
4. Los pesos de los bias a las neuronas de la capa de salida (w_bias, l).

El proceso de evolución de las redes de base radial para los juegos no cooperativos, consistió en evolucionar los pesos de las conexiones entre la capa oculta y la capa de salida, así como los vectores centro de las funciones kernel en las neuronas de la capa oculta. En la Fig. 3.5 se muestra una representación general de una red de base radial., dado que existen dos neuronas en la capa de entrada, el vector centro de cada neurona debe tener dos coordenadas que deben oscilar entre un valor máximo y mínimo que será obtenido experimentalmente, así como también el número de neuronas en la capa oculta.

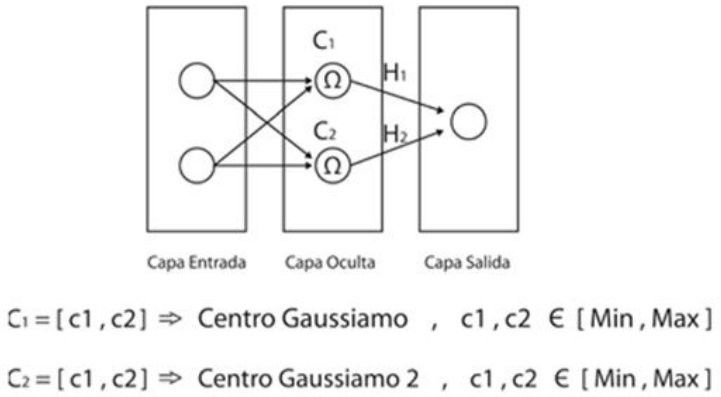


FIGURA 3.5. Evolución Red de base Radial.

En el caso específico del ejemplo, se deben evolucionar ocho pesos: $(H1, H2)$, que corresponden a los pesos de las interconexiones de las neuronas entre la capa oculta y la capa de salida. Adicionalmente, los pesos de los centros de las funciones kernel $(C1, C2)$ para cada una de las neuronas presentes en la capa oculta, y por último, el parámetro de dispersión (Desviación estándar) de las funciones Kernel especificadas en las neuronas de la capa oculta.

Con el fin de generalizar el esquema de evolución y permitir varias neuronas en la capa oculta, se puede determinar el número de pesos a evolucionar de acuerdo a la Ec. 3.11

$$W = ((H * O) + (H * I) + H) \tag{3.11}$$

Donde W es el número de pesos a evolucionar, I es el número de neuronas en la capa de entrada, H es el número de neuronas en la capa oculta, O el número de neuronas en la capa salida.

Aplicando la ecuación al caso presentado en la Fig. 3.5, obtenemos 3.11:

$$W = ((2) * 1) + ((2 * 2) + 2 = 2 + 4 + 2 = 8 \tag{3.12}$$

Es importante indicar, que el tamaño del cromosoma corresponde con el número de pesos a evolucionar. Para el caso de los pesos de las interconexiones entre las neuronas el rango de valores oscila entre *[-0.5, 0.5]*, mientras los valores de los centros de las Gausianas oscilan entre los valores máximo y mínimos que se especifique.

Así como en el perceptrón multicapa, las redes de base radial tendrán seis neuronas en la capa de entrada y una neurona en la capa de salida. El número de neuronas en la capa oculta, así como el valor máximo y mínimo entre los cuales oscilan los centros de las neuronas de la capa oculta, serán determinadas a través de procesos de experimentación.

3.3.3. Algoritmos Genéticos

Los Algoritmos Genéticos constituyen una técnica de búsqueda y optimización inherentemente paralela, inspirada en el principio Darwiniano de selección natural y adaptación. Bajo el enfoque de algoritmos genéticos se han desarrollado una serie de trabajos orientados a realizar procesos de evolución de estrategias para el dilema del prisionero iterado. Axelrod pionero en utilizar enfoques basados en computador, diseñó un entorno de simulación con el fin de estudiar la emergencia de la cooperación y el proceso de cambio de estrategias dentro de un marco de computación evolutiva, específicamente algoritmos genéticos [37]. Bajo este enfoque, cada cromosoma corresponde a una estrategia de juego a usar dentro del dilema del prisionero iterado.

Dado que convencionalmente la solución candidata a un problema dentro del esquema de algoritmos genéticos es codificada en una cadena de longitud fija, el trabajo de Axelrod determinó que utilizar una memoria de tres juegos previos era suficiente para decidir el próximo movimiento a ejecutar. Para cada juego del dilema del prisionero hay cuatro posibilidades: *C (Cooperar) C* , *CD (No cooperar)*, *DC*, *DD*, así que para un dilema del prisionero con una historia de tres hay $64(4^3)$ posibles historias.

Por lo tanto, una cadena de longitud 64 podría ser usada para representar la acción a tomar (C o D) basado en la historia de juegos previos.
Como no existe memoria para los tres primeros movimientos, se tomó en consideración el trabajo realizado por Errity [2]. Bajo este enfoque, el primer bit del cromosoma corresponde a la primera jugada, los dos siguientes bits corresponden con la segunda jugada, basado en si el oponente cooperó o no cooperó en el primer movimiento. Los bits 4, 5, 6, 7 indican

el tercer movimiento a ejecutar basado en los dos movimientos previos del oponente. Estos 7 Bits adicionados a los 64 propuestos por Axelrod determinan un cromosoma de longitud 71.

Definida una historia de juego a considerar para tomar las decisiones de juego y bajo un esquema de representación binaria, se puede determinar para cada bit la historia que le corresponde usando la Ec.(6):

$$IndicedelMovimiento = Mov.Iniciales + binEnt(MovHistorico) \tag{3.13}$$

Donde: *binEnt* corresponde a realizar la operación de convertir una representación binaria a número entero y *MovHistorico*, corresponde a los juegos previos de los jugadores, considerando como cooperación (C) el número binario 1 y como No-cooperación (D) el número cero.

De esta forma se conoce el bit dentro del cromosoma de cada uno de los movimientos históricos y por ende el movimiento a ejecutar en la repetición respectiva. En la Fig.3.6 , se presenta un ejemplo del proceso de elección de la jugada a realizar en una ronda basada en la historia de juego de cada jugador; para el caso particular, el movimiento a ejecutar en la próxima ronda corresponde a cooperar para el jugador A y no cooperar para el jugador B.

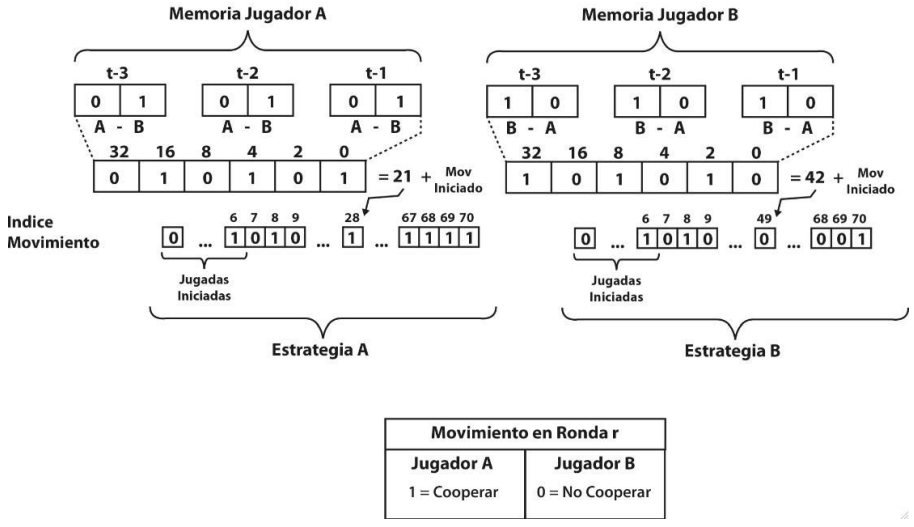


FIGURA 3.6. Evolución Red de base Radial.

3.3.4. Enjambre de Partículas (Particle Swarm Optimization)

PSO es una técnica meta heurística evolutiva desarrollada por Kennedy y Eberhart en 1995, inspirada en el comportamiento social de los enjambres en la naturaleza, como: pájaros, pescados, etc. En este algoritmo, la solución candidata es presentada como una partícula que se hecha a volar en el espacio de búsqueda. Con el fin de encontrar el óptimo

global cada partícula se desplaza a través de un espacio de búsqueda multidimensional, ajustando su posición de acuerdo a su propia experiencia y la de otras partículas en su vecindario, es decir, una partícula hace uso de la mejor posición encontrada por sí misma y de la mejor posición de su vecindario, para dirigirse hacia la solución óptima [27].

En general, el trabajo con enjambres requiere de una representación del problema mediante un vector de flotantes; cada vector es una estructura de datos que representa una de las posibles soluciones del espacio de búsqueda del problema.
Además, con el fin de evaluar cada partícula se debe definir una función de evaluación, la cual asigna un valor a cada posible solución codificada indicando la bondad de la solución.
La aproximación Kennedy and Eberhart puede ser detallada de la siguiente forma:

Suponga que la i-ésima partícula volando sobre un espacio del hiperplano, sea:

- $\vec{x_i}^k$, la posición de la partícula i en el instante k
- $\vec{v_i}^k$ velocidad de la partícula i en el instante k
- $pbest_i$ la mejor posición previa de i-ésima
- $gbest^k$ la mejor posición entre todas las partículas en el instante k

La próxima velocidad de vuelo y posición de la particula es actualizada para la iteración $k+1$ usando las ecuaciones 3.14, 3.15.

$$\vec{v_i}^{k+1} = \vec{v_i}^k + C_1 * rand_1() + (pbest_i - \vec{x_i}^k) + C_2 * rand_2() + (gbest^k - \vec{x_i}^k) \tag{3.14}$$

$$\vec{x_i}^{k+1} = \vec{x_i}^k + \vec{v_i}^{k+1} \tag{3.15}$$

donde, C_1 y C_2 son las tasas de aprendizaje cognitivo y social respectivamente. Estos dos parámetros, controlan la importancia relativa de la memoria (Posición) de la partícula en sí a la memoria de la vecindad y son frecuentemente definidos con el mismo valor para dar igual peso a ambas tasas.
Las variables $rand_1()$ y $rand_2()$ son funciones aleatorias que están uniformenmente distribuidas en el rango [0,1].

Sin embargo, la versión original del PSO desarrollada por Kennedy and Eberhart es efectiva en la determinación de soluciones óptimas en ambientes estáticos, pero sufre resultados pobres cuando existen cambios extremos.

Por lo tanto, en 1998, Shi y Eberhart mostraron que PSO busca en amplias zonas eficientemente, pero tiende a perder precisión en la búsqueda local.

Ellos introdujeron un parámetro de control llamado el peso de la inercia w, para amortiguar la velocidad a través del tiempo, permitiendo al enjambre converger más exacta y eficientemente.

La modificación del PSO para actualizar el vector velocidad es reformulada como sigue.

$$\vec{v_i}^{k+1} = W * \vec{v_i}^{k} + C_1 * rand_1() + (pbest_i - \vec{x_i}^{k}) + C_2 * rand_2() + (gbest^k - \vec{x_i}^{k}) \quad (3.16)$$

$$\vec{x_i}^{k+1} = \vec{x_i}^{k} + \vec{v_i}^{k+1} \quad (3.17)$$

Analizando la Ecuación 3.16, se evidencia que una gran inercia facilita la exploración global mientras un valor pequeño, facilita la búsqueda local.

El esquema de codificación propuesto para representar el vector solución de cada partícula fue igual al propuesto bajo el esquema de algoritmos genéticos. Por lo tanto, se tendrá un cromosoma de igual longitud 71 (adaptable dinámicamente) y se usará el mismo esquema de evaluación de la solución.

3.4. Resultados Evolución de Estrategias

Para el dilema del prisionero, juego de la gallina y caza del ciervo, cuando el juego es repetitivo, existen una serie de estrategias estándar que han sido creadas por seres humanos para diversos torneos, por lo tanto, en la literatura son utilizadas como posibles estrategias contrincantes en procesos de evolución.

Se usaron estrategias estándar utilizadas en competencias del dilema del prisionero CEC04 [10], dichas estrategias son de diversos tipos: estrategias que deciden la jugada basado en los movimientos previos de los oponentes (TFT, GRIM, TFTT, STFT, NEG), estrategias que deciden la jugada sin importar los movimientos previos (ALLC, ALLD), estrategias que cuyas decisiones son tomadas aleatoriamente (RAND) y por último estrategias que inician aleatoriamente y posteriormente se adaptan basado en los resultados parciales que vayan obteniendo (PAVLOV). Las estrategias RAND y PAVLOV son utilizadas con el fin de verificar la capacidad de las estrategias evolutivas frente al ruido.

Con el fin de hacer una comparación cuantitativa entre las diversas técnicas de evolución, se usará como métrica de comparación, métrica basada en recompensa [55], por lo tanto, se buscará obtener a través del proceso de evolución, maximizar el pago promedio en los diferentes tipos de juegos.

El proceso de experimentación para cada una de las técnicas es realizado usando como parámetros la mejor combinación obtenida a través de los procesos de experimentación realizados. En las tablas 3.4, 3.5 , 3.6 y 3.7, se presenta la configuración de parámetros usada para cada una de las técnicas. Es importante indicar que se usó para los diversos esquemas de evolución, operador de cruce monopunto y esquema de mutación bit a bit. Adicionalmente, el número de rondas establecido en los enfrentamientos entre estrategias corresponde a 100

Los resultados se dividen en dos partes, en primera instancia se procedió a realizar juegos estándar para cada uno de los dilemas estudiados, obteniendo como conclusión

TABLA 3.4. Parámetros Perceptrón

Selección	%Cruce	%Mutación	Núm. capas Ocultas	Fun. Activación	Neuronas en la capa Oculta.
Torneo	0.7	0.2	1	Tang.Hiperbólica	20

TABLA 3.5. Parámetros de la Red de Base Radial

Selección	%Cruce	%Mutación	Núm. Neuronas	Max-Min
Torneo	0.7	0.2	20	(-10,10)

que todas las técnicas convergen a la maximización del pago cuando compiten con las estrategias del CEC04. Posteriormente, se presentan los resultados en torneos con y sin ruido.

3.4.1. Juego Estándar

1. Dilema del Prisionero.

   En las Fig. 3.7, 3.8, 3.9, 3.10 se presentan los resultados de los juegos estándar para perceptron multicapa, red de base radial, algoritmos genéticos y PSO respectivamente.

   Con los algoritmos genéticos, PSO, Perceptrón multicapa, red de base radial si el contrincante tiene la estrategia ALLC, se obtiene como resultado de la evolución, la no-cooperación dado que se maximiza la recompensa. Es decir, si no se coopera se maximiza el pago promedio obtenido que corresponde a 5 en cada juego de acuerdo con tabla de pagos. Dicho resultado también es equivalente en caso que el contrincante decida jugar con NEG, en este caso al no cooperar, el contrincante siempre decidirá cooperar.

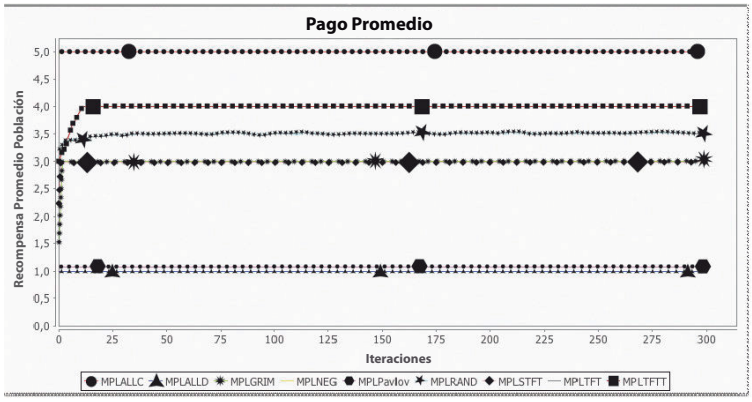


FIGURA 3.7. Resultados juego estándar Perceptrón

Es interesante ver, adicionalmente, que en todas las técnicas utilizadas para la evolución de estrategias, si el contrincante juega con ALLD, PAVLOV, la mejor decisión para maximizar el pago es no-cooperar, esto lo refleja el pago promedio obtenido de 1. El resultado con Pavlov sucede dado que siempre está buscando maximizar la

Tabla 3.6. Parámetros Algoritmo Genético

Selección	%Cruce	%Mutación	Evoluciones	Tam. Población
Torneo	0.7	0.2	300	500

Tabla 3.7. Parámetros PSO

Apren. Social	Apren. Cognitivo	Tasa Inercia	Partículas	Tam. Vecindario inicial	%Incre. Vecindario	Iteraciones
0.7	0.9	0.9	200	10	0.7	200

recompensa y a diferencia de las otras estrategias puede cambiar su estrategia en el juego, luego la mejor decisión es buscar la estrategia dominante (No cooperación) que coincide en este caso con el equilibro de Nash.

En todas las técnicas para el caso cuando el contrincante juega con TFT, STFT, GRIM emerge la cooperación mutua (Pago promedio de 3.0), dado que ambos jugadores maximizan la recompensa obtenida, lo cual es consistente con la hipótesis planteada inicialmente. Adicionalmente, si el contrincante juega RAND, según los resultados obtenidos todas las técnicas se adaptan y según el puntaje promedio obtenido busca la cooperación mutua, lo mismo sucede con la estrategia TFTT (Que puede llegar a perdonar una cooperación, pero luego de dos se comporta como un ALLD).

2. Juego de la Gallina.

   En las Fig. 3.11, 3.11 se presentan los resultados de los juegos estándar para perceptron multicapa, red de base radial respectivamente.

   En todas las técnicas de evolución de estrategias, si el contrincante juega ALLC, PAVLOV, NEG luego la mejor decisión en todos los casos es no cooperar (No desviarse); es decir el contrincante siempre va ser una gallina. Lo anterior garantiza maximizar el pago obtenido. Lo que sucede con NEG y PAVLOV es que son estrategias que son dependientes de la decisión del otro jugador, por lo tanto; para NEG

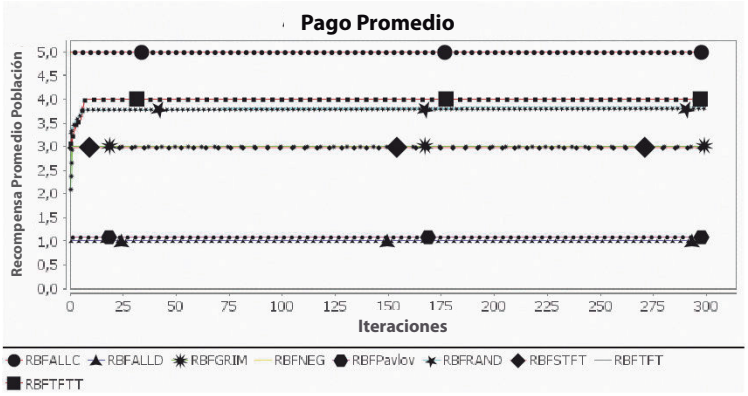


Figura 3.8. Resultados dilema juego estándar Red base radial

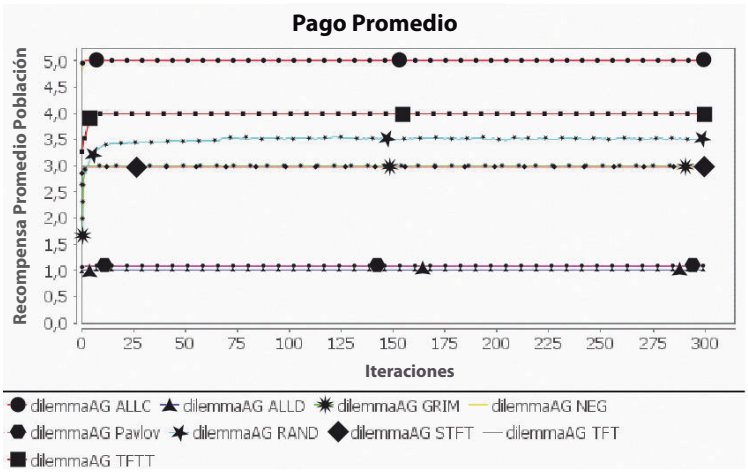


FIGURA 3.9. Resultados juego estándar Alg. genéticos

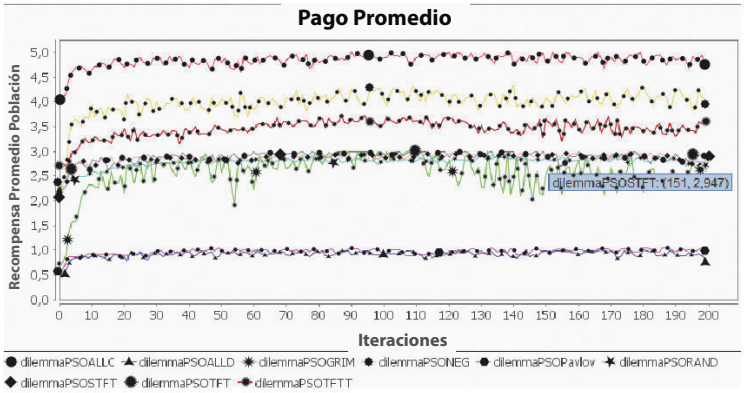


FIGURA 3.10. Resultados juego estándar PSO

si se juega No-cooperación, el contrincante va jugar siempre la estrategia Cooperar, luego va ser un gallina. En Pavlov, sucede algo similar, si se juega no-cooperación, y dado que se está buscando mejorar su puntaje, lo mejor que puede hacer es cooperar para obtener una recompensa de 1, si decidiera no cooperar el pago sería de 0.

Si el contrincante juega con ALLD (Nunca se desvía), la estrategia dominante que debe encontrar cada una de las técnicas de evolución corresponde a desviarse (Cooperar). Si no se coopera (No desviarse), ambos jugadores se destruyen y no reciben recompensa, luego si se determina que el contrincante nunca se va desviar, la mejor decisión es desviarse (Es mejor ser una gallina en todos los casos con el fin de obtener algún beneficio).

Para el caso cuando el contrincante juega con TFT, STFT, GRIM emerge la cooperación mutua, dado que ambos jugadores maximizan la recompensa obtenida. Obteniendo un resultado similar al del dilema del prisionero.
Si el contrincante juega RAND, según los resultados obtenidos, todas las técnicas se adaptan satisfactoriamente y según el puntaje obtenido busca la cooperación cuando el contrincante coopera o no-cooperación cuando se cree que el contrincante no va a cooperar, lo mismo sucede con la estrategia TFTT (Que puede llegar a perdonar una cooperación, pero luego de dos se comporta como un ALLD).

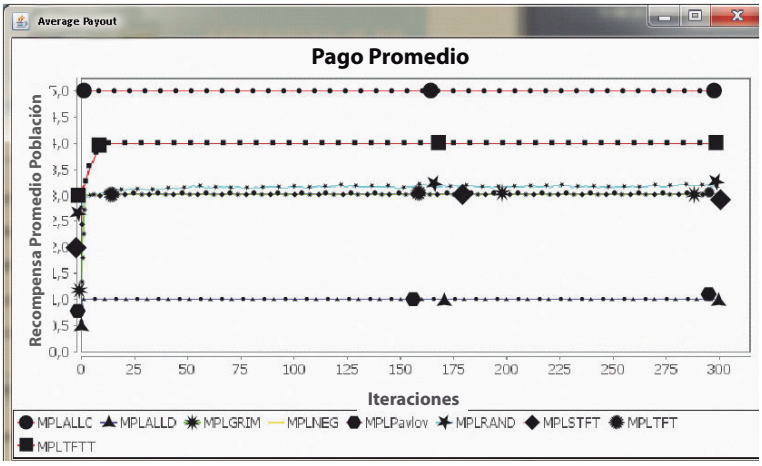


FIGURA 3.11. Resultados juego estándar Perceptrón

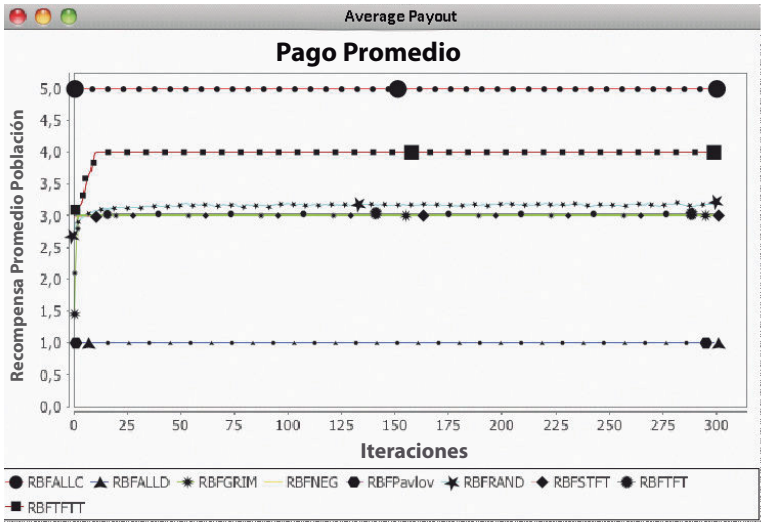


FIGURA 3.12. Resultados Gallina juego estándar Red base radial

3. Caza del Ciervo

   En las Fig. 3.13, 3.14 se presentan los resultados de los juegos estándar para perceptron multicapa y red de base radial respectivamente.

   Para todas las técnicas de evolución, si el contrincante juega ALLC, TFT, TFTT, STFT, GRIM; luego la mejor decisión en todos los casos es cooperar; es decir buscar cazar el ciervo (Confianza mutua) dado que se maximiza la recompensa esperada. Adicionalmente, si el contrincante juega con ALLD, PAVLOV, la mejor decisión para maximizar el pago es no-cooperar, esto lo refleja el pago obtenido promedio de 1.

   En todos los casos, si el contrincante decide jugar RAND o NEG, el resultado es un promedio entre cooperar y no cooperar, lo cual es lógico. El algoritmo busca adaptarse de acuerdo a su estrategia dominante (Cooperar cuando, el contrincante cooperar; o no cooperar cuando se cree que el contrincante no va a cooperar).
   Con el fin de completar el análisis de los resultados, a continuación se procede a

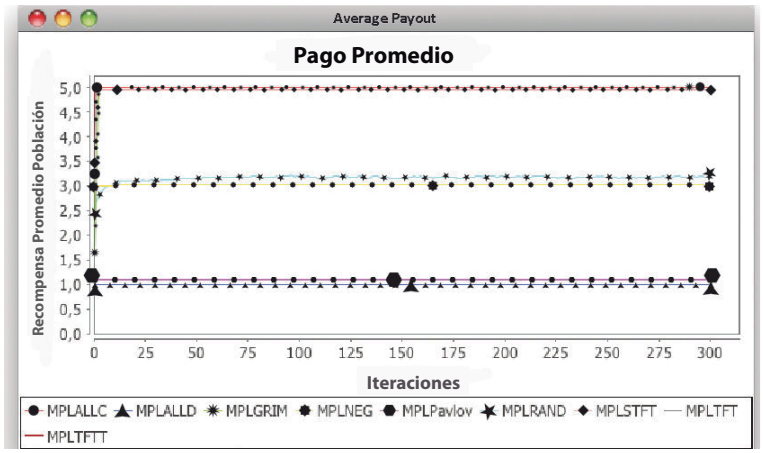


FIGURA 3.13. Resultados juego estándar Perceptrón

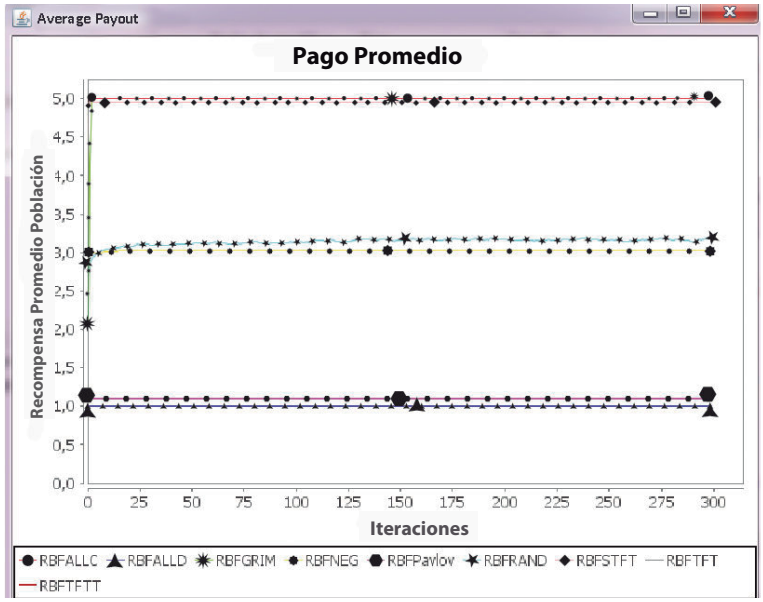


FIGURA 3.14. Resultados Ciervo juego estándar Red base radial

hacer la comparación de las diferentes técnicas cuando se experimenta con torneos; es decir, todas las técnicas compiten entre sí.

En primera instancia se procedió a realizar juegos estándar, obteniendo como conclusión que todas las técnicas convergen a la maximización del pago cuando compiten con las estrategias del CEC04. A continuación se presenta un resumen de los resultados obtenidos.

Dilema el Prisionero.

- Con todas las técnicas, si el contrincante juega con ALLD, PAVLOV, la mejor decisión para maximizar el pago es no-cooperar, esto lo refleja el pago promedio obtenido de 1.

- Cuando el contrincante juega con TFT, STFT, GRIM emerge la cooperación mutua (Pago promedio de 3.0), dado que ambos jugadores maximizan la recompensa obtenida.

En el Juego de la Gallina

- Si el contrincante juega ALLC, PAVLOV, NEG luego la mejor decisión en todos los casos es no cooperar (No desviarse); es decir el contrincante siempre va ser una gallina.
- Si el contrincante juega con ALLD, la mejor decisión es desviarse (Es mejor ser una gallina en todos los casos con el fin de obtener algún beneficio).
- Si el contrincante juega con TFT, STFT, GRIM emerge la cooperación mutua, dado que ambos jugadores maximizan la recompensa obtenida.

En la caza del Ciervo.

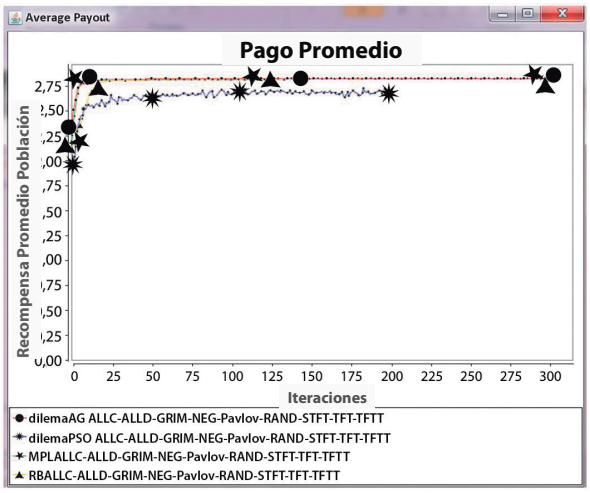
- Si el contrincante juega ALLC, TFT, TFTT, STFT, GRIM; luego la mejor decisión en todos los casos es cooperar; es decir buscar cazar el ciervo (Confianza mutua) dado que se maximiza la recompensa esperada.
- Si el contrincante juega con ALLD, PAVLOV, la mejor decisión para maximizar el pago es no-cooperar, esto lo refleja el pago obtenido promedio de 1.

Con el fin de completar el análisis de los resultados a continuación se procede a hacer la comparación de las diferentes técnicas cuando se experimenta con torneos; es decir, todas las técnicas compiten entre sí.

3.4.2. Torneos

En las Fig. 3.15, 3.16 y 3.17 se pueden observar como los resultados evidencian la emergencia de la cooperación dado los pago promedios obtenidos (Dilema del Prisionero: Aprox 2.75, Juego de la gallina: Aprox 3.0 y Caza del Ciervo: Aprox 4.0) para cada de las técnicas: Perceptrón (Estrella) , Red de base Radial (Triángulo), Algoritmo genético (Círculo) y PSO.

Como se observa en los gráficos, las redes neuronales obtienen resultados similares a los alcanzados por los algoritmos genéticos. En todos los juegos el perceptron tiene un leve puntaje mayor que las redes de base radial, sin embargo la diferencia no es significativa. Lo anterior, da sustento a la hipótesis inicial la cual indica que las redes neuronales evolucionadas pueden realizar aprendizaje de estrategias en juegos a pesar de actuar en ambientes con ruido. Adicionalmente, los resultados son consistentes con los obtenidos en los algoritmos genéticos en diferentes trabajos en la literatura.



AG Evolution	PSO	MPL	RBF
AG=2866.0	GRIM=2780.0	**NN=2806.0**	**Radial=2795.0**
GRIM=2780.0	**PSO=2769.0**	GRIM=2792.0	GRIM=2792.0
Pavlov=2637.0	Pavlov=2603.0	TFT=2593.0	TFT=2597.0
TFT=2590.0	TFT=2588.0	TFTT=2548.0	TFTT=2559.0
TFTT=2547	TFTT=2573.0	Pavlov=2400.0	Pavlov=2418.0
ALLC=2238.0	ALLC=2259.0	ALLC=2244.0	ALLC=2247.0
ALLD=2028.0	ALLD=2008.0	STFT=2014.0	ALLD=2032.0
STFT=2014.0	STFT=2002.0	ALLD=2004.0	STFT=2007.0
NEG=1983.0	RAND=1892.0	NEG=1913.0	NEG=1940.0
RAND=1831.0	NEG=1677.0	RAND=1729.0	RAND=1799.0

FIGURA 3.15. Resultados Torneo Dilema del Prisionero

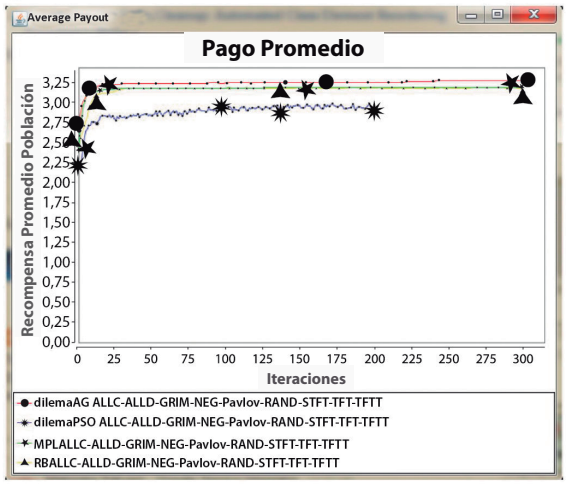
3.5. Resumen

Se presenta la evolución de redes neuronales para el aprendizaje de estrategias de decisión en juegos no cooperativos repetitivos. A partir de los resultados se pueden extraer las siguientes elementos:

Los perceptores multicapa como las redes de base radial obtenidas por proceso de evolución vencen a estrategias estándar tradicionalmente utilizadas en la literatura, adicionalmente se puede evidenciar en cada uno de los juegos no cooperativos la emergencia de la cooperación cuando se maximiza el puntaje. Es decir, los agentes tiene la capacidad de interpretar las reglas de juego y producto de ello, generar y retener estrategias que garantizan la maximización del beneficio.

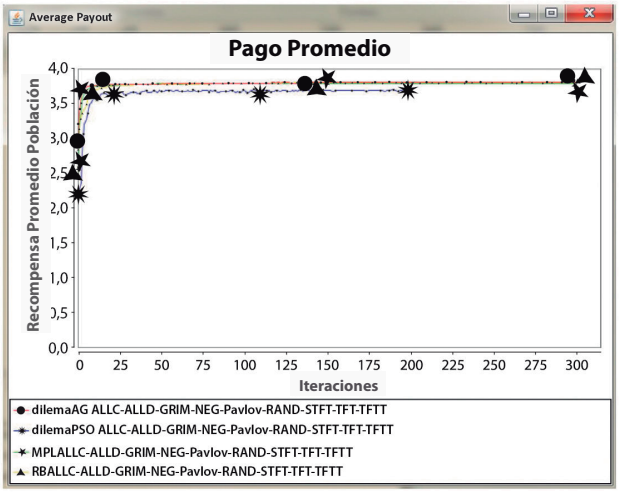
Las redes neuronales (perceptron, base radial) tienen comportamientos similares, se evidencia en los gráficos de comportamiento basado en la recompensa obtenida, cuando compiten en juegos no cooperativos bajo la modalidad torneo.

Las redes neuronales tienen comportamientos equivalentes a los algoritmos genéticos y enjambre de partículas ampliamente estudiados como estrategias de evolución. Lo anterior se puede evidenciar dado que los gráficos de comportamiento convergen siguiendo la misma tendencia.



AG Evolution	PSO	MPL	RBF
AG=3199.0	**PSO=3091.0**	**NN=3180.0**	**Radial=3072.0**
GRIM=2728.0	GRIM=2770.0	TFT=2536.0	GRIM=2543.0
TFT=2534.0	TFT=2531.0	GRIM=2530.0	TFT=2531.0
Pavlov=2496.0	Pavlov=2513.0	TFTT=2519.0	TFTT=2518.0
RAND=2419.0	TFTT=2421.0	Pavlov=2505.0	ALLC=2502.0
ALLC=2292.0	RAND=2390.0	ALLC=2498.0	Pavlov=2494.0
ALLD=2265.0	ALLC=2318.0	ALLD=2240.0	ALLD=2295.0
TFTT=2233.0	ALLD=2275.0	NEG=2221.0	NEG=2224.0
NEG=2201.0	NEG=2223.0	RAND=2163.0	RAND=2192.0
STFT=1951.0	STFT=1942.0	STFT=1960.0	STFT=1949.0

FIGURA 3.16. Resultados Torneo Juego de la Gallina



AG Evolution	PSO	MPL	RBF
TFTT=4029.0	TFTT=4034.0	TFTT=4017.0	TFTT=4005.0
AG=3751.0	**PSO=3747.0**	**NN=3760.0**	**Radial=3700.0**
TFT=3720.0	TFT=3734.0	ALLC=3740.0	TFT=3698.0
ALLC=3720.0	ALLC=3745.0	GRIM=3702.0	Pavlov=3509.0
GRIM=3694.0	GRIM=3690.0	TFT=3650.0	GRIM=3295.0
Pavlov=3516.0	Pavlov=3297.0	Pavlov=2901.0	ALLC=3280.0
STFT=2481.0	STFT=2523.0	STFT=2508.0	STFT=2485.0
RAND=2151.0	NEG=2198.0	NEG=2010.0	RAND=2200.0
NEG=1541.0	RAND=2152.0	RAND=1924.0	NEG=1518.0
ALLD=1514.0	ALLD=1506.0	ALLD=1514.0	ALLD=1514.0

FIGURA 3.17. Resultados Torneo Caza del Ciervo

CAPÍTULO 4

Laboratorio Computaciona en el ámbito de la economía evolucionista

4.1. Introducción

En el presente trabajo con el fin de desarrollar un laboratorio computacional para la especificación de simulaciones en el ámbito de la economía evolucionista, se tendrán en cuenta las ideas introducidas por Dopfer et al [24] en el cual, se plantea la idea de definir un sistema económico en términos de reglas. En las aproximaciones tradicionales, las reglas son preconfiguradas en el agente, significa que el conjunto de acciones de los agentes no cambian durante el proceso de simulación, sin embargo, se han venido introduciendo mecanismos de aprendizaje utilizando técnicas de aprendizaje reforzado, redes neuronales, programación evolutiva (algoritmos genéticos, programación genética, programación genética gramatical) entre otras, con el fin de cambiar el conjunto de reglas a medida que avanza el proceso de simulación.

El desarrollo de simulaciones basadas en agentes en el contexto económico (Economía computacional basada en agentes), requiere comprender o construir una plataforma de modelamiento con el fin de configurar los diferentes sistemas que se quieren simular. Actualmente, existen una variedad de plataformas de simulación basada en agentes (StartLogo, NetLogo, MASON, etc.), sin embargo, para la implementación de modelos se deben conocer elementos sintácticos del lenguaje complicando la tarea para la implementación. Adicionalmente, proveer a los agentes mecanismos de adaptabilidad a partir de proceso de aprendizaje no están en el núcleo central de dichas plataformas y si lo están, requieren de cierta experticia en programación de computadores y las técnicas de aprendizaje particulares que se deseen aplicar [36].

Con el fin de dar una solución a lo anterior, varios trabajos de investigación se han realizado, entre los cuales se encuentra el lenguaje planteado por Okuyama et al [65]. A pesar que es un lenguaje más simple que los provistos por las plataformas de simulación mencionadas anteriormente, tiene elementos que puede dificultar la especificación de modelos: no existe un entorno de desarrollo integrado que facilite las tareas de programación, no permite la integración con librerías de aprendizaje externas.

El capítulo presenta el diseño e implementación de un laboratorio computacional para la especificación de simulaciones basadas en agentes, usando un lenguaje basado en reglas. El capítulo está organizado de la siguiente manera: en la sección I se presenta el diseño conceptual del laboratorio computacional, en la sección II se muestra el lenguaje específico de dominio para la especificación de modelos de simulación, el entorno de desarrollo integrado es presentado en la sección III.

Por último, se presentan modelos de simulación para juegos repetitivos no cooperativos, los cuales integran estrategias de aprendizaje basado en redes neuronales artificiales.

4.2. Laboratorio Computacional

Buscando un esquema unificado para la especificación de simulaciones económicas, se ha planteado el desarrollo de un entorno de simulación bajo la propuesta de agentes inteligentes planteado por Russell et al [69], para lo cual es necesario definir la arquitectura y el programa de cada uno de los agentes involucrados en un proceso de simulación. La arquitectura debe definir los sensores, actuadores, entorno y la medida de rendimiento. Por su parte, el programa debe implementar la función del agente, es decir, definir las acciones a ejecutar basado en las percepciones, el estado interno, la función de utilidad esperada en cada juego y los mecanismos de aprendizaje.

En el caso de la economía computacional basada en agentes, cada agente (Arquitectura Micro) participante debe tener la capacidad de percibir: las decisiones del contrincante basado en las reglas de decisión utilizadas en cada una de las iteraciones, la recompensa entregada por el entorno en función de la tabla de recompensas alterando de esta forma el comportamiento (Arquitectura Macro).

Dicha información debe ser almacenada temporalmente en un estado interno con capacidad limitada que posee cada agente, con el fin que el módulo de aprendizaje pueda evaluar el resultado obtenido y basado en criterios de calidad y experiencia modificar los elementos de actuación definidos, es decir, modificar las reglas de tal manera que se pueda mejorar la selección de acciones (retención de reglas) buscando optimizar las medidas de rendimiento.

El esquema conceptual planteado para el laboratorio computacional es presentado en la Fig. 4.1.

Como se puede observar el laboratorio computacional tiene los siguientes componentes:

1. Lenguaje Específico de Dominio para la especificación de los modelos de simulación.
2. Plataforma de Simulación basada en agentes que soporte los planteamientos especificados por Russell et al [69]. Los agentes envían las decisiones al ambiente en cada una de las iteraciones del juego, las decisiones pueden ser tomadas con base en los resultados producidos por estrategias de aprendizaje que usan como fuente de información el historial de percepciones de cada uno de los agentes.
   El ambiente en la plataforma de simulación, recibe las acciones de cada uno de los agentes y basado en las reglas configuradas realiza cada uno de los pagos a los respectivos agentes.
3. Reporte de resultados de las simulaciones especificados

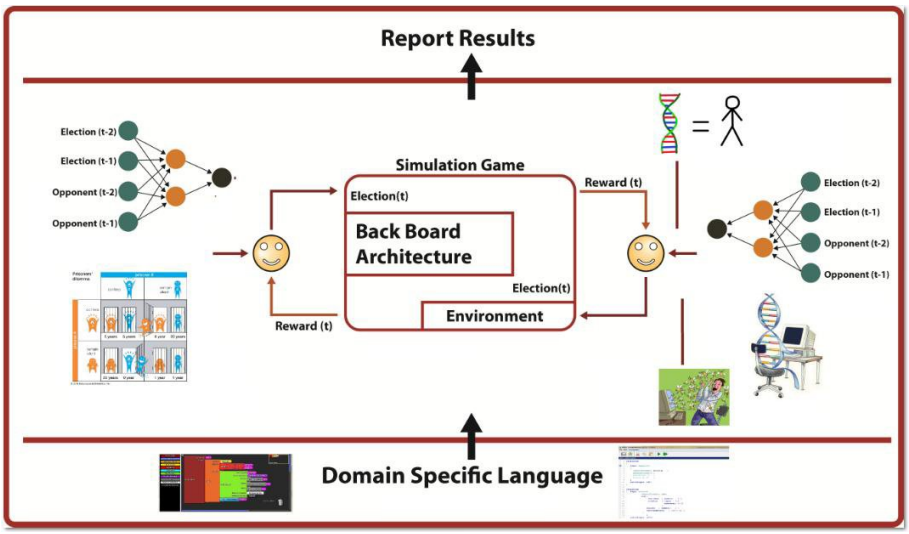


FIGURA 4.1. Esquema conceptual plataforma de simulación

A continuación se presenta cada uno de los elementos especificados anteriormente para el laboratorio de simulación construido, el cual se denominó *UNALCOL*.

4.3. Lenguaje específico de dominio.

Con el fin de poder realizar la especificación de las simulaciones se planteó un lenguaje específico de dominio textual. Un lenguaje especifico se compone internamente de varias etapas, entre ellas podemos encontrar: analizador léxico, analizador sintáctico, analizador semántico, generador de código, entre otros [73].

1. **Analizador léxico:** Se definieron expresiones regulares para las siguientes categorías léxicas:

   - Palabras reservadas de alto nivel.
   - Palabras reservadas que identifican los tipos de datos
   - Símbolos del lenguaje.

   En la Fig 4.2, se presenta los token identificados.

   Posterior a la definición de las expresiones regulares, se procedió a especificar el analizador sintáctico. Los componentes de la gramática son:

   - **Símbolo inicial de la gramática:** Unalcol.
   - **Símbolos no terminales:** Unalcol, Agents, Estructure, Asignation, Attributes, Condition, etc.
   - **Reglas de producción**

   En primera instancia, un modelo de simulación en *UNALCOL* debe tener configurados: Los agentes, el ambiente y los parámetros de configuración para ser utilizados en la ejecución de la simulación.

```
TOKEN : /* KERWORDS */
{
  < TYPE: "type" > | < AGENT: "agent" >  | < ATTRIBUTE : "attribute" > | < PERCEPT:
"percept" >| < ACTION : "action" >| < PROGRAM: "program" >| < MAIN: "main" > | < END:
"end" > | < AS: "as" > | <ENV:"enviroment" > | < CONFIGURATION: "configuration" > | <
MAXITERATION: "max" > | < SINCRONIZED: "sincronized" > | < TRUE  : "true" > | < FALSE :
"false" > | <NAME: "name" > | < FUNCTION: "function" >| < IMPORT : "import" >
}

TOKEN : /* DATATYPE */
{
  <STRING: "string"> | <DOUBLE:"double"> | <INTEGER: "integer">| < BOOLEAN:"boolean"> | <
PERCEPTIONS : "perception">
}


TOKEN : /* STRUCTURES */
{
  < IF : "if"> | < THEN : "then"> | < ELSE: "else"> | < TO : "to"> | <DIM : "dim">
 |<AND : "and"> | < OR : "or"> | < NOT : "not"> | < IN : "in"> | < NULL : "null">
}

TOKEN : /* SIMBOL */
{
  < DP: ":"> | < CO: ","> | < PT: "."> | < EQ: "="> | < PA: "("> | < PC: ")">
| < AT: "@">| < MINUS: "-"> | < PLUS: "+"> | < MULTIPLY: "*"> | < DIV: "/">
| < LA: "["> | < LC: "]"> | < AD: "!"> | < MY: ">"> | < MN: "<">
}

TOKEN :{
  < ID: < LETRA > (< LETRA > | <DIGIT>)* > | < #LETRA: [ "a"-"z" ] >
}
```

FIGURA 4.2. Expresiones regulares lenguaje

Para la definición del cuerpo del agente se tienen en cuenta la regla de Producción *Agents*, la cual está conformada por los siguientes componentes.

- **Integración con librerías externas para el proceso de toma de decisiones.**
  En este caso, se debe especificar la clase principal que cumple con las restricciones de la plataforma con el fin de ser utilizada para procesos de toma de decisiones.
  Específicamente, debe extender de la clase *Learning* y definir el método *compute* como se explicará más adelante. Lo anterior, con el fin de utilizar las funciones de la librería Common Discovery del proyecto Apache, que permite el descubrimiento dinámico de clases que implementen una Interface o extiendan una clase abstracta dada.
  En la gramática el componente léxico que indica el inicio de la definición de las librerías es *IMPORT*
- **Definiciones de atributos del agente:** Representa las propiedades internas que definen el agente. Dichos atributos pueden ser observados por el ambiente o por otros agentes. Cada atributo puede tener los siguientes tipos de datos: *Integer*, *Double*, *Boolean*, *String* y *Perception*.
  El tipo de dato *Perception* es un tipo de datos definido dentro del lenguaje *UNALCOL* con el fin de comunicar las decisiones y pagos por parte del agente y el ambiente.
  Un tipo de dato *Perception* está conformado por los siguientes elementos: identificador único del agente, iteración, acción a ejecutar en el ambiente y la recompensa entregada por el ambiente basado en la acción a ejecutar. Por su parte, los restantes atributos tienen correspondencia directa con los tipos de datos *Integer*, *Double*, *Boolean* y *String* del lenguaje de propósito general Java.
  En la gramática, el componente léxico que indica el inicio de la definición de atributos es *ATTRIBUTE*.

Es importante indicar que por defecto un agente tiene los siguientes atributos: Id, cx (coordenada x), cy (Coordenada Y), color.

- **Definición de percepciones del agente:** Son las características observables del ambiente por parte del agente o los atributos propios que son usados para la toma de decisiones. Dado lo anterior, cuando se define una percepción se debe especificar el origen del atributo a utilizar; es decir, si el atributo hace parte del ambiente o del propio agente.
  En la gramática el componente léxico que indica el inicio de la definición de percepciones es *PERCEPT*.

- **Definición de acciones:** Son las posibles decisiones que puede tomar el agente en el ambiente. En la gramática el componente léxico que indica el inicio de la definición de percepciones es *ACTION*.

- **Definición del Programa del agente:** Se definen las reglas de decisión basado en las percepciones. Es importante indicar, que en las reglas de decisión se puede hacer uso de las la librerías definidas como librerías externas en la sección de *IMPORT*.

Como se observa en la gramatica planteada en el algoritmo 1, las reglas de decisión deben ser especificadas en términos de reglas *IF-THEN-ELSE*.

Dentro de la definición de cada una de las reglas, se debe definir el tipo de regla, entre los cuáles se encuentran: *@PROGRAM*, *@ACTION*, *@PERCEPT*.
Si se especifica el metadato *@PROGRAM* en la definición de la regla, se indica que dicha regla corresponden al programa que define la acción que debe llevar a cabo el agente.

En caso que se especifique el metadato *@ACTION*, indica al agente, cuales son los cambios internos que se deben ejecutar producto de la decisión tomada. Los cambios se ven reflejados en los valores de los diferentes atributos que definen al agente.

Por último, en caso que se especifique *@PERCEPT*, indica al agente las reglas a aplicar luego de obtener a través de los sensores las características observables entregados por el ambiente.

Otro de los elementos que deben ser definidos para la realización de una simulación en el lenguaje *UNALCOL*, es el ambiente, el cual queda completamente definido a través de los siguientes componentes 2.

- Atributos del ambiente.
- Programa: Tal como se indicó anteriormente, el ambiente recibe las decisiones tomadas por cada uno de los agentes en cada periodo de simulación y a partir de esto, da a cada uno de ellos una recompensa.

Algorithm 1 Regla de producción para la definición del agente

Unalcol $\rightarrow$
(**Agent**) $*$ **EnviromentStructure Main**
Agent $\rightarrow$
$< AGENT >< ID >$
(
$< IMPORT >< DP > (< ID >< AS >< ID > (< PT >< ID >)*)+)?$
$< ATTRIBUTE >< DP >$
(
$< ID > (< PA >< CONSTANT >< PC >)?$
$< AS > (< STRING > | < DOUBLE > | < INTEGER >< BOOLEAN >< PERCEPTIONS > | < AGENT > | < ID >$
)+
$< PERCEPT >< DP > (< ID >< IN > (< ENV > | < AGENT >))+$
$< ACTION >< DP >< ID > (< CO >< ID >)*$
$< PROGRAM >< ID >< DP >$ (**Estructure**|**AsignationsAttibutes**)+
$< ENG >< AGENT >$
Estructure $\rightarrow$
$(< AT > (< PROGRAM > | < ACTION > | < PERCEPT >))?$
$< IF >$ **Condition** $< THEN >$ ((**AsignationAttibutes**) + |**Estructure**)
(
$< ELSE >$ ((**AsignationAttibutes**) + |**Estructure**))?
$< END >< IF >$
Condition $\rightarrow$
Asignation$((< AND > | < OR > | < NOT >)$**Asignations**$)*$

Algorithm 2 Regla de producción para la definición del ambiente

$EnviromentStructure \rightarrow$
$< ENV >$
$(< ID > (< PA >< PC >)? < AS >$
$< STRING > | < DOUBLE > | < INTEGER > | < BOOLEAN > | < PERCEPTIONS > | < AGENT > | < ID >)$
)+
$< PROGRAM >< ID >< DP > (Estructure|Asignations)+$
$< END >< ENV >$

Por último, se debe especificar la configuración de la simulación, configuración que será tenida en cuenta en el proceso de ejecución del modelo. Entre los elementos a especificar están:

- El nombre del modelo.
- El número de iteraciones a ejecutar el modelo de simulación.
- Modo de ejecución: Especificación si se requiere que el modelo se ejecute en modo síncrono o asíncrono. Si se especifica que el modelo sea ejecutado de modo síncrono, indica que el ambiente debe esperar a que cada uno de los agentes envíen las decisiones. Luego de recibirlas, debe proceder a realizar los pagos a cada uno de ellos basado en las reglas de decisión especificadas en el programa.
- Declarar cada uno de los agentes y realizar la inicialización de los respectivos atributos, tal como se presenta en el algoritmo 3.

Algorithm 3 Configuración modelo de simulación

$Main \rightarrow < MAIN > (Definitions) * (Iniciations) *$
(
$< CONFIGURATION >< PT >< NAME >< EQ >< ID >$
$< CONFIGURATION >< PT >< MAXITERATION >< EQ >< CONSTANT >$
$< CONFIGURATION >< PT >< SINCRONIZED >< EQ > (< TRUE > | < FALSE >)$
)?
$< END >< MAIN >$
$Definitions \rightarrow < DIM >< ID >< AS >< ID >$
$Iniciations \rightarrow$
$(< ID > | < ENV >) < PT >< ID >$
$< EQ >$
$((< ID > | < CONSTANT >)$
$| < LA >< ID > (CO >< ID >) * < LC >$
)

2. **Analizador semántico:** El objetivo principal es validar que las cadenas derivadas a partir de la gramática tiene sentido dentro del lenguaje. Entre las validaciones realizadas se tienen: comprobación de tipos (operadores y operandos deben ser compatibles), comprobación de unicidad (objetos que deben definirse una única vez), etc.

   Las validaciones semánticas especificadas en *UNALCOL* están:

   - El identificador del agente debe ser único. Es decir que no deben existir diferentes clases de agentes con el mismo nombre.
   - No deben existir identificadores de atributos, percepciones, import y acciones en la definición del agente con nombres equivalentes.
   - En la definición de las reglas en los agentes, se deben usar solo elementos especificados en las percepciones.

- Las asignaciones realizadas como resultado de la aplicación de las reglas, solo pueden afectar atributos definidos en el agente.
- No deben existir atributos en el ambiente con nombre equivalente.
- Solo se pueden declarar tipo de agentes de los tipos definidos en la especificación del programa.
- Solo se pueden inicializar atributos que existan en la definición de cada tipo de agente.

En caso que no existan errores léxicos, sintácticos y semánticos se procede a realizar un proceso de generación de código con el fin de obtener un programa que cumpla con las restricciones de la plataforma de simulación desarrollada para la ejecución de modelos en *UNALCOL*.

En la siguiente sección se procederá a especificar los componentes principales para la especificación de modelos de simulación

4.4. Entorno de desarrollo Integrado.

Como parte del lenguaje específico de dominio, se procedió a construir un entorno de desarrollo Integrado (IDE) con el fin de facilitar las tareas de programación. El editor construido tiene soporte para los siguientes elementos.

- **Coloreado de sintaxis:** los token definidos en el analizador léxico cuando aparecen en el programa fuente, presentan diferentes colores.
- **Numeración de líneas:** el programador tiene a disposición el número de línea a medida que el programa se especifica, dicha información es importante cuando se produce un error léxico, sintáctico o semántico.
- **Autocompletación - Ayuda:** si un programador no conoce exactamente la palabra reservada que debe utilizar, el editor le presenta una lista de opciones cuando es presionado la combinación de teclas CRT + SPACE.
- **Plantillas:** con el fin de agilizar el proceso de programación se definieron plantillas de código comúnmente utilizadas: Entre estas se pueden encontrar: Plantilla para la definición del agente (ag + CTR + SHIFT - SPACE), reglas (ifr + CTR + SHIFT - SPACE), ambiente (env + CTR + SHIFT - SPACE) y main (mai + CTR + SHIT - SPACE).

Lo anterior fue implementado usando las siguientes herramientas.

- RsintaxTextArea: Adiciona la funcionalidad de resaltado de sintaxis, numeración de línea y autocompletación a un componente jTextArea del lenguaje JAVA.
- AutoComplete: Permite definir las palabras y las plantillas que pueden ser completadas por el editor, usando la combinación de teclas definidas.

- TokenMakerMaker: Permite personalizar el analizador léxico para el RsintaxTextArea.

En la Fig 4.3, se presenta una imagen del editor construido para el lenguaje UNALCOL. Por su parte, en la Fig 4.4, se presenta el mecanismo de marcado de errores.

Entre las características adicionales del editor a mencionar se encuentran.

- Tiene las características de un editor de texto tradicional: Abrir archivos con extensión .alife, guardar, guardar como, compilar, generar código, seleccionar, copiar y cortar.
- Tiene soporte para realizar búsquedas sobre el código fuente a través de: palabras clave y expresiones regulares.

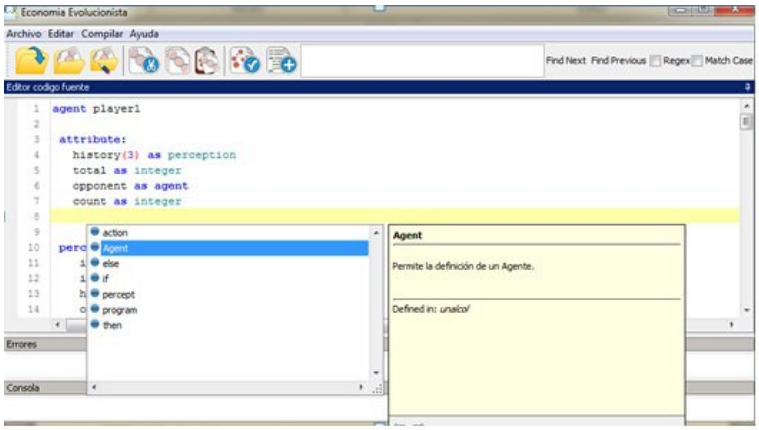


FIGURA 4.3. Editor de código fuente UNALCOL

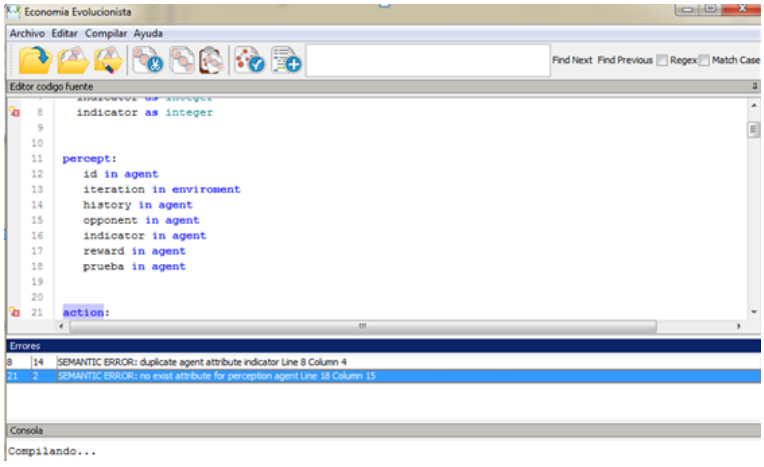


FIGURA 4.4. Presentación de errores en UNALCOL

4.5. Plataforma de Simulación.

Con el fin de permitir procesos de simulación se tuvo en consideración los planteamientos especificados por Russell et al. En la Fig 4.5, se presenta las interfaces y clases

abstractas principales que permiten la especificación de un modelo de simulación.

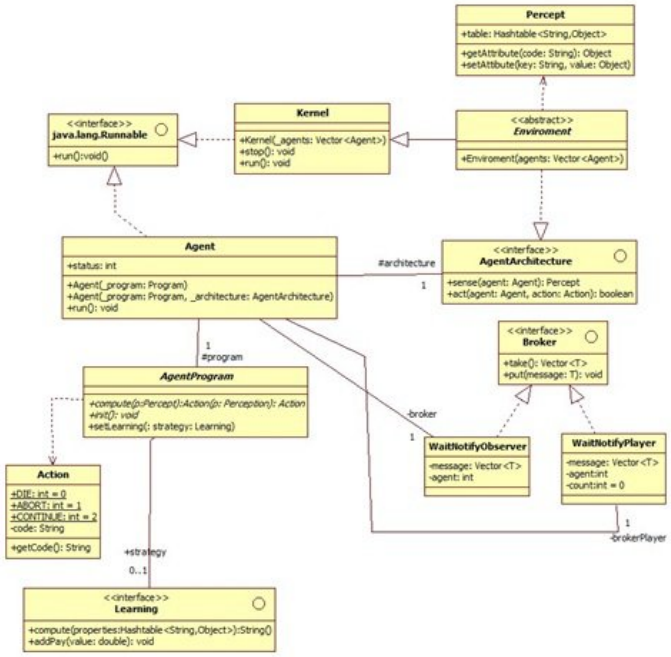


FIGURA 4.5. Clases principales simulador UNALCOL

Agent: clase que implementa *Runnable* que permite la especificación de agentes, y el control del ciclo de vida del agente. Tiene la función de observar el ambiente, obtener percepciones e invocar al programa para la obtención de las respectivas acciones a ejecutar.

AgentProgram: interfaz que define el esquema de especificación de programas de los agente. Dicho programa queda completamente especificado con la definición del método *compute*, el cual recibe las percepciones y retorna la acción a ejecutar.

AgentArchitecture: interface que permite especificar los mecanismos para los sensores y actuadores del agente.

Broker: interface que define los métodos necesarios para gestionar la concurrencia. Específicamente, contiene los métodos *take* y *put*. Con el método *take* los agentes pueden tomar los pagos realizados por el ambiente cuando el modelo de simulación es síncrono, por su parte con el método *put*, los agentes ponen las decisiones a disposición del ambiente.

Learning: clase Abstracta que define los elementos para la integración de estrategias de aprendizaje basado en librerías externas. Dicha interfaz define que se debe sobreescribir los métodos *compute* y *addPay*. El método *compute* recibe las percepciones y debe retornar la acción. Por su parte, el método *addPay* es utilizado cuando se debe dar una calificación a la acción ejecutada.

Enviroment: clase abstracta que define las operaciones que debe contener el ambiente. A través del ambiente, se pueden procesar las acciones de cada uno de los agentes y a partir de estas, asignar recompensas a cada uno de ellos.
Basado en las clases base anteriormente explicadas, se puede proceder a implementar un modelo de simulación.

A continuación se especifican las reglas de transformación aplicadas con el fin de generar la simulación basado en el lenguaje específico de dominio.

- Para cada uno de los agentes especificados en el lenguaje se genera una clase que extiende de la clase *Agent*, cuyo nombre corresponde con el identificador del agente. Cada clase tiene definida una variable de instancia, que almacenará todos los atributos que definen al agente.
- Por cada agente especificado, se genera una clase de tipo *Percept*. Dicha clase permite indicar los atributos del ambiente o propios del agente, que son usados por el programa para especificar la acción a ejecutar.
- Basado en el programa definido en el agente, se crea una clase que extiende de *AgentProgram*. Dicha clase contiene la traducción de las reglas *IF-THEN-ELSE* en términos del lenguaje de programación de propósito general JAVA.
- A partir del ambiente, se define una clase que extiende de *Enviroment*. La clase *Enviroment* contiene los siguientes elementos: Los atributos del ambiente, las reglas de decisión especificadas en el programa del ambiente, las reglas de decisión de tipo *@action* para cada agente en el método *act* y las reglas de decisión de tipo *@percept* para cada agente en el método *sense*.
- Clase ejecutable *Main* que permite inicializar los atributos del agente basado en los elementos indicados en el lenguaje específico de dominio.

El resultado de lo anterior, es un programa ejecutable que permite simular el modelo desarrollado por el programador en el lenguaje específico de dominio.

Con el fin de validar el funcionamiento del lenguaje *UNALCOL*, se plantea el desarrollo de los siguientes modelos:

- Juegos repetitivos no cooperativos, modelos que entran dentro de la línea de investigación de normas de comportamiento, especificado anteriormente.
- Juegos repetitivos no cooperativos usando esquemas de Modelamiento del Oponente.

4.6. Modelos de Simulación.

4.6.1. Juegos no Cooperativos repetitivos.

En el capítulo 2 del documento se presentó una descripción detallada de la teoría de Juegos no cooperativos, en los cuales se mostró la especificación de los juegos: dilema del prisionero, juego de la gallina y caza del ciervo.
Teniendo en cuenta lo anterior, se plantearon los siguientes escenarios: juegos entre estrategias usadas en competencia CEC04 y juegos integrando estrategias de aprendizaje basado en modelamiento del oponente.
Adicionalmente, se usará la métrica basada en recompensa como mecanismos de evaluación [55], por lo tanto, se buscará obtener a través del proceso de evolución, maximizar el pago promedio en los diferentes tipos de juegos.

- **Dilema del Prisionero**
  - **Especificación de competencias entre diversas estrategias del CEC04.**

En la Fig. 4.6 se presenta la especificación de los agentes *Player1*, *Player2* en el cual se definen los componentes de los agentes: atributos, percepciones, acciones y programa.
El programa del agente para el *Player1* tiene modelada la estrategia de juego *ALLC*. Por su parte, el *Player2* tiene como programa la estrategia *TFT*, es decir, que el agente toma como decisión la acción que fue ejecutado por el contrincante en la iteración anterior: *(history(iteration-1).opponent.action)*.
Dado que *history* es una variable de tipo *perception*, esta almacena la información de las decisiones y recompensas de cada uno de los agentes en el modelo de simulación de acuerdo al tamaño especificado. Para el caso particular del ejemplo, se están realizando los siguientes filtros: seleccionar la percepción de la iteración anterior *(history (iteration-1)*. Específicamente, la acción ejecutada por el oponente *(history(iteration-1).opponent.action)*.

```
agent player1
 attribute:
  history (1) as perception
  total as integer
  opponent as agent

 percept:
   id in agent
   iteration in enviroment
   history in agent
   opponent in agent
   reward in agent

 action:
  CO, DEF, EXIT

  program COOPERATE:
  @Program
  if iteration = 0 then
     action=CO
  else
    action=CO
  end if

  @Action
   if action=CO then
     color=yellow
   else
     color=blue
   end if

  @Percept
   if reward !=null then
      total =  total + reward
   end if
end agent
```

```
agent player2
 attribute:
  history(3) as perception
  total as integer
  opponent as agent

 percept:
   id in agent
   iteration in enviroment
   history in agent
   opponent in agent
   reward in agent

 action:
  CO, DEF, EXIT

  program TFT:
  @Program
  if iteration =0 then
     action=CO
  else
    if history(iteration-1).opponent.action=CO  then
           action=CO
   else
           action=DEF
   end if
  end if

  @Action
   if action=CO then
       cx=50
       cy=10
       color=yellow
   else
     cx=10
     cy=10
     color=blue
   end if

  @Percept
   if reward !=null then
      total =  total + reward
   end if
end agent
```

FIGURA 4.6. Programa modelo de simulación

Para el caso del ambiente, se especifica la matriz de pagos del dilema del prisionero usando reglas de decisión. Por ejemplo: si el jugador 0 (*Player1*) seleccionó en la iteración anterior la acción cooperar *(history(iteration).player(0).action=CO)* y el jugador 1 (*Player2*) seleccionó cooperación. Se debe dar como pago de acuerdo a la matriz del dilema

del prisionero, tres unidades a cada uno de los jugadores *(player(0).reward =3, player(1).reward=3)* y así sucesivamente. Lo anterior es presentado en la Fig. 4.7.

```
enviroment
 attribute:
   player() as agent


 program Enviroments:
  @Program
  if history(iteration).player(0).action=CO AND  history(iteration).player(1).action=CO  then
     player(0).reward =3
     player(1).reward=3

  else
     if history(iteration).player(0).action=CO AND  history(iteration).player(1).action=DEF  then
          player(0).reward=0
          player(1).reward=5
     else
          if history(iteration).player(0).action=DEF AND  history(iteration).player(1).action=CO then
             player(0).reward=5
             player(1).reward=0
          else
            if history(iteration).player(0).action=DEF AND  history(iteration).player(1).action=DEF then
                player(0).reward=1
                player(1).reward=1
            end if
          end if
     end if
  end if

end enviroment
```

FIGURA 4.7. Programa para la especificación del ambiente

Por último se especifican los parámetros de configuración como se presenta en la Fig. 4.8. Se declaran dos agentes, uno por cada tipo de agente declarado, se realiza la inicialización de los atributos para cada uno de ellos, se define el nombre del modelo como dilema con un total de 20 iteraciones y se especifica que se debe realizar un modelo sincronizado.

```
main
/*
 Declaracion de los agentes
*/
 dim fabian as player1
 dim isabel as player2

 fabian.total=0
 fabian.opponent=isabel
 fabian.id=fabian
 fabian.cx=10
 fabian.cy=20
 fabian.color=blue

 isabel.total=0
 isabel.opponent=fabian
 isabel.id=isabel
 isabel.cx=30
 isabel.cy=20
 isabel.color=blue

 enviroment.player=[fabian,isabel]

 configuration.name=dilema
 configuration.max=20
 configuration.sincronized=true

end main
```

FIGURA 4.8. Parámetros configuración de la simulación

Como resultado de la ejecución se obtiene la cooperación mutua, representado en un puntaje de 60 para cada uno de los jugadores, como se presenta en la Fig. 4.9 y 4.10.

FIGURA 4.9. ALLC Vr. TFT

FIGURA 4.10. Aplicación para la competencia entre ALLC Vr. TFT

A continuación se presentan los gráficos de resultados a partir de competencias entre estrategias del CEC04 modeladas como programas de los agentes. La competencia entre *RAND* y *TFT* da como resultado una recompensa de 56 puntos para el agente *Fabián (RAND)* y 56 puntos para el agente *Isabel (TFT)*. Como se puede observar en la Fig. 4.11 en las dos primeras iteraciones ambos agentes deciden cooperar obtenido un puntaje de 3. En la iteración 3, *TFT* coopera y *RAND* selecciona No cooperar. Por lo tanto, en la iteración 4 se nota como *TFT* cambia su estrategia de juego.

En la Fig. 4.12 se ven los resultados del enfrentamiento entre el agente *Fabián* con estrategia *ALLD* e *Isabe*l con estrategia *TFT* en cada una de las iteraciones. Los resultados dan 24 y 19 puntos respectivamente.
Como se observa el agente *Fabián* en la primera iteración obtiene un puntaje de 5 dado que *TFT* cooperó, sin embargo, en las siguientes iteraciones *TFT* juega No cooperar basado en la decisión de las iteraciones anterior de *ALLD*.

En la Fig. 4.13 se presenta los resultados del enfrentamiento entre el agente *Fabián* con estrategia *GRIM* e *Isabel* con estrategia *STFT*. Los puntajes

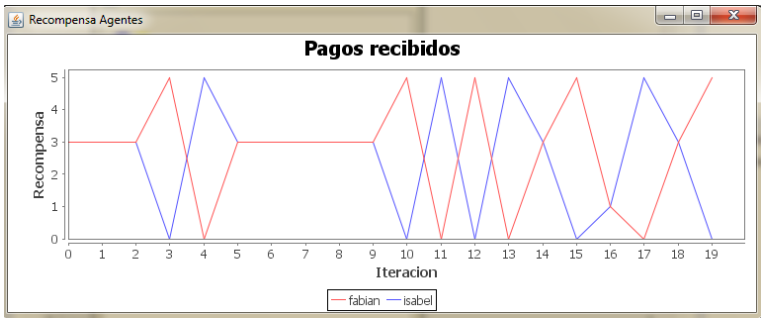


FIGURA 4.11. RAND Vr. TFT

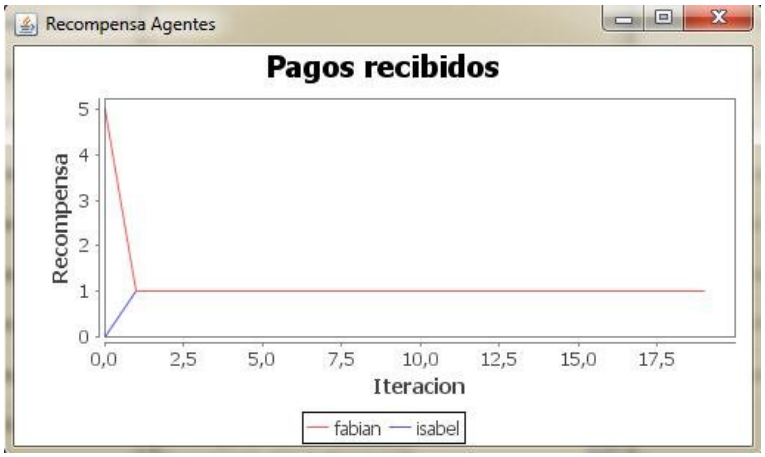


FIGURA 4.12. ALLD Vr. TFT

recibidos luego de 20 iteraciones son 23 puntos para cada uno.
Como se observa, el agente *Fabián* coopera mientras el agente *Isabel* No coopera en la iteración 0, dado lo anterior, en la próxima iteración *Fabián* No coopera e *Isabel* coopera, luego *Fabián* recibe 5 puntos e Isabel 0. Posteriormente, *Fabián* siempre jugará No Cooperar y *STFT* se adaptará jugando siempre la No cooperación.

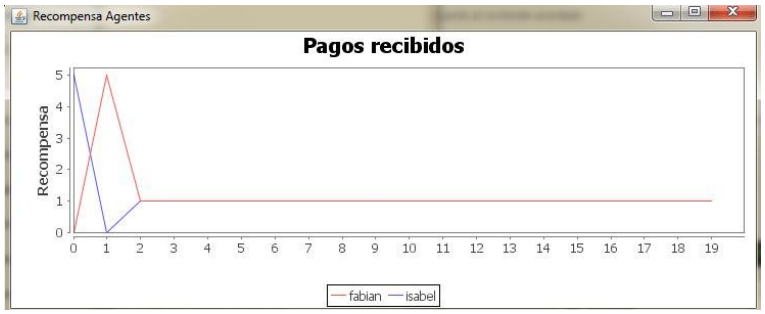


FIGURA 4.13. GRIM Vr. STFT

En la Fig. 4.14, se presentan los resultados del enfrentamiento del agente *Fabián* con estrategia *ALLD* e *Isabel* con estrategia *TFTT*. El puntaje obtenido por *Fabián* e *Isabel* son 28 y 18 respectivamente.
Como se presenta en el gráfico, las dos primeras iteraciones *Fabián* recibe un

puntaje de 5 dado la cooperación de *Isabel*, sin embargo, *Isabel* posteriormente da dos No cooperaciones consecutivas se adapta a No cooperación.

FIGURA 4.14. ALLD Vr. TFTT

En la Fig. 4.15, se enfrenta Fabián con estrategia *RAND* e Isabel con estrategia *TFTT*. Como resultado se obtiene como puntaje 65 y 35 respectivamente.
Como se evidencia el agente *Fabián* cambia aleatoriamente la estrategia e *Isabel* siempre coopera a no ser que se presente dos No cooperaciones consecutivas por parte del agente *Fabián*.

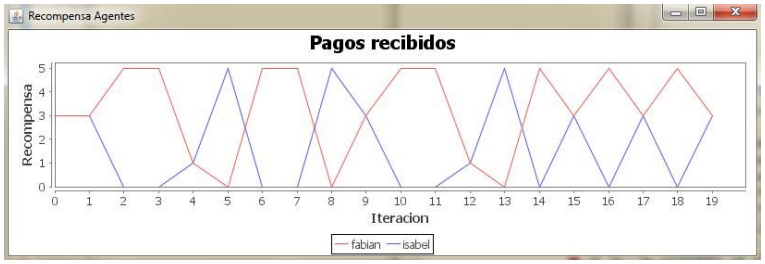


FIGURA 4.15. RAND Vr. TFTT

- **Juego de la Gallina**

  En el caso del juego de la gallina, se realizaron enfrentamientos similares a los del dilema del prisionero. A continuación se presentan los gráficos de comportamientos y los puntajes obtenidos para cada uno de estos.

  En la Fig 4.16 se observa la gráfica de comportamiento cuando se enfrenta el Agente *Fabian* con estrategia de juego *ALLD* y el Agente *Isabel* con estrategia *TFT*.
  Los puntajes obtenidos para el Agente *Fabian* e *Isabe*l son 5 y 0 respectivamente. Como se puede observar, *ALLD* recibe un puntaje de 5 en la primera iteración dado que *TF T* Coopero, sin embargo, posteriormente *TFT* se adapta y se obtiene mutua No cooperación.

  Para el caso del enfrentamiento entre *ALLD* (Agente *Fabian*) y *TFTT* (Agente *Isabel*) se obtiene un puntaje de 10 y 0 respectivamente. La Fig 4.17 presenta las jugadas realizadas a lo largo de las 20 iteraciones. En este caso, el comportamiento es similar al caso anterior, sin embargo, *TFTT* luego de dos No cooperaciones, decide no Cooperar.

FIGURA 4.16. ALLD Vrs TFT

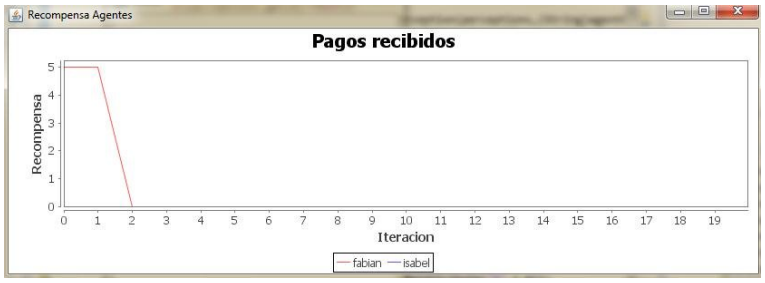


FIGURA 4.17. ALLD Vrs TFTT

Cuando compiten las estrategias *GRIM* (Agente *Fabian*) y *STFT* (Agente *Isabel*) los resultados obtenidos son 5 puntos para cada jugador. El gráfico de comportamiento es presentado en la Fig 4.18.
En este caso, *STFT* empieza no cooperando y *GRIM* cooperando. Posteriormente, *GRIM* No coopera dada la traición y *STFT* juga a Cooperar.

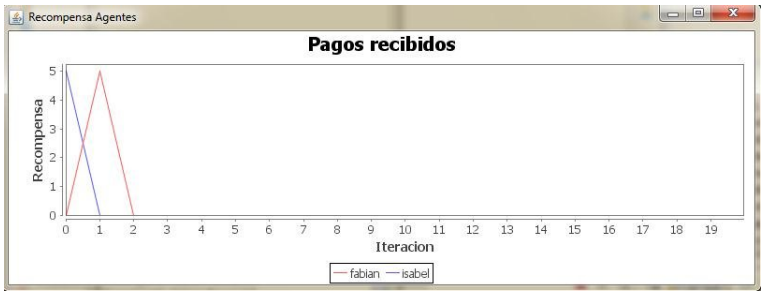


FIGURA 4.18. GRIM Vrs STFT

Los resultados de otros enfrentamientos es presentado en la tabla 4.1.

- **Caza del Ciervo**

  En el proceso de enfrentamiento entre las diferentes estrategias para el juego caza del ciervo, los resultados son similares, sin embargo, el puntaje es mayor dado que la no cooperación se da un puntaje de 1 punto en comparación a los 0 puntos del dilema del prisionero y el juego de la gallina.
  Los resultados de enfrentamientos son presentados en la tabla 4.2

TABLA 4.1. Resultados Juego de la Gallina

Estr. Jugador1	Estr. Jugador2
RAND (37 Puntos)	TFT (32 Puntos)
Perceptron Multicapa (0)	ALLD (5 Puntos)
Perceptron Multicapa (60)	ALLC (60)
Perceptron Multicapa (60)	TFT (60)
Perceptron Multicapa (24)	RAND (29)
Radial (0)	ALLD (5 Puntos)
Radial (60)	ALLC (60)
Radial (60)	TFT (60)
Radial (35)	RAND (35)

4.6.2. Modelamiento del Oponente.

En esta sección, se plantea el desarrollo de evolución de estrategias de decisión usando redes neuronales, especificamente: perceptrón multicapa y redes de base radial.

Tal como se indicó anteriormente, el proceso de evolución de estrategias para el caso del presente proyecto de investigación consiste en entrenar a un agente para enfrentarse contra otros agentes en Juegos repetitivos no cooperativos. Se presentarán dos esquemas, un modelo ímplicito y un modelo explícito.

Para el modelo ímplicito, se realiza un proceso de entrenamiento fuera de línea (offline) con el fin tener redes neuronales obtenidas por procesos de neuroevolución, que tengan la capacidad de defenderse cuando se enfrenta con oponentes específicos.

Para el caso del modelo explícito, se adapta en línea el comportamiento del agente basado en el contrincante con el cual se enfrenta, para tal fin, el agente basado en la historia de juego, busca obtener un contrincante prototipo usando esquemas de neuroevolución con el fin de adaptar su estrategia de juego dinamicamente.

- **Modelo del Oponente ímplicito.**

  Con el fin de realizar el proceso de experimentación para el modelo implícito, se tomó como base el proceso de evolución de estrategias presentado en el capítulo II.

  En este caso, se obtuvieron redes neuronales (perceptrón multicapa, red de base radial), que fueron entrenadas compitiendo en modo torneo, con un conjunto de estrategias usadas en la competencia CEC04. Para realizar el proceso de integración de las redes neuronales a los agentes, se procede de forma equivalente a lo especificado en el ítem anterior, es decir:

  1. Usando el lenguaje UNALCOL se realiza la especificación del juego No cooperativo, como se presenta en las Fig 4.19 y 4.20.

TABLA 4.2. Resultados Caza del Ciervo

Estr. Jugador1	Estr. Jugador2
ALLD (22 Puntos)	TFT (19 Puntos)
RAND (52)	TFTT (34 Puntos)
GRIM (21)	STFT (21)
TFT (48)	RAND (48)
Perceptron Multicapa (22)	ALLD (19)
Perceptron (100)	ALLC (100 Puntos)
Perceptron (36)	RADIAL (36)
Radial (33)	RAND (33)
Radial (100)	TFT (100)

2. Basado en el programa, la plataforma desarrollada genera automáticamente una aplicación que cumple con los elementos configurados por el programador. Una parte del código es presentado en la Fig 4.21.

Como se puede observar, para el proceso de toma decisiones, se realiza la carga dinámica de la clase en el cual está modelada la red neuronal *(unalcol.agents.nn.WeighDilemmaNn)*, luego de los tres primeros movimientos, la red neuronal es la que realiza el proceso de inferencia, usando como fuente de información la percepción del agente.

La clase *unalcol.agents.nn.WeighDilemmaNn*, debe ser especificada por el usuario para poder ser integrada a la plataforma de simulación. En este caso, se integró cada una de las redes neuronales obtenidas en el proceso de evolución del capítulo II.

En la Fig 4.22, se presenta la configuración para el caso de perceptrón multicapa.

- Los pesos obtenidos por el proceso de neuroevolución.
- Seis neuronas en la capa de entrada que corresponde con 3 movimientos previos para cada jugador.
- Una capa Oculta con un total de 20 neuronas.
- Una neurona en la capa de salida.

De igual forma se procedió con la red de base radial la cual cuenta con la siguiente configuración.

- Los pesos obtenidos por el proceso de neuroevolución.
- Seis neuronas en la capa de entrada que corresponde con 3 movimientos previos para cada jugador.
- Una capa Oculta con un total de 20 neuronas.
- Una neurona en la capa de salida.
- Kernel Gaussianos en las neuronas de la capa Oculta.

```
agent player1

import:
  decision as unalcol.agents.nn.WeightDilemmaNn

 attribute:
   history(3) as perception
   total as integer
   opponent as agent


 percept:
    id in agent
    iteration in enviroment
    history in agent
    opponent in agent
    reward in agent


 action:
   CO, DEF, EXIT
```

FIGURA 4.19. Integración red neuronal modelo ímplicito

```
program NN:
  @Program
   if iteration = 0 then
        action=CO
   else
      if iteration=1 then
        action=CO
      else
          if iteration =2 or iteration = 3 then
                  if history(iteration-1).opponent.action=CO AND  history(iteration-2).opponent.action=CO then
                      action=CO
                  else
                      if history(iteration-1).opponent.action=CO AND  history(iteration-2).opponent.action=DEF then
                          action=DEF
                      else
                         if history(iteration-1).opponent.action=DEF AND  history(iteration-2).opponent.action=CO then
                           action=CO
                         else
                           if history(iteration-1).opponent.action=CO AND  history(iteration-2).opponent.action=CO then
                             action=CO
                           else
                                action=CO
                           end if
                         end if
                      end if
                  end if
          else
                action=decision
          end if
```

FIGURA 4.20. Integración red neuronal modelo ímplicito (2)

A continuación se presentan los resultados obtenidos.

1. **Dilema del Prisionero**.
   Para la competencia entre el perceptrón multicapa y las estrategias de juego *GRIM*, *TFT*, *ALLC* se obtiene la cooperación mutua, tal como se observa en la Fig 4.23. Como se puede ver ambos jugadores reciben 3 puntos en cada una de las iteraciones, logrando al final de las 20 rondas un puntaje de 60 puntos.

   Uno de los elementos que se deben tener en cuenta, es que al realizar el proceso de entrenamiento usando el torneo, el proceso de neuroevolución tal como se concluyó en el capítulo II, converge a la cooperación mutua. Si se realizará el proceso de entrenamiento compitiendo contra estrategias individuales, el algoritmo se comporta de forma egoísta con el contrincante, logrando de esta

```
package unalcol.agents.example.dilema;

import java.util.Hashtable;
public class NN implements AgentProgram{
    private Language language;
    private Learning decision =null;


    public NN  (Language language){
        this.language=language;
        if(this.decision ==null){
            this.decision = (Learning) DiscoverSingleton.find( Learning.class, "unalcol.agents.nn.WeightDilemmaNn");
        }
    }
    public Action compute(Percept p) {

        String  id = (String) p.getAttribute("ID");
        Integer   iteration = (Integer) p.getAttribute("ITERATION");
        java.util.Vector history= (java.util.Vector) p.getAttribute("HISTORY");
        String  opponent = (String) p.getAttribute("OPPONENT");
        Integer reward =(Integer) p.getAttribute("REWARD");

        Hashtable<String, Object> properties = new Hashtable<String, Object> ();
        properties.put("ID", id);
        properties.put("ITERATION", iteration);
        properties.put("HISTORY", history);
        properties.put("OPPONENT", opponent);
        properties.put("REWARD", reward);


        if (iteration== 0){
            return ( new Action("CO"));
        }
        else {
            if (iteration== 1){
```

FIGURA 4.21. Código integración con red neuronal modelo implícito

```
import java.util.Hashtable;

public class WeightDilemmaNn extends Learning{

    private MultiLayerPerceptron perceptron=null;
    public WeightDilemmaNn(){
        double weight[]={-0.3713697184358862,-0.36357978720493056,0.485534504949962,-0.
        perceptron=new MultiLayerPerceptron(weight,6,20,1,1,2);
    }


    @Override
    public Object compute(Hashtable<String, Object> properties) {

        String  id = (String) properties.get("ID");
        Integer iteration =(Integer) properties.get("ITERATION");
        java.util.Vector history= (java.util.Vector) properties.get("HISTORY");
        String  opponent = (String) properties.get("OPPONENT");
        double XOR_TEST[]=new double[6];

        System.out.println("iteration" + iteration);
```

FIGURA 4.22. Perceptron modelo del Oponente ímplicito

forma un mayor puntaje.

En el caso del perceptrón (*Agente fabian*) compitiendo con la estrategia *RAND*, se ve como la red neuronal buscar adaptarse, evidenciándose en el puntaje final obtenido en dos competencias realizadas. Para el caso de la Fig. 4.24, el Perceptron (*Agente fabian*) obtiene 67 puntos contra 17 de (*RAND*, y para la Fig 4.25, se obtienen 54 vrs. 34 respectivamente.

Es importante notar también, como la misma red neuronal, cuando compite con *ALLD*, pierde en puntos en las 3 primeras iteraciones, pero luego se adapta respondiendo No cooperar, dado que maximiza su puntaje. Este comportamiento puede ser visualizado en la Fig 4.26. El puntaje total obtenido corresponde a 16 puntos por parte del perceptrón contra 36 de la estrategia ALLD *(Agente Isabel)*

Para caso de la competencia entre el perceptrón y *STFT* se puede observar cómo se obtiene un puntaje de 28 puntos (Fig. 4.27). En primera instancia el Agente Isabel *(STFT)* gana cinco puntos dada la cooperación del *Agente Fabian*, sin embargo, posteriormente, el perceptron adapta su estrategia de

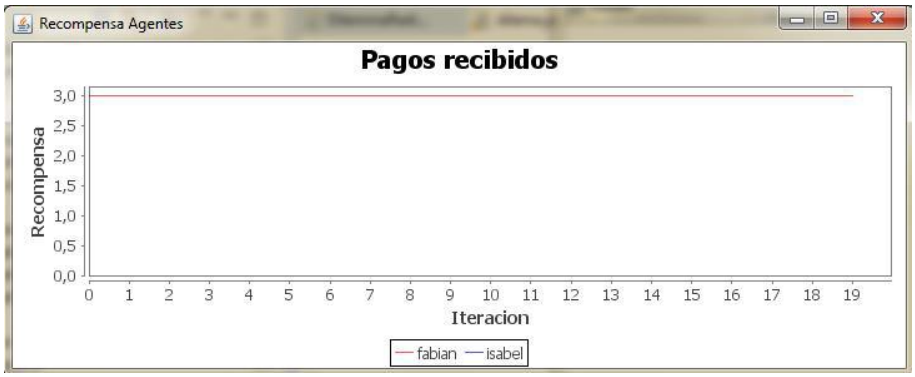


FIGURA 4.23. Perceptrón vrs GRIM

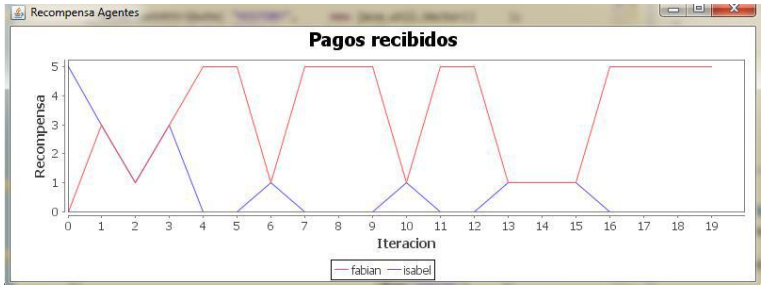


FIGURA 4.24. Perceptrón vrs RAND (1)

juego en función de las jugadas del contrincante.

De igual forma se procedió con la red de base radial, los resultados son similares a los obtenidos por el perceptrón multicapa. Cuando compite la red de base Radial contra la estrategia *ALLD*, tal como lo presenta la Fig 4.28, se obtiene 36 puntos por parte del *Agente Isabel (ALLD)* contra 16 del *agente Fabian*. Como se observa la red neuronal luego de la tercera iteración adapta su comportamiento con el fin de maximizar su puntaje.

El enfrentamiento entre la red de base radial *(Agente Fabian)* y *RAND* da como resultado 66 puntos para el Agente Fabian contra 26 de *RAND* (Fig. 4.29). La red se comporta de tal manera que garantice un buen puntaje. Es importante también mencionar, como la red neuronal evolucionada es poco sensible al ruido, producto de jugadores sin un patrón de comportamiento

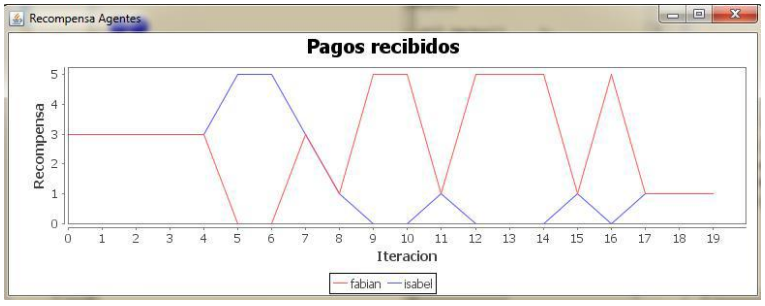


FIGURA 4.25. Perceptrón vrs RAND (2)

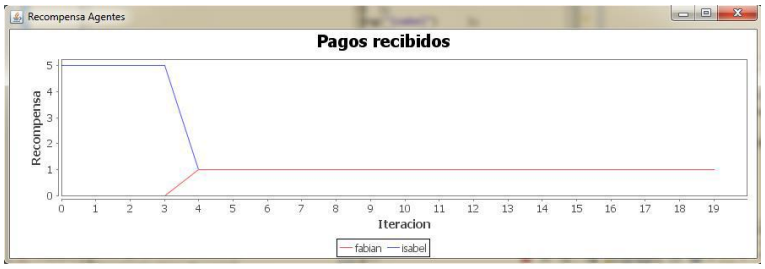


FIGURA 4.26. Perceptrón vrs ALLD

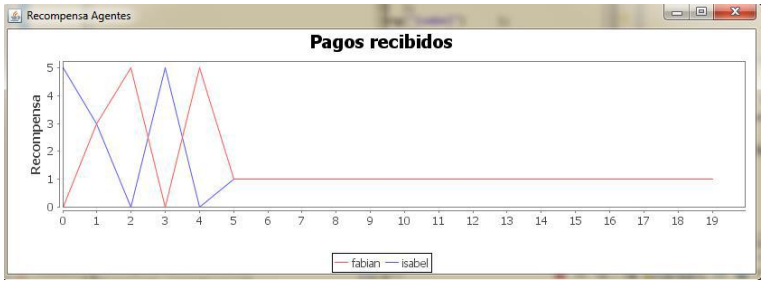


FIGURA 4.27. Perceptrón vrs STFT

específico.

Para el caso de la competencia entre la Red de base radial contra *ALLC* ó *GRIM*, se obtiene como resultado la cooperación mutua, es decir, un puntaje de 60 puntos producto de las 20 rondas de juego. Por último, el enfrentamiento entre la red de base Radial *(Agente Fabian)* y la estrategia de juego *STFT*, se presenta un fenómeno de adaptación continua, el comportamiento puede ser observado en la Fig 4.30. El resultado final da como resultado un puntaje de 51 para el *Agente Fabian* y 56 para el *Agente Isabel*.

Del proceso de experimentación, se puede indicar, que el perceptron multicapa y la red de base radial obtenidos por procesos de neuroevolución, tienen la capacidad de adaptarse cuando compite con diversas estrategias de juegos. Adicionalmente, tienen poco sensibilidad al ruido producto de jugadores que no tengan un patrón de juego determinado; lo anterior se refleja en los puntajes obtenidos cuando se enfrenta contra estrategias de juego Aleatoria *(RAND)*.

FIGURA 4.28. Radial vrs ALLD

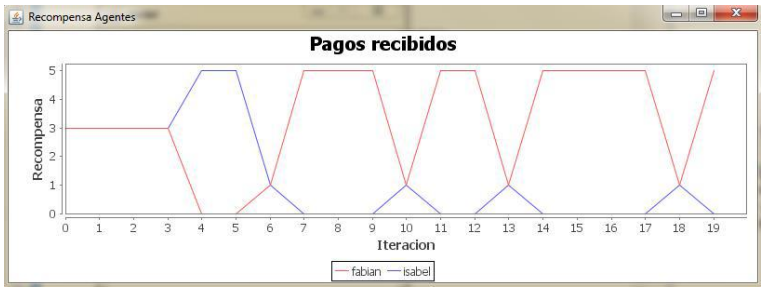


FIGURA 4.29. Radial vrs RAND

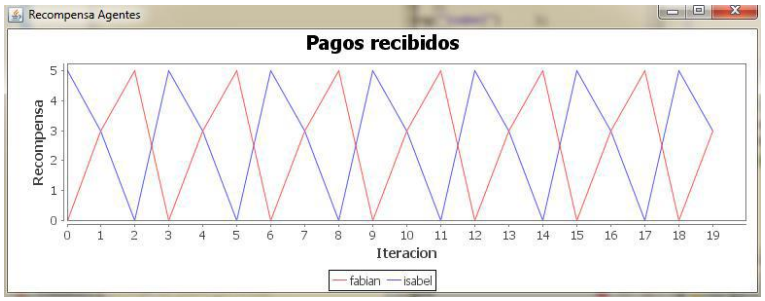


FIGURA 4.30. Radial vrs STFT

Dado que las redes Neuronales obtenidas por procesos de neuroevolución, puede ser aplicados para el caso del juego de la gallina y caza del ciervo iterado, tal como se concluyó en el capítulo II. A continuación, se procede a mostrar la tabla de resultados. Los resultados del enfrentamiento de las redes neuronales en el juego de la gallina son presentadas en la tabla 4.3, por su parte, los resultados de la caza del ciervo son presentados en la tabla 4.4.

- Modelo del Oponente Explícito
  Tal como se especificó en el estado del arte, el modelamiento explícito, entrena una forma funcional tomando información del oponente como entrada y produce una representación del oponente como salida, en términos de las acciones que el oponente puede tomar. Para el caso específico de los juegos repetitivos no cooperativos, se usaran las decisiones pasadas del oponente con el fin de predecir acciones futuras.

  El esquema general de funcionamiento del aprendizaje de estrategias será de la siguiente manera:

  1. Jugar Aleatoriamente las primeras N rondas de juego.
  2. Almacenar el historial de juegos tanto del oponente como propio.
  3. Buscar una red neuronal que tenga un comportamiento de juego similar al contrincante, cuando se enfrenta con el mismo patrón de juego inicial de la estrategia propia.
  4. Buscar una red neuronal como contra-estrategia, que tenga capacidad de maximizar puntaje en próximas M rondas.
  5. Repetir el proceso, cada cierto número de iteraciones.

TABLA 4.3. Resultados Juego de la Gallina

Estr. Jugador1	Estr. Jugador2
NN (48 Puntos)	RAND (13 Puntos)
NN (0 Puntos)	ALLD (20 Puntos)
NN (60)	ALLC (60)
NN (60)	TFT (60)
NN (13)	STFT (13)
NN (60)	TFTT (60 Puntos)
Radial (48)	RAND (18)
Radial (0)	ALLD (20)
Radial (60)	ALLC (60)
Radial (60)	TFT (60)
Radial (51)	STFT (56)

En el presente trabajo, se hará uso de dos modelos de redes neuronales: perceptron multicapa y redes de base radial. A continuación, se especifica el esquema de trabajo realizado.

Para el caso del perceptrón multicapa se procedió de la siguiente manera:

1. Generación de un perceptron multicapa totalmente conectado, cuya estructura es: 2 neuronas en la capa de entrada, una capa oculta con 3 neuronas y una neurona en la capa de salida. Los pesos son aleatoriamente generados entre [-1, 1] tal como se realizó en el capítulo II. Según proceso de experimentación realizado las N rondas de juego seleccionadas corresponde a *8*.
2. Se procede a almacenar el historial de juegos (8 Jugadas) tanto del contrincante como las propias.
3. Teniendo la historia de juego, se busca un perceptron multicapa que tenga un comportamiento de juego similar al contrincante, cuando se enfrenta con el mismo patrón de juego inicial de la estrategia propia. Con el fin de encontrar dicho imitador se hace uso de un algoritmo genético, especificamente, el algoritmo busca los pesos de la red que garanticen que se minimiza la diferencia entre los juegos del oponente y del imitador. El proceso de evolución de los pesos en este caso es equivalente al proceso explicado en el capítulo II.
4. Obtenido el imitador del contrincante, se busca un perceptron multicapa como contra-estrategia que tenga capacidad de maximizar puntaje en próximas M rondas. Para el caso del proyecto, luego de procesos de experimentación se determinó que M corresponde a *5* Rondas. Las búsqueda del contrincante se hace de forma similar al caso anterior, un algoritmo genético selecciona la red neuronal que maximiza la recompensa esperada en las *5* rondas.
   El proceso de evolución de los pesos para la red de contraestrategia, en este caso, es equivalente al proceso explicado en el capítulo II.

TABLA 4.4. Resultados Juego Caza del Ciervo

Estr. Jugador1	Estr. Jugador2
NN (29 Puntos)	RAND (26 Puntos)
NN (16 Puntos)	ALLD (28 Puntos)
NN (100)	ALLC (100)
NN (100)	TFT (100)
NN (100)	STFT (100)
NN (100)	TFTT (100 Puntos)
Radial (38)	RAND (29)
Radial (16)	ALLD (26)
Radial (100)	ALLC (100)
Radial (100)	TFTT (100)
Radial (53)	STFT (56)

Con el fin de realizar pruebas sobre el esquema planteado, en primera instancia se procedió a realizar juegos estándar, obteniendo como conclusión que todas las técnicas convergen a la maximización del pago cuando compiten con las estrategias del CEC04.
En las Figuras 4.31, 4.32 y 4.33 se presenta un resumen de los resultados obtenidos para el caso del dilema del prisionero, el juego de la gallina y la caza del ciervo, cuando se realiza el proceso de evolución de estrategias usando perceptron multicapa.
Los parámetros usandos para el proceso de experimentación está basado en los obtenidos en el capítulo II, solo varía el número de iteraciones y la población. Para el caso del perceptron multicapa los valores encontrados a través de experimentación con los cuales se obtiene mejor rendimiento son: En la identificación de Imitador (Población=100, Generaciones= 500) y para la identificación de la contraestrategia (Población=200, Generaciones= 200).

A continuación, se presenta el análisis de los resultados para cada uno de los juegos:

En el Dilema el Prisionero.

- Con todas las técnicas, si el contrincante juega con ALLD, GRIM, la mejor decisión para maximizar el pago es no-cooperar, esto lo refleja el pago promedio obtenido de 1.
- Cuando el contrincante juega con TFT, STFT emerge la cooperación mutua, dado que ambos jugadores maximizan la recompensa obtenida.
- Cuando el contrincante juega con ALLC, TFTT, la mejor decisión para maximizar el puntaje es no-cooperar.

En el Juego de la Gallina

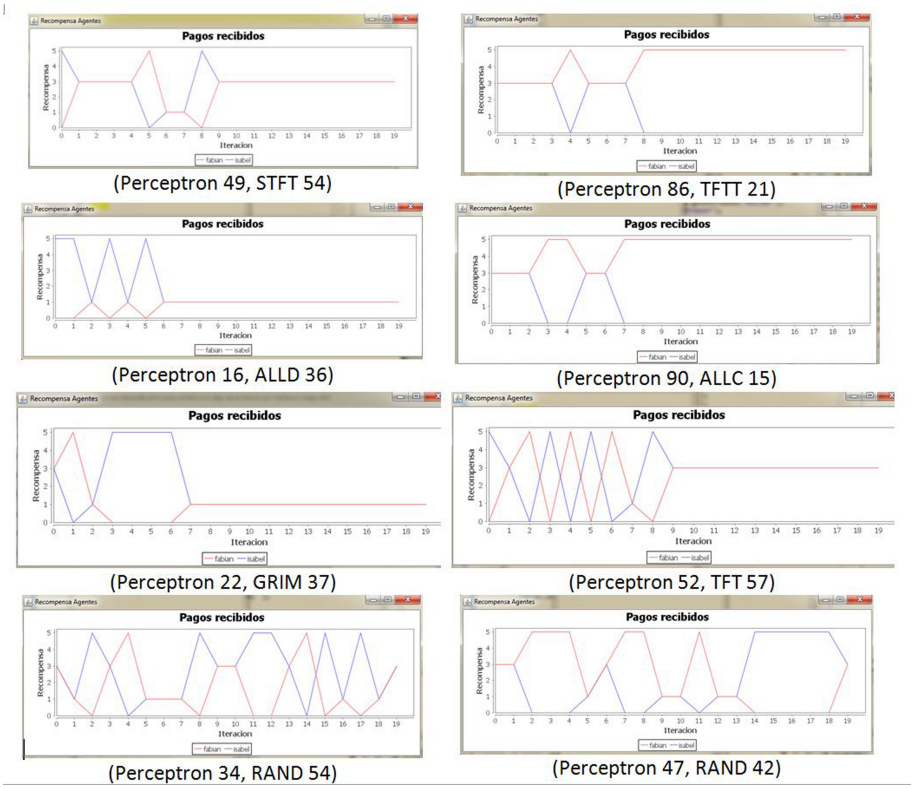


FIGURA 4.31. Resultados dilema del prisionero usando modelamiento perceptron multicapa

- Si el contrincante juega ALLC, NEG luego la mejor decisión en todos los casos es no cooperar (No desviarse); es decir el contrincante siempre va ser una gallina.
- Si el contrincante juega con ALLD, GRIM, la mejor decisión encontrada es no cooperar. La red neuronal en este caso dado que no tiene la capacidad jugar múltiples veces como se realiza en el proceso de evolución offline, no identifica que es mejor ser una gallina en todos los casos con el fin de obtener algún beneficio.
- Si el contrincante juega con TFT, STFT, emerge la cooperación mutua, dado que ambos jugadores maximizan la recompensa obtenida.

En la caza del Ciervo.

- Si el contrincante juega TFT, STFT luego la mejor decisión en todos los casos es cooperar; es decir buscar cazar el ciervo (Confianza mutua) dado que se maximiza la recompensa esperada.
- Si el contrincante juega con ALLD, la mejor decisión para maximizar el pago es no-cooperar, esto lo refleja el pago obtenido promedio de 1.
- Si el contrincante juega con RAND, la red neuronal adapta su estrategia de juego cambiando constantemente las decisiones en cada una de las rondas. Los resultados garantizan una maximización del pago.

Comparando con los resultados obtenidos usando el proceso de evolución offline se pueden notar los siguientes elementos.

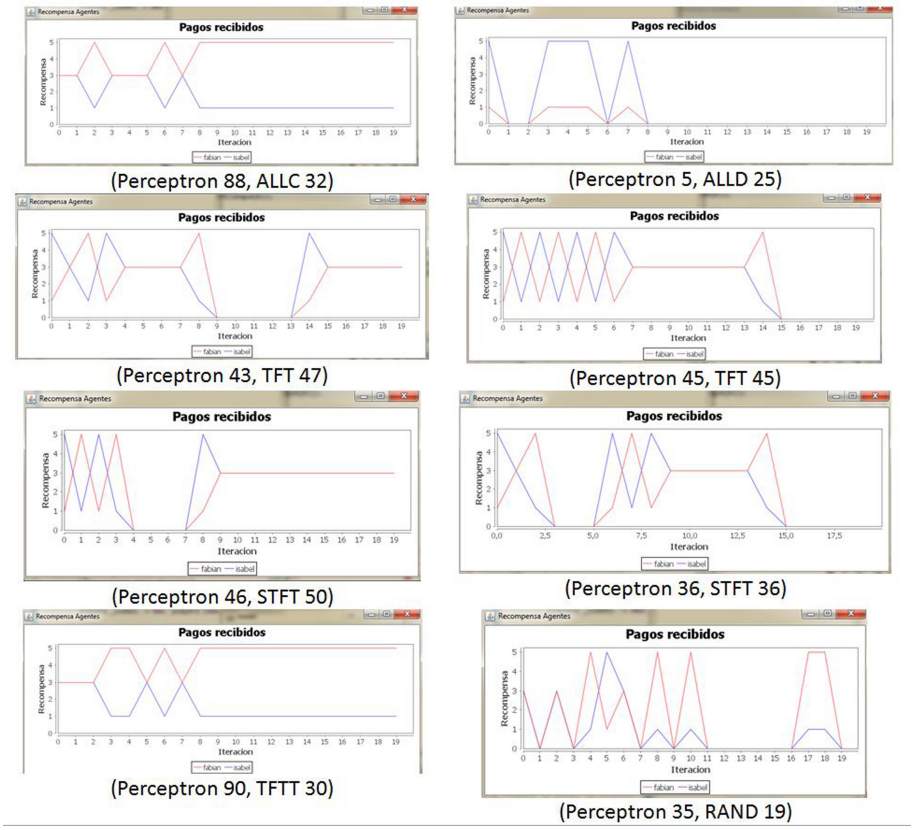


FIGURA 4.32. Resultados juego de la gallina usando modelamiento perceptron multicapa

1. El proceso de cooperación mutua emerge cuando se enfrenta con estrategias como TFT, STFT.
2. Cuando se enfrenta con estrategias de juego en los cuales puede identificar debilidades del oponente como ALLC, juega a la no cooperación.
3. Cuando se enfrenta con estrategia GRIM, en todos los casos, luego de la iteración 8, la decisión es la no cooperación. Lo anterior, dado que generalmente en la red neuronal obtenida aleatoriamente para los primeros juegos se presentan no cooperaciones.

Para el caso de las redes de base radial, el esquema utilizado es similar al caso del perceptron multicapa. El único item que cambia respecto al esquema es el primero, el cual consiste en generar una red de base radial para los primeros enfrentamientos. En este caso, se generó una red cuya estructura es: 2 neuronas en la capa de entrada, una capa oculta con 4 neuronas y una neurona en la capa de salida. Los pesos son aleatoriamente generados entre [-10, 10] tal como se realizó en el capítulo II. Según proceso de experimentación realizado las N rondas de juego seleccionadas corresponde a 8.

En las Figuras 4.34, 4.35 y 4.36 se presenta un resumen de los resultados obtenidos para el caso del dilema del prisionero, el juego de la gallina y la caza del ciervo, cuando se realiza el proceso de evolución de estrategias usando perceptron multicapa.
Los parámetros usandos para el proceso de experimentación está basado en los obtenidos en el capítulo II, solo varía el número de iteraciones y la población. Para el caso de las redes de base radial, los valores encontrados a través de experimentación

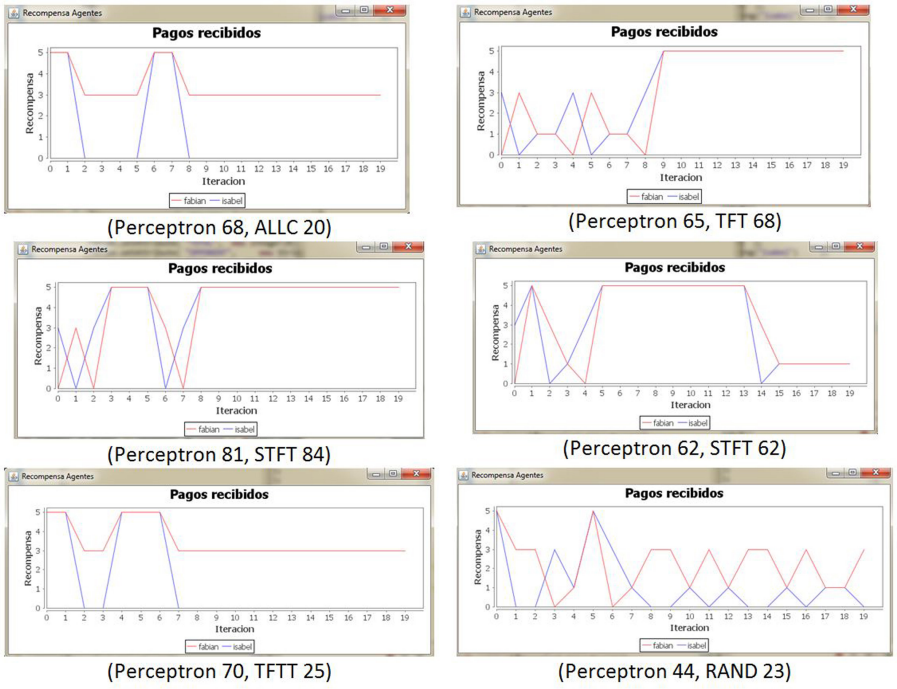


FIGURA 4.33. Resultados caza del ciervo usando modelamiento perceptron multicapa

con los cuales se obtiene mejor rendimiento son: En la identificación de Imitador (Población=100, Generaciones= 300) y para la identificación de la contraestrategia (Población=200, Generaciones= 300).

El análisis de los resultados para cada tipo de juego, es presentado a continuación:

En el Dilema el Prisionero.

- Si el contrincante juega con ALLD, GRIM, la mejor decisión para maximizar el pago es no-cooperar, esto lo refleja el pago promedio obtenido de 1. Sin embargo, es de anotar que la red neuronal obtenida por proceso de neuroevolución cuando se enfrenta con estrategia GRIM tiene a cooperar, luego en las pruebas realizados en algunas circunstancias se obtenida pérdida de puntos dada dicha tendencia.
- Cuando el contrincante juega con TFT, luego de la iteración 8 tiene a la cooperación mutua, sin embargo, en etapas posteriores busca la no cooperación tratando de identificar posibles debilidades.
- STFT no emerge la cooperación mutua como el caso anterior, en este caso la red neuronal realiza el proceso de adaptación a la no cooperación.
- Cuando el contrincante juega con ALLC, TFTT, la mejor decisión para maximizar el puntaje es no-cooperar.

En el Juego de la Gallina.

Los resultados son similares al caso anterior del dilema del prisionero, sin embargo se puede realizar las siguientes observaciones.

- Si el contrincante juega SFTT, no se obtiene la cooperación mutua como el caso del perceptron multicapa, y a diferencia del caso analizado anteriormente, la

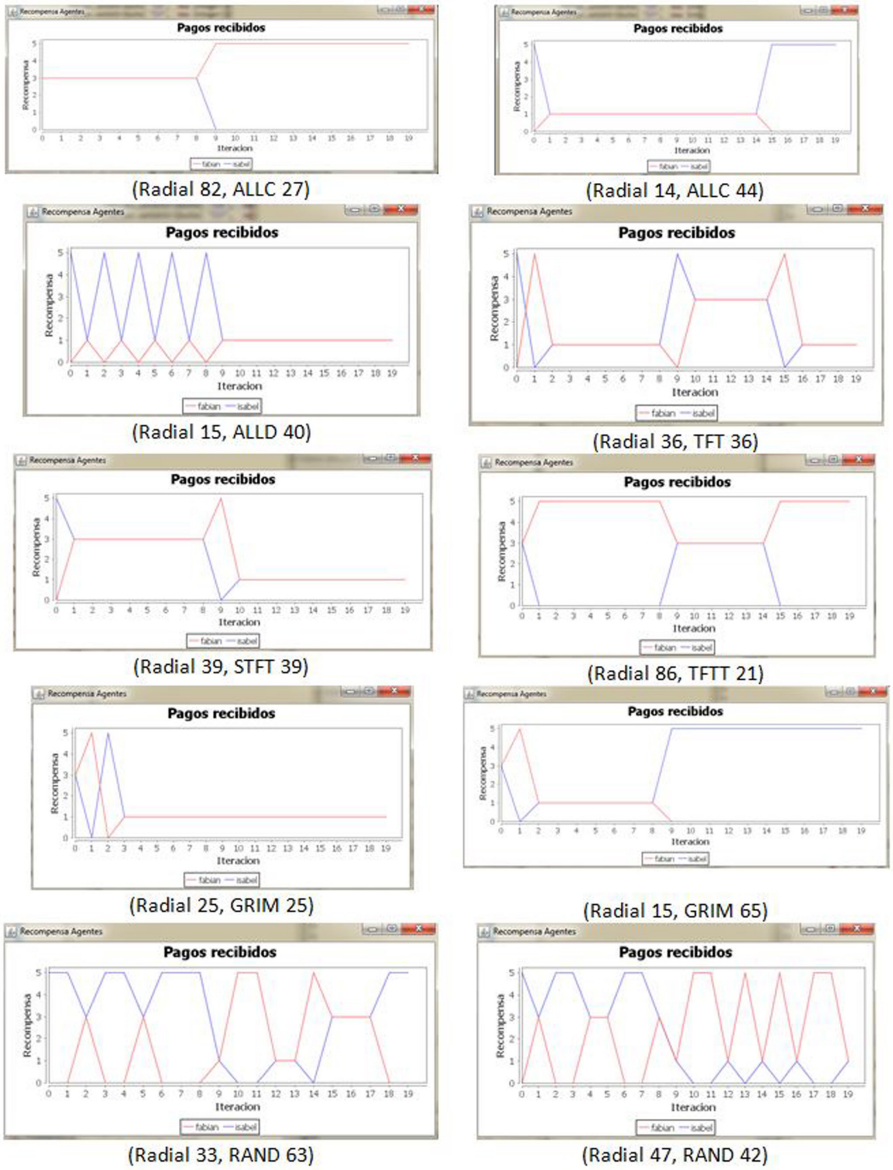


FIGURA 4.34. Resultados dilema del prisionero usando modelamiento red de base radial

red realiza fluctuaciones en las decisiones luego de la iteración 8 que es cuando se realiza el proceso de adaptación de la estrategia.

– Dada la tendencia a la cooperación de las red obtenida por procesos de neuro-evolución, cuando se enfrenta contra la estrategia de juego ALLD, se obtiene un puntaje extra en comparación a lo que sucede con el perceptron multicapa; lo anterior, dado que es mejor ser siempre ser una gallina dado que se identifica que el contrincante nunca va a cooperar.

En la caza del Ciervo.

– Si el contrincante juega TFT, STFT luego la mejor decisión en todos los casos es cooperar; es decir buscar cazar el ciervo (Confianza mutua) dado que se maximiza la recompensa esperada.

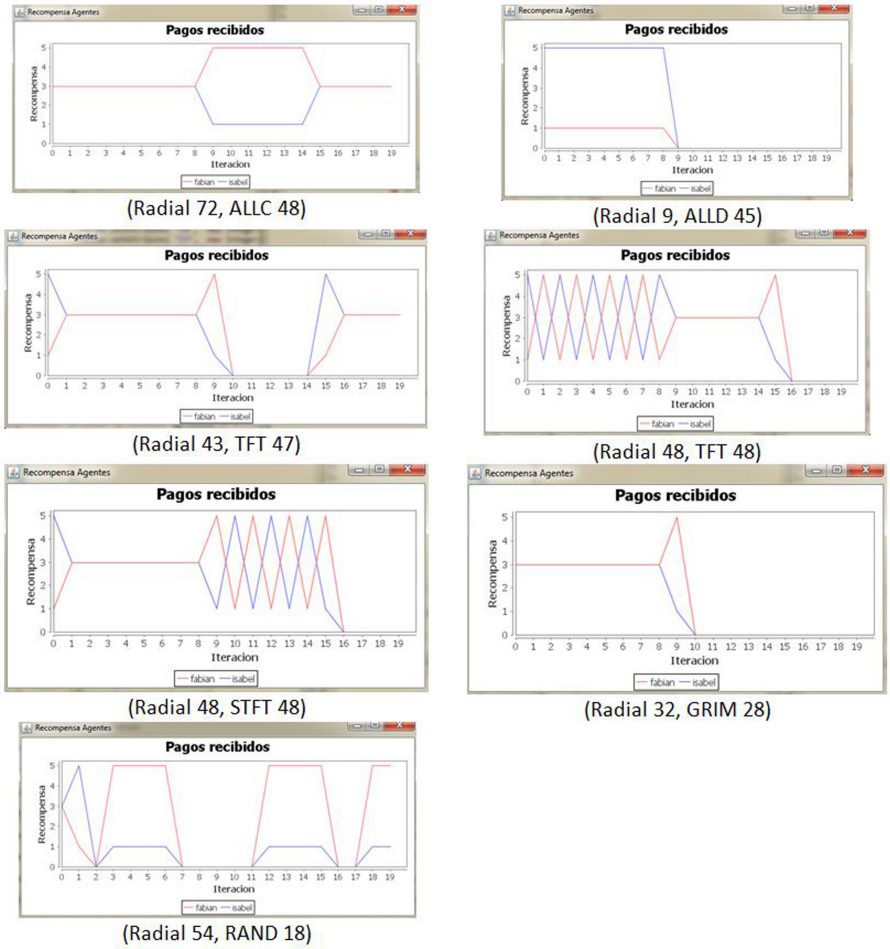


FIGURA 4.35. Resultados juego de la gallina usando modelamiento red de base radial

- Si el contrincante juega con ALLD, la mejor decisión para maximizar el pago es no-cooperar.
- Si el contrincante juega con RAND, la red neuronal adapta su estrategia de juego cambiando constantemente las decisiones en cada una de las rondas. Los resultados garantizan una maximización del pago.

4.7. Resumen

En el presente capítulo se muestra un laboratorio computacional para el desarrollo de modelos de simulación el el ámbito de la economía computacional basada en agentes. Adicionalmente, se desarrollan modelos de simulacíon para juegos repetitivos no cooperativos, en los cuales, se especifican estrategias de aprendizaje basado en técnicas de modelamiento del oponente.
Luego del proceso de experimentación se puede indicar los siguientes elementos:

Tal como sucedió en el capítulo II, al usar modelo del oponente ímplicito, los perceptores multicapa como las redes de base radial obtenidas por proceso de evolución vencen a estrategias estándar tradicionalmente utilizadas en la literatura, adicionalmente se pue-

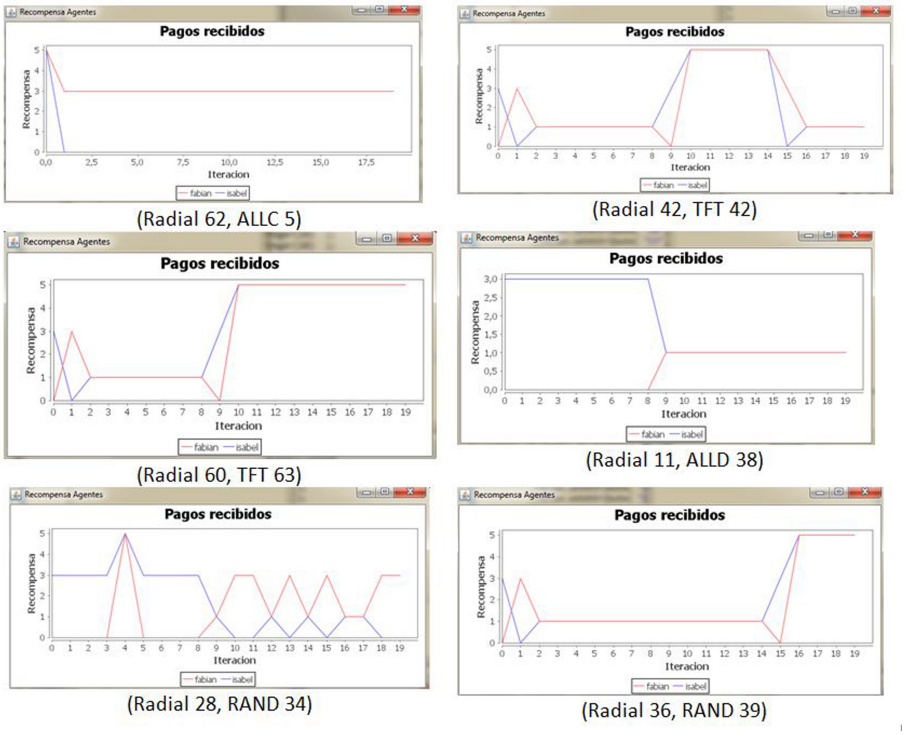


FIGURA 4.36. Resultados caza del ciervo usando modelamiento red de base radial

de evidenciar en cada uno de los juegos no cooperativos la emergencia de la cooperación cuando se maximiza el puntaje.

En este caso, los agentes son egoístas dado que explotan las debilidades el contrincante, esto por la posibilidad de enfrentarse múltiples veces. Para el caso del modelamiento del oponente explícito, se nota como existen una disminución en puntos respecto al caso anterior, por varias razones.

Existen juegos aleatorios al inicio de los enfrentamientos, lo que puede causar que los contrincantes tomen acciones radicales para los restantes juegos.

Los dos modelos de redes neuronales (perceptron, red de base radial) tienen a la cooperación, este caso puede ser evidenciado cuando se realizan enfrentamientos con TFT, TFTT.

El perceptron multicapa en general tiene mejor comportamiento que la red de base radial, se evidencia en los cambios que produce luego de obtener el modelo de aprendizaje cuando se enfrenta con ALLC, TFT, TFTT. Es decir, la red de base radial busca hacer un cambio de estrategia causando una disminución en la recompensa esperada en muchos casos.

Conclusiones

En el trabajo de investigación se presentaron esquemas de aprendizaje de estrategias de decisión en juegos repetitivos no cooperativos usando redes neuronales, especificamente, redes de base radial y perceptron multicapa.

Se plantearon diferentes escenarios con el fin de evaluar la capacidad de adaptación de las redes neuronales, entre ellos: modelos de aprendizaje ímplicito o offiline, en el cual, a través de procesos de neuroevolución se obtenían redes neuronales que tenían la capacidad de identificar una contra-estrategia para vencer a otras estrategias, en juegos estandar y torneos. Adicionalmente, modelos de aprendizaje explícito o online, en el cual, las redes neuronales a través de procesos de neuroevolución, tiene la capacidad de identificar la estrategia del oponente en tiempo de simulación, sin un conocimiento previo del contrincante, usando como fuente de información juegos previos y por lo tanto, obtener una contra-estraegia de juego.

Con el fin de analizar los diferentes juegos y facilitar el proceso de implementación, también se diseñó un lenguaje especifico de dominio textual en el cual los modeladores pueden plantear juegos no cooperativos, teniendo la capacidad de integrar estrategias de aprendizaje a los agentes participantes.
A partir de lo anterior, se pueden obtener la siguientes conclusiones.

- Los resultados obtenidos confirman que es posible ver los sistemas económicos como procesos de conocimiento que evolucionan en el tiempo y que pueden ser simulados en ambientes computaciones, en los cuales, los agentes tienen la capacidad de adaptarse a través de procesos de aprendizaje basados en computación evolutiva (algoritmos genéticos) e inteligencia colectiva. Esto se evidencia, con la emergencia de estrategias cuando se compiten contra estrategias predefinidas en dilema del prisionero, juego de la gallina y caza del ciervo.

  Los resultados experimentales muestran que la evolución de estrategias usando técnicas metaheurísticas (AG, PSO, redes neuronales) garantizan la maximización de pagos, por lo tanto, los juegos evolutivos pueden ser utilizados para estudiar las condiciones de cómo y porqué ciertos comportamientos en un entorno complejo puede ser aprendido por agentes con racionalidad limitada, a través de un proceso de adaptación guiado por planes estratégicos.

- Los perceptores multicapa como las redes de base radial obtenidas por proceso de evolución, vencen a estrategias estándar tradicionalmente utilizadas en la literatura

(ALLC, ALLD, TFT, TFTT, STFT, RAND, etc.), se puede evidenciar en la recompensa final obtenida en cada una de las competencias estándar, en cada uno de los juegos repetitivos no cooperativos analizados (dilema del prisionero, juego de la gallina y caza del ciervo).

- Las redes neuronales (perceptron multicapa, base radial) tienen comportamientos similares en procesos de evolución offline, los gráficos de recompensa siguen la misma tendencia cuando compiten en juegos repetitivos no cooperativos bajo la modalidad torneo. En general, se ve una fuerte tendencia a la cooperación mutua dado el puntaje promedio obtenido de 2.75 en el dilema del prisionero, 3.0 en juego de la gallina y 4.0 en la caza del ciervo.
  Sin embargo, en todos los juegos el perceptron tiene un leve puntaje mayor que las redes de base radial, sin embargo la diferencia no es significativa. Lo anterior, da sustento a la hipótesis inicial la cual indica que las redes neuronales evolucionadas pueden realizar aprendizaje de estrategias en juegos a pesar de actuar en ambientes con ruido.

- Las redes neuronales tienen comportamientos equivalentes a los algoritmos genéticos y enjambre de partículas (PSO), ampliamente estudiados como estrategias de aprendizaje de estrategias de decisión en juegos no cooperativos. Los gráficos de recompensa convergen siguiendo la misma tendencia y las recompensas finales obtenidas, luego de las competencias tipo torneos no tienen grandes diferencias.

- Los perceptrones multicapas obtenidos por procesos de evolución en línea (Modelamiento explícito) obtiene mejores resultados que las redes de base radial, en los juegos repetitivos no cooperativos analizados. Lo anterior, se ve reflejado en el comportamiento de los gráficos de recompensa obtenidos y en el puntaje final. Los perceptrones cuando compiten con las estrategias del CEC04, en general, siempre realizan el proceso de adaptación esperado, es decir, cooperan cuando predicen la maximización del pago.

  Las redes de base radial en su defecto, en algunas circunstancia (competencias contra ALLC, TFT, STFT) no encuentran la contra-estrategia, dado que hacen cambios bruscos en algunas iteraciones causando una disminución notable en la recompensa esperada.
  Basado en el análisis de los gráficos de recompensa obtenida se puede observar que el perceptron acierta en aproximadamente el 90 % en la adaptación de la estrategia y diferencia de la red de base radial que acierta en aproximadamente el 70 %.

- Comparando los resultados obtenidos entre la evolución de estrategias online y offline se puede indicar:
  El aprendizaje offline obtiene mayores puntajes que el proceso online, lo anterior, dado las múltiples veces que se puede competir con el contrincante. Es decir, se explotan fuertemente las debilidades del oponente. Adicionalmente, en el caso del aprendizaje online, los juegos iniciales en algunos casos predisponen al contrincante, ejemplo de ello es la competencia con estrategias como GRIM. El hecho que se realice una No cooperación hace que el contrincante nunca vuelva a cooperar.

  El aprendizaje online tiene más tendencia a la cooperación cuando puede predecir que el oponente va a cooperar. Las competencias contra ALLC, TFT, TFTT lo

evidencian. En su contraste, el modelo offline traiciona en el caso de ALLC y TFTT dado que identifica la tendencia a la cooperación.

- Los resultados del aprendizaje online son los esperados y son consistentes con los encontrados en la literatura, se puede decir, que son acertados dado que no siempre convergen a las estrategias dominantes, ejemplo de ello, es que para el dilema del prisionero no siempre tiende a la no cooperación, en su defecto emerge la cooperación en varios de los enfrentamientos.

- El desarrollo del laboratorio computacional facilitó la tarea de modelamiento de los juegos no cooperativos y por ende el análisis de los resultados. Adicionalmente, el proceso de integración con la librerías externas de toma de decisiones como la redes de base radial y perceptron multicapas, facilitó el análisis del proceso de aprendizaje, tanto offline (modelo del oponente implícito) como online (modelo del oponente explícito).

Trabajo futuro

Con el fin de complementar el trabajo de investigación desarrollado se plantea las siguientes propuestas como trabajo futuro:

- Extender el lenguaje específico de dominio con el fin de permitir definir las diferentes características del territorio que van a conformar el ambiente, por ejemplo: territorio de dos, tres dimensiones, recursos, entre otras.
- Integrar a la librería de simulación un módulo de reportes el cual de la opción al usuario de seleccionar entre diversos tipos de gráficos, así como la comparación de diversos escenarios sobre modelos de simulación especificados.
- Extender el lenguaje específico de dominio con el fin de soportar nuevas estructuras de control, por ejemplo, estructuras iterativas, funciones que permitan realizar filtros sobre elementos de los agentes y ambiente: agentes con atributos determinados, recursos del territorio, etc.
- Con el fin de complementar el estudio sobre los juegos repetitivos no cooperativos estudiados, usar otras métricas diferentes a las basadas en recompensa, tales como: convergencia al equilibro de Nash, que permita determinar la distancia entre los resultados obtenidos por los agentes a la estrategia dominante.
- Migrar modelos de simulación basados en agentes realizados en los diferentes proyectos desarrollados en el grupo de investigación ALIFE, a la plataforma de simulación UNALCOL.
- Integrar a los agentes utilizados en las competencias de los juegos no cooperativos un modelo mixto, que integre los modelos de aprendizaje online y offline. Lo anterior, permtirá tener la capacidad de maximizar el puntaje cuando se enfrente con contrincantes conocidos y adicionalmene, poner defenderse cuando aparezcan nuevos oponentes.
- Plantear el diseño experimental o usar un optimizador global, con el fin de determinar la mejor combinación de parámetros para las diversas técnicas utilizadas en los procesos de experimentación. A pesar que con los procesos de experimentación realizados se lograron buenos resultados, se daría más soporte científico a los resultados finales obtenidos.

Bibliografía

[1] Begel A and Klopfer E, *Starlogo tng: An introduction to game development*, 2005.

[2] Errity A, *Evolving strategies for the prisoner's dilemma*, July 2003.

[3] Repenning A, *Agentsheets: an interactive simulation environment with end-user programmable agents*, Interaction 2000, Tokyo, Japan, 2000.

[4] Bernhard Sendhoff Alexandra Mark and Heiko Wersing, *Evolution of a learning and anticipating decision system*, SOAVE 2004 3rd Workshop on SelfOrganization of AdaptiVE Behavior, 2004, pp. 271–281.

[5] Floortje Alkemade, *Evolutionary agent-based economics*, Technische Universiteit Eindhoven, 2004.

[6] Skyrms B, *The Stag Hunt and the Evolution of Social Structure*, illustrated edition ed., Cambridge University Press, December 2003.

[7] Sander C.J. Bakkes, Pieter H.M. Spronck, and H. Jaap van den Herik, *Opponent modelling for case-based adaptive game {AI}*, Entertainment Computing **1** (2009), no. 1, 27 – 37.

[8] David F. Batten, *Discovering artificial economics*, Westview Press, Boulder, Colo. [u.a.], 2000.

[9] Ann Maria Bell, *Reinforcement learning rules in a repeated game*, Computational Economics **18** (2001), 89–110, 10.1023/A:1013818611576.

[10] M. Bergsma, *Opponent modelling in machiavelli*, Bachelor Thesis,Department of Knowledge Engineering, Maastricht University, Maastricht, 2005.

[11] Paulo Blikstein, Dor Abrahamson, and Uri Wilensky, *Netlogo: Where we are, where we re going*, In Proceedings of the annual meeting of Interaction Design and Children, Press, 2005.

[12] Paulo Blikstein, Dor Abrahamson, and Uri Wilensky, *Netlogo: Where we are, where we're going*, Proceedings of the annual meeting of Interaction Design and Children (Boulder, Colorado) (M. Eisenberg and A. Eisenberg, eds.), 2005.

[13] A. Browning, L; Colman, *Evolution of coordinated alternating reciprocity in repeated dyadic games*, journal of Theoretical Biology **229** (2004), no. 4, 549–557.

[14] Richard Brunauer, Andreas Löcker, Helmut A. Mayer, Gerhard Mitterlechner, and Hannes Payer, *Evolution of iterated prisoner's dilemma strategies with different history lengths in static and cultural environments*, Proceedings of the 2007 ACM symposium on Applied computing (New York, NY, USA), SAC '07, ACM, 2007, pp. 720–727.

[15] Triebig C, Credner T, Kigl F, Fischer P, Leskien T, Deppisch A, and Landvogt S, *Simulating automatic high bay warehouses using agents*, Multi-Agent Systems and Applications IV, Lecture Notes in Computer Science, vol. 3690, Springer Berlin Heidelberg, 2005, pp. 653–656.

[16] Kai Cao, Xiao Feng, and Hui Wan, *Applying agent-based modeling to the evolution of eco-industrial systems*, Ecological Economics **68** (2009), no. 11, 2868–2876.

[17] Bengt Carlsson, *Chapter 7 simulating how to cooperate in iterated chicken and prisoner dilemma games*, 2001.

[18] Yu Chen, Xiaomeng Su, Jinghai Rao, and Hui Xiong, *Agent-based microsimulation of economy from a complexity perspective*, July 2000.

[19] Siang Yew Chong and Xin Yao, *Multiple choices and reputation in multiagent interactions*, Evolutionary Computation, IEEE Transactions on **11** (2007), no. 6, 689 –711.

[20] Dinther Clemens, *Agent-based simulation for research in economics*, Handbook on Information Technology in Finance (Detlef Seese, Christof Weinhardt, and Frank Schlottmann, eds.), International Handbooks on Information Systems, Springer Berlin Heidelberg, 2008, pp. 421–442.

[21] Andrew T Crooks, *The repast simulation/modelling system for geospatial simulation*, 2007.

[22] Rivero D, Dorado J, Rabu nal J, and Pazos A, *Generation and simplification of artificial neural networks by means of genetic programming*, Neurocomput. **73** (2010), no. 16-18, 3200–3223.

[23] Kurt Dopfer, *The economic agent as rule maker and rule user: Homo sapiens oeconomicus*, Journal of Evolutionary Economics **14** (2004), no. 2, 177–195.

[24] Kurt Dopfer, John Foster, and Jason Potts, *Micro-meso-macro*, Journal of Evolutionary Economics **14** (2004), no. 3, 263–279.

[25] Kurt Dopfer and Jason Potts, *Evolutionary realism: a new ontology for economics*, Journal of Economic Methodology **11** (2004), no. 2, 195–212.

[26] Hemberg E, Gilligan C, O'Neill M, and Brabazon A, *A grammatical genetic programming approach to modularity in genetic algorithms*, Proceedings of the 10th European conference on Genetic programming (Berlin, Heidelberg), EuroGP'07, Springer-Verlag, 2007, pp. 1–11.

[27] Toro E, Restrepo Y, and Granada S., *Adaptaciíon de la técnica de particle swarm al problemas de secuenciamiento de trareas*, Scientia Et Technica **32** (2006).

[28] Alkemade F, La Poutre H, and Amman H, *On social learning and robust evolutionary algorithm design in economic games*, Evolutionary Computation, 2005. The 2005 IEEE Congress on, vol. 3, sept. 2005, pp. 2445 – 2452 Vol. 3.

[29] Luna F, *Computable learning, neural networks and institutions*, Tech. report, 1996.

[30] A Rocha F Okayuma, R Bordini, *Elms: An enviroment description languaje for multiagent simulation*, First International Workshop, E4MAS 2004, New York, NY, 2004.

[31] Karl O. Faxen, *An evolutionary theory of economic change*, Journal of Economic Behavior Organization **6** (1985), no. 1, 92–94.

[32] Magda Fontana, *Can neoclassical economics handle complexity? the fallacy of the oil spot dynamic*, Journal of Economic Behavior Organization **76** (2010), no. 3, 584–596.

[33] N. Franken and A.P. Engelbrecht, *Particle swarm optimization approaches to coevolve strategies for the iterated prisoner's dilemma*, Evolutionary Computation, IEEE Transactions on **9** (2005), no. 6, 562 – 579.

[34] E. Frias-Martinez, A. Sanchez, and J. Velez, *Support vector machines versus multilayer perceptrons for efficient off-line signature recognition*, Engineering Applications of Artificial Intelligence **19** (2006), no. 6, 693 – 704, ¡ce:title¿Special Section on Innovative Production Machines and Systems (I*PROMS)¡/ce:title¿.

[35] Pappa G and Freitas A, *Automatically evolving rule induction algorithms tailored to the prediction of postsynaptic activity in proteins*, Intell. Data Anal. **13** (2009), no. 2, 243–259.

[36] Gomez Jonatan Giraldo Fabian, *Aprendizaje de estrategias de decisión en juegos repetitivos no cooperativos*, Tecnura 0123-921X **17** (2013), no. 35, 63–76.

[37] David E. Goldberg, *Genetic algorithms in search, optimization and machine learning*, 1st ed., Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[38] Dawid H, *Adaptive learning by genetic algorithms: Analytical results and applications to economic models*, 1st ed., Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1999.

[39] Heap Shaun Hargreaves and Varoufakis Yanis., *Game theory : a critical text*, 2nd ed. ed., Routledge, New York :, 2004 (English).

[40] Shu heng Chen, *Fundamental issues in the use of genetic programming in agent-based computational economics. working paper, ai-econ research*, 2001.

[41] Ralf Herbrich, Max Keilbach, Thore Graepel, Peter Bollmann-Sdorra, and Klaus Obermayer, *Neural networks in economics: Background, applications and new developments*, in Advances in Computational Economics: Computational Techniques for Modelling Learning in Economics, Kluwer Academics, 1999, pp. 169–196.

[42] Philip Hingston and Graham Kendall, *Learning versus evolution in iterated prisoner's dilemma*, in Proceedings of the 2004 Congress on Evolutionary Computation (CEC 04). IEEE Publications, 2004, pp. 364–372.

[43] Cho In-Koo, *Bounded rationality, neural network and folk theorem in repeated games with discounting*, Economic Theory **4** (1994), 935–957 (English).

[44] Villalobos J. and Mejía G., *Redes de petri y algoritmos genéticos, una propuesta para la programación de sistemas de manufactura flexible*, Ingenieria y Universidad **10** (2010), no. 1.

[45] XiangHua Jin, DongHeon Jang, and TaeYong Kim, *Evolving game agents based on adaptive constraint of evolution*, Convergence Information Technology, 2007. International Conference on, nov. 2007, pp. 1241 –1246.

[46] Metcalfe J.S. and Foster John, *Evolution and economic complexity / edited by j. stanley metcalfe and john foster*, Edward Elgar, Northampton, Mass. :, 2004 (English).

[47] Chellapilla K and Fogel D, *Evolution, neural networks, games, and intelligence*, 1999.

[48] Eric Klopfer and Hal Scheintaub, *Starlogo tng: science in student-programmed simulations*, Proceedings of the 8th international conference on International conference for the learning sciences - Volume 3, ICLS'08, International Society of the Learning Sciences, 2008, pp. 59–60.

[49] Donald E. Knuth, *The TEXbook*, Computers and Typesetting, pub-AW, pub-AW:adr, 1986.

[50] D. Koesrindartoto, Junjie Sun, and L. Tesfatsion, *An agent-based computational laboratory for testing the economic reliability of wholesale power market designs*, Power Engineering Society General Meeting, 2005. IEEE, june 2005, pp. 2818 – 2823 Vol. 3.

[51] John R. Koza, *Genetic programming - on the programming of computers by means of natural selection.*, Complex adaptive systems, MIT Press, 1993.

[52] Tuba Kurban and Erkan Besdok, *A comparison of rbf neural network training algorithms for inertial sensor based terrain classification*, Sensors **9** (2009), no. 8, 6312–6329.

[53] Browning L and Colman A, *Evolution of coordinated alternating reciprocity in repeated dyadic games*, Journal of Theoretical Biology (2004), 549–557.

[54] Llano L, Hoyos A, Arias F, and Velásquez J., *Comparación del desempeno de funciones de activación en redes feedforward para aproximar funciones de datos con y sin ruido*, RASI (2007), 79–88.

[55] Asher G. Lipson, *Empirically evaluating multiagent reinforcement learning algorithms in repeated games*, 2005.

[56] Alan J. Lockett, Charles L. Chen, and Risto Miikkulainen, *Evolving explicit opponent models in game playing*, Proceedings of the 9th annual conference on Genetic and evolutionary computation (New York, NY, USA), GECCO '07, ACM, 2007, pp. 2106–2113.

[57] Izquierdo L.R, Galan J.M, Santos J.I, and del Olmo R, *Modelado de sistemas complejos mediante simulación basada en agentes y mediante dinámica de sistemas*, Empiria. Revista de Metodología de Ciencias Sociales **1** (2008), no. 16, 85–112.

[58] Sean Luke, Claudio Cioffi-Revilla, Liviu Panait, Keith Sullivan, and Gabriel Balan, *Mason: A multiagent simulation environment*, Simulation **81** (2005), no. 7, 517–527.

[59] Melanie M, *An Introduction to Genetic Algorithms (Complex Adaptive Systems)*, third printing ed., A Bradford Book, February 1998.

[60] Charles M. Macal and Michael J. North, *Agent-based modeling and simulation*, Winter Simulation Conference, 2009, pp. 86–98.

[61] Nelson Minar, Rogert Burkhart, Chris Langton, and Manor Askenazi, *The swarm simulation system: A toolkit for building multi-agent simulations*, Working papers, Santa Fe Institute, 1996.

[62] Franken N. and Engelbrecht A.P., *Pso approaches to coevolve ipd strategies*, Evolutionary Computation, 2004. CEC2004. Congress on, vol. 1, june 2004, pp. 356 – 363 Vol.1.

[63] Richard R. Nelson and Sidney G. Winter, *Evolutionary theorizing in economics*, Journal of Economic Perspectives **16** (2002), no. 2, 23–47.

[64] Christoph Oechslein, Alexander Hornlein, and Franziska Klugl, *Evolutionary optimization of societies in simulated multi-agent systems*, 2000.

[65] Fabio Y. Okuyama, Rafael H. Bordini, and Antônio Carlos da Rocha Costa, *Elms: an environment description language for multi-agent simulation*, Proceedings of the First international conference on Environments for Multi-Agent Systems (Berlin, Heidelberg), E4MAS'04, Springer-Verlag, 2005, pp. 91–108.

[66] Chiong R, *Applying genetic algorithms to economy market using iterated prisoner's dilemma*, Proceedings of the 2007 ACM symposium on Applied computing (New York, NY, USA), SAC '07, ACM, 2007, pp. 733–737.

[67] Lum R, Meighan M, and Alfaro H, *Using neat to train a player for the iterated prisoner dilemma*, Machine Learning FALL, 2008.

[68] Stuart J. Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach*, Pearson Education, 2003.

[69] Stuart J. Russell, Peter Norvig, John F. Candy, Jitendra M. Malik, and Douglas D. Edwards, *Artificial intelligence: a modern approach*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.

[70] H.A. Sally and R.M.A. Muhammad, *A survey of game theory using evolutionary algorithms*, Information Technology (ITSim), 2010 International Symposium in, vol. 3, june 2010, pp. 1319 –1325.

[71] L. Samuelson, *Evolution and game theory*, Journal of Economic Perspectives **16** (2002), no. 2.

[72] H. Scholl, *Agent based and system dynamics modeling: A call for cross study and joint research*, Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 3 - Volume 3 (Washington, DC, USA), HICSS '01, IEEE Computer Society, 2001, pp. 3003–.

[73] Alfred V. Aho Ravi Sethi, *Compiladores, principios, tecnicas y herramientas*, Pearson Educacion., 1990.

[74] Daniel Sgroi and Daniel J. Zizzo, *Neural networks and bounded rationality*, Physica A: Statistical Mechanics and its Applications **375** (2007), no. 2, 717–725.

[75] Yu Shun-kun and Yuan Jia-hai, *Agent-based computational economics: methodology and its application in electricity market research*, Power Engineering Conference, 2005. IPEC 2005. The 7th International, 29 2005-dec. 2 2005, pp. 1 –415.

[76] T. Stockheim, M. Schwind, and W. Konig, *A model for the emergence and diffusion of software standards*, System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on, jan. 2003, p. 10 pp.

[77] Jeff Heaton T., *Introduction to neural networks with java*, Heaton Research, Inc., 2005.

[78] Riechmann T, *Learning in economics: Analysis and application of genetic algorithms*, Springer, 2001.

[79] Leigh Tesfatsion, *Agent-based computational economics*, ISU Economics Working Paper **1** (2003), 2003.

[80] Leigh Tesfatsion, *Agent-based computational economics: A constructive approach to economic theory*, Handbook of Computational Economics (Leigh Tesfatsion and Kenneth L. Judd, eds.), Handbook of Computational Economics, vol. 2, Elsevier, 2006, pp. 831–880.

[81] Leigh S Tesfatsion, *Agent-based computational economics: Modeling economies as complex adaptive systems*, Staff general research papers, Iowa State University, Department of Economics, 2008.

[82] S. Tisue and U. Wilensky, *NetLogo: a simple environment for modeling complexity*, 2004.

[83] Seth Tisue and Uri Wilensky, *Netlogo: A simple environment for modeling complexity*, in International Conference on Complex Systems, 2004, pp. 16–21.

[84] Ioannis Tsoulos, Dimitris Gavrilis, and Euripidis Glavas, *Neural network construction and training using grammatical evolution*, Neurocomput. **72** (2008), no. 1-3, 269–277.

[85] Robert van Rooij, *The stag hunt and the evolution of social structure.*, Studia Logica **85** (2007), no. 1, 133–138.

[86] S. Widergren, J. Sun, and L. Tesfatsion, *Market design test environments*, Power Engineering Society General Meeting, 2006. IEEE, 0-0 2006, p. 6 pp.

[87] Michael Woolridge and Michael J. Wooldridge, *Introduction to multiagent systems*, John Wiley & Sons, Inc., New York, NY, USA, 2001.

[88] Yau Y.J and J. Teo, *An empirical comparison of non-adaptive, adaptive and self-adaptive co-evolution for evolving artificial neural network game players*, Cybernetics and Intelligent Systems, 2006 IEEE Conference on, june 2006, pp. 1 –6.