Aprendizaje en Agentes con Mecanismos de Selección de Acción basados en Redes de Comportamiento

por

Selene Ivette Jiménez Castillo

Tesis submitida por
LANIA - Universidad Veracruzana
para el grado de
Maestría en Inteligencia Artificial

Facultad de Física e Inteligencia Artificial
Universidad Veracruzana
2005

Índice general

Introducción ix

Objetivos xiii

1. Redes de Comportamiento, un Mecanismo de Selección de Acción **1**
1.1. Agente . . . 1
1.1.1. Ambiente . . . 2
1.1.2. Autonomía y adaptación . . . 3
1.1.3. La selección de acción . . . 4
1.1.4. Aprender de la experiencia . . . 6
1.2. Mecanismos de selección de acción . . . 6
1.2.1. Arquitectura de Subsumción . . . 7
1.2.2. Redes de Comportamiento . . . 8
1.2.3. Hamsterdam . . . 10
1.3. Redes de Comportamiento . . . 11
1.3.1. Algoritmo . . . 11
1.3.2. Modelo matemático . . . 13
1.3.3. Limitaciones y críticas . . . 16

2. Aprendizaje **19**
2.1. ¿Qué es aprendizaje? . . . 19
2.1.1. ¿Por qué, qué y de dónde aprender? . . . 20
2.2. Tipos de Aprendizaje Autómatico . . . 21
2.3. El Aprendizaje por Reforzamiento . . . 22
2.3.1. Concepto . . . 22
2.3.2. Historia del ApR . . . 23
2.4. Modelo del Aprendizaje por Reforzamiento . . . 26
2.4.1. Elementos . . . 27
2.4.2. Algoritmo general . . . 28
2.5. Equilibrio explotación-exploración . . . 29
2.6. El Problema del Bandido . . . 30
2.7. Algoritmos . . . 31
2.7.1. Algoritmo del Bandido . . . 31
2.7.2. Métodos de evaluación de acciones . . . 31
2.7.2.1. Estrategias codiciosas . . . 32

2.7.2.2. Estrategias aleatorias . . . 33
2.7.3. Estrategias basadas en vectores . . . 34
2.7.4. Métodos de comparación de reforzamientos . . . 34
2.7.5. Técnicas basadas en intervalos . . . 35

3. Antecedentes **37**
3.1. Herramienta ABC . . . 37
3.1.1. Extensión . . . 41
3.1.2. Alcance . . . 43
3.2. Herramienta XLearn . . . 44
3.3. Estudios etológicos . . . 45
3.4. Mejoramiento propuesto . . . 47

4. Módulo de Aprendizaje por Reforzamiento con Redes de Comportamiento **49**
4.1. Red de Comportamiento y Aprendizaje por Reforzamiento . . . 49
4.2. Módulo de aprendizaje . . . 50
4.2.1. Algoritmos de Aprendizaje por Reforzamiento . . . 52
4.2.1.1. Algoritmo del Bandido . . . 52
4.2.1.2. Algoritmo Greedy . . . 53
4.2.1.3. Algoritmo ε-Greedy . . . 54
4.2.1.4. Algoritmo Softmax . . . 54
4.2.1.5. Algoritmo LRP . . . 55
4.2.1.6. Algoritmo LR . . . 55
4.2.1.7. Algoritmo RC . . . 55
4.2.1.8. Algoritmo IE . . . 56
4.2.1.9. Comparativos . . . 56
4.2.2. Ambiente . . . 57
4.2.3. Impacto del ambiente en la selección de acciones . . . 58
4.3. Impacto del aprendizaje en la Red de Comportamiento . . . 59
4.4. Variaciones ambientales . . . 60
4.4.1. Cantidad de acciones . . . 60
4.4.1.1. Ambiente escaso . . . 61
4.4.1.2. Ambiente abundante . . . 61
4.4.2. Distribución de acciones . . . 61
4.4.2.1. Aleatoria . . . 62
4.4.2.2. Por zonas . . . 62
4.4.3. Datos y gráficas obtenidas . . . 63

5. Simulaciones **65**
5.1. Simulaciones sin aprendizaje . . . 65
5.1.1. Estudio etológico de Rodríguez Luna . . . 66
5.1.1.1. Modelo ABC_RodríguezLuna . . . 67
5.1.2. Estudio etológico de Domingo Balcells . . . 69
5.1.2.1. Modelo ABC_DomingoBalcells . . . 70
5.2. Simulaciones con aprendizaje . . . 72
5.2.1. Modelo ABC_RodríguezLuna . . . 74

5.2.2. Modelo ABC_DomingoBalcells 77
5.3. Desempeño de los algoritmos de aprendizaje 79
5.3.1. Por recompensa promedio y factor XLearn 80
5.3.1.1. Desempeño en relación al número de opciones 81
5.3.1.2. Desempeño en relación al tipo de mundo 82
5.3.1.3. Desempeño en relación al caso probabilístico 82
5.3.2. Por solicitud del óptimo 84
5.3.2.1. Desempeño en relación al número de opciones 86
5.3.2.2. Desempeño en relación al tipo de mundo 86
5.3.2.3. Desempeño en relación al caso probabilístico 87
5.3.3. Observaciones 88
5.3.3.1. Algoritmos con tendencia a obtener la misma recompensa 88
5.3.3.2. Algoritmos con tendencia a tener el mismo porcentaje de solicitud del óptimo 89
5.3.3.3. Porcentaje de algoritmos que convergen a la recompensa deseada 89
5.3.3.4. Porcentaje de algoritmos con la máxima recompensa por corrida 90
5.3.3.5. Comparación de porcentaje de sobrevivencia 91

Conclusiones **93**

A. Apéndice **99**

Bibliografía **114**

Índice de figuras

1.1. Descripción general de un agente 2
1.2. Arquitectura de Subsumción 8
1.3. Red de Comportamiento 9
1.4. Selección de acción en Hamsterdam 10

2.1. Agente ApR 23
2.2. Breve historia del ApR 25
2.3. Modelo estándar del Aprendizaje por Reforzamiento 26
2.4. Esquematización del proceso de Aprendizaje por Reforzamiento 29

3.1. Ventana principal de ABC (primera versión) 38
3.2. Gráfica de activación en el tiempo 39

4.1. Modelo de Red de Comportamiento con Aprendizaje por Reforzamiento 50
4.2. Ambiente del módulo de aprendizaje de ABC 57
4.3. Mundo de distribución aleatoria con cantidad escasa 61
4.4. Mundo de distribución aleatoria con cantidad abundante 62
4.5. Mundo de distribución por zonas con cantidad escasa 63
4.6. Mundo de distribución por zonas con cantidad abundante 63

5.1. Modelo ABC_RodríguezLuna 67
5.2. Segmento de gráfica de activación en el tiempo 68
5.3. Patrón de actividades del modelo ABC_RodríguezLuna 69
5.4. Modelo ABC_DomingoBalcells 70
5.5. Patrón de actividades del modelo ABC_DomingoBalcells 71
5.6. Segmento de gráfica de activación en el tiempo 72
5.7. Curva de aprendizaje del ejemplo del modelo ABC_RodríguezLuna 75
5.8. Curva de convergencia del ejemplo del modelo ABC_RodríguezLuna 76
5.9. Curva de aprendizaje del ejemplo del modelo ABC_DomingoBalcells 78
5.10. Curva de convergencia del ejemplo del modelo ABC_DomingoBalcells 78

A.1. Gráfica de activación en el tiempo del modelo ABC_RodríguezLuna 100
A.2. Gráfica de activación en el tiempo - Ejemplo ABC_RodríguezLuna 103
A.3. Patrón de actividades - Ejemplo 104
A.4. Curvas de aprendizaje y convergencia - Corrida general del ejemplo 105
A.5. Curvas de aprendizaje - Corrida general del ejemplo 105

A.6. Gráfica de activación en el tiempo del modelo ABC_DomingoBalcells 107
A.7. Gráfica de activación en el tiempo - Ejemplo ABC_DomingoBalcells 110
A.8. Patrón de actividades - Ejemplo 111
A.9. Curvas de aprendizaje y convergencia - Corrida general del ejemplo 112
A.10.Curvas de aprendizaje - Corrida general del ejemplo 112
A.11.Gráfica de tiempo de búsqueda 113

Índice de cuadros

5.1. Porcentaje y tiempos promedio para estaciones húmeda (EH) y seca (ES). 66
5.2. Matriz de probabilidades de transición para la estación húmeda. 66
5.3. Matriz de probabilidades de transición para la estación seca. 67
5.4. Porcentaje y tiempos promedio del modelo ABC_RodríguezLuna. 68
5.5. Matriz de probabilidades de transición del modelo ABC_RodríguezLuna. 68
5.6. Porcentajes reportados por Domingo. 69
5.7. Porcentaje y ciclos promedio del modelo ABC_DomingoBalcells. 71
5.8. Matriz de probabilidades de transición del modelo ABC_DomingoBalcells. 71
5.9. Casos probabilísticos de recompensa para $N = 2$. 73
5.10. Casos probabilísticos de recompensa para $N = 3$. 73
5.11. Casos probabilísticos de recompensa para $N = 4$. 73
5.12. Parámetros de los algoritmos de aprendizaje y sus valores. 74
5.13. Porcentaje y tiempos promedio del ejemplo del modelo ABC_RodríguezLuna. . . 76
5.14. Matriz de probabilidades de transición del ejemplo del modelo ABC_RodríguezLuna. 76
5.15. Porcentaje y ciclos promedio del ejemplo del modelo ABC_DomingoBalcells. . . . 79
5.16. Matriz de probabilidades de transición del ejemplo del modelo ABC_Domingo-Balcells. 79
5.17. Jerarquización de algoritmos y comparativos por recompensa promedio y XLearn para el modelo ABC_RodríguezLuna. 80
5.18. Jerarquización de algoritmos y comparativos por recompensa promedio y XLearn para el modelo ABC_DomingoBalcells. 81
5.19. Jerarquización global de algoritmos y comparativos por recompensa promedio y XLearn. 81
5.20. Jerarquización de mundos en función a la recompensa promedio recibida - Modelos. 82
5.21. Jerarquización de los tipos de mundo en base a la recompensa promedio recibida. 82
5.22. Jerarquización de los casos probabilísticos en base al factor XLearn. 83
5.23. Diferencias entre las probabilidades de recompensa 83
5.24. Jerarquización de los casos probabilísticos en base a la distancia entre probabilidades. 84
5.25. Jerarquización de algoritmos por convergencia - modelo ABC_RodríguezLuna. . . 85
5.26. Jerarquización de algoritmos por convergencia - modelo ABC_DomingoBalcells. . 85
5.27. Jerarquización global de algoritmos ApR por convergencia al óptimo. 86
5.28. Porcentaje de convergencia según el tipo de mundo - modelo ABC_RodríguezLuna. 86
5.29. Porcentaje de convergencia según el tipo de mundo - modelo ABC_DomingoBalcells. 86

5.30. Jerarquización de los tipos de mundo en función a la convergencia al óptimo - ambos modelos 87
5.31. Jerarquización global de los casos probabilísticos en base al porcentaje de convergencia. 87
5.32. Frecuencia de algoritmos con tendencia a obtener la misma recompensa. 88
5.33. Frecuencia de algoritmos con tendencia a tener el mismo porcentaje de solicitud del óptimo. 89
5.34. Jerarquización de algoritmos que convergieron a la recompensa deseada. 90
5.35. Jerarquización de algoritmos en base a la obtención de la máxima recompensa por corrida. 90
5.36. Sobrevivencia en los mejores algoritmos ApR y en el comparativo SinApR. 91
5.37. Jerarquización de mundos en función al porcentaje de sobrevivencia promedio - Algoritmos ApR y SinApR. 91

Introducción

Uno de los temas de mayor importancia en la Inteligencia Artificial es el referente a la toma de decisiones que debe realizar un agente autónomo, para decidir qué hacer o cómo comportarse en cada momento de su existencia. Esto es conocido como el problema de la selección de acción, y ha sido ampliamente estudiado; en el presente trabajo se enfrenta la selección de acción mediante la interacción de dos procesos: un mecanismo de selección conocido como Redes de Comportamiento [Mae89, Mae90a, Mae90b] y un aprendizaje de políticas de acción por Reforzamiento [Kae93, SyB98].

Computacionalmente, la herramienta diseñada implementa el algoritmo de Redes de Comportamiento [Mae90a], un Mecanismo de Selección de Acción para modelar las interacciones de un agente con su ambiente al momento de elegir qué hacer, teniendo que decidir, en base a un conjunto de comportamientos posibles pre-compilados, lo que va a ejecutar. Por su parte, el Aprendizaje por Reforzamiento (ApR) también está relacionado con la toma de decisiones de un agente, pero únicamente basa su selección de acción en torno a una evaluación de las acciones que ha tomado, aprendiendo con base en su experiencia. Cabe destacar que las Redes de Comportamiento son estáticas al estar precompiladas, en tanto que el Aprendizaje por Reforzamiento introduce un aspecto dinámico a la selección de acción.

En este trabajo, el módulo de Aprendizaje por Reforzamiento implementado actúa sobre un módulo de comportamiento particular perteneciente a la Red de Comportamiento. El objetivo del módulo de aprendizaje es que una conducta sujeta a técnicas de aprendizaje experimente una *especialización*, ante la posibilidad de elegir una opción de entre varias opciones para la conducta; dicho en otras palabras, que mediante aprendizaje se logre la optimización de la selección de opciones del agente, quien deberá aprender a identificar a la mejor opción de entre todas aquellas que le sea posible ejecutar, a fin de que siempre elija la opción que mejor desempeño tenga dados el ambiente del agente y sus motivaciones.

Para una mejor comprensión de estos procesos, considere un animal viviendo en su medio ambiente, teniendo continuamente que tomar decisiones a fin de satisfacer sus necesidades básicas, como alimentarse, descansar, huir de sus depredadores, explorar su entorno, etc. Cada una de estas actividades implica la realización de muchas otras, por ejemplo, para alimentarse tiene que buscar alimento y para descansar debe tener un lugar que le proporcione comodidad y seguridad. Aún más, al ingerir alimento, los animales desarrollan una preferencia por aquellos que les proporcionan más nutrientes o que son ricos en reservas energéticas. Estas actividades implican aprender a partir de lo que se experimenta; la vida del animal es un aprendizaje constante, a costa de la superviviencia.

El animal puede ser visto como un agente, ya que vive en un ambiente que debe conocer y sobre el que va a actuar al decidir qué hacer. La Red de Comportamiento de este agente se conforma de módulos que representan las diversas actividades del animal (alimentar, descansar, explorar, evitar depredación, etc.). El aprendizaje en esta Red puede aplicarse a aquellas actividades que requieran una selección más especializada; como un ejemplo, el comportamiento *alimentar* puede ser objeto de aprendizaje para decidir el tipo de alimento que el animal debe ingerir ante varias opciones alimenticias. Situaciones similares a las de este agente/animal son consideradas en la parte experimental de la investigación.

Cabe mencionar que los experimentos realizados tienen sus fundamentos en estudios etológicos y en investigaciones computacionales que plantean el uso y desarrollo de modelos que ayuden a explicar las observaciones de campo. En particular, en este caso se analiza el comportamiento de forrajeo de los monos aulladores (*Alouatta Palliata*), y computacionalmente se considera el Proyecto Monots, que durante años se ha dedicado al modelado de tal conducta [Mar96a, Mar96b, Gar97, Gue97a, Gue97b, Gue98].

Este trabajo plantea primero los fundamentos teóricos de los procesos de selección de acción mencionados: la Red de Comportamiento y el Aprendizaje por Reforzamiento. Se describe además ABC [Gue97c, Mon98], una herramienta para simular Redes de Comportamiento. Finalmente, se explica la implementación del aprendizaje y el impacto que tiene ante diversos experimentos planteados. La organización capitular de la tesis es la siguiente:

En el capítulo uno se exponen diversos conceptos de *agente*, así como lo referente a los ambientes en los que un agente puede estar inmerso y las ideas de autonomía y adaptación, características deseables en todo agente. Como punto principal de este capítulo se explica lo que es la selección de acción y los diversos mecanismos existentes para llevarla al cabo, de los cuales se destaca finalmente la arquitectura bajo la que está elaborada la implementación de esta investigación: las Redes de Comportamiento, propuestas por Pattie Maes en su artículo "How do the right thing" [Mae90a], así como en "Situated agents can have goals" [Mae90b] y "A bottom-up mechanism for behavior selection in an artificial creature"[Mae91a], de las que se indica su funcionamiento tanto en algoritmo como en modelo matemático.

El capítulo dos introduce el concepto de aprendizaje y plantea la importancia de aprender en un contexto de agente computacional. El énfasis se concentra en el enfoque que ha sido tomado para el desarrollo del módulo de aprendizaje propuesto en esta investigación: el Aprendizaje por Reforzamiento. Se da una breve reseña histórica de ApR y se presentan sus características principales, tanto en sus componentes y algoritmo general, como en el problema de decidir entre aprovechar el conocimiento obtenido o buscar nuevo conocimiento en el ambiente. Se presenta un escenario simplificado del mismo: el problema del Bandido, así como diversos métodos que enfrentan este problema desde varias perspectivas de solución, como la evaluación de acciones, los vectores de probabilidad, la comparación de reforzamiento y la construcción de intervalos de confianza.

En el capítulo tres se presentan los antecedentes inmediatos de este trabajo. En primer lugar, la herramienta ABC, una implementación de Redes de Comportamiento multi-aplicación, de-

sarrollada en 1997 por Alejandro Guerra [Gue97c] y extendida en 1998 por Fernando Montes [Mon98]. Esta herramienta ha demostrado gran aplicabilidad en diferentes entornos: se muestran los resultados obtenidos al modelar el robot de Charniak, el Mundo de los Cubos y, de gran importancia a esta investigación, el Proyecto Monots. De este último trabajo surge la inquietud de realizar un estudio más profundo de simulación del comportamiento animal, específicamente del mono aullador. Montes modifica ABC para emular la conducta de forrajeo de este animal, experimentando con variaciones en la Red de Comportamiento y el uso de motivaciones. En segundo lugar se hace una reseña de XLearn, una herramienta de estudio de algoritmos de Aprendizaje por Reforzamiento, como antecedente del módulo de aprendizaje. En tercer lugar, se exponen los estudios etológicos con monos aulladores que sirven de referencia a los experimentos desarrollados: las investigaciones de Ernesto Rodríguez Luna [Rod94] y de Cristina Domingo Balcells [Dom04]. En éste último se analiza la plasticidad conductual (capacidad de modular la conducta en función de las circunstancias del medio) de un grupo de monos aulladores ante variaciones en la disponibilidad de recursos alimenticios dentro de un espacio restringido, concepto de gran importancia en los experimentos realizados.

El capítulo cuatro presenta el desarrollo del módulo de aprendizaje propuesto en esta investigación. Se presenta la implementación de los diversos algoritmos de Aprendizaje por Reforzamiento (Bandido, Greedy, ε-Greedy, Softmax, LRP, LR, RC e IE, además de dos comparativos no-ApR: óptimo y sin aprendizaje), para plantear el impacto del mundo en las decisiones del agente simulado y, más importante aún, el impacto del módulo de aprendizaje en la Red de Comportamiento. Asimismo, se indican las variantes ambientales en base a la cantidad y distribución de las acciones posibles en el mundo del agente. Lo anterior constituye las modificaciones realizadas a la herramienta ABC, ahora *ABC_ApR*.

En el capítulo cinco se exponen los experimentos con los que se probó ABC_ApR, los cuales giran en torno a la simulación del comportamiento de forrajeo del mono aullador (el *agente*), por lo cual el aprendizaje es concerniente a la conducta de alimentación que es modelada por una Red de Comportamiento. El módulo diseñado modela la toma de decisiones que el animal enfrenta al momento de alimentarse, pues ante varias opciones alimenticias (*acciones posibles*), el mono selecciona un alimento de preferencia que busca como primera opción, como se ha demostrado en estudios etológicos [Dom04]; además, tiene que tomar decisiones concernientes a situaciones en las que la distribución ambiental del alimento complique la rápida ingesta del mismo, o peor aún, cuando el alimento deseado no esté disponible en el ambiente (escasez de comida).

Estos experimentos están sustentados en los modelos etológicos de Rodríguez [Rod94] y Domingo [Dom04], de los cuales se presentan los datos arrojados por la observación de monos reales. Fue necesario el diseño de una Red de Comportamiento que simulara cada uno de los modelos; de estas redes se exponen los resultados obtenidos con sus versiones sin aprendizaje. Sobre estas simulaciones se realizaron los experimentos con aprendizaje, mismos que se ejemplifican brevemente en este capítulo. Posterior a esto, se expone el desempeño de los diversos algoritmos de aprendizaje en los experimentos bajo cada modelo. En el análisis de resultados se consideran diversos criterios, a través de los cuales se analizan ambas simulaciones, y se presentan categorías y resultados globales según el desempeño obtenido.

Finalmente, se presentan las conclusiones derivadas de este trabajo de tesis y se plantea el po-

sible trabajo futuro.

Como apoyo, se cuenta con un apéndice que contiene diversas gráficas y resultados de las corridas, así como los modelos de Red de Comportamiento diseñados para las simulaciones.

Objetivos

Objetivo principal

- Implementar un módulo de Aprendizaje por Reforzamiento que actúe sobre un módulo de comportamiento particular perteneciente a un Mecanismo de Selección de Acción basado en Redes de Comportamiento.

Objetivos secundarios

- Investigar si experimentar con aprender sobre una conducta repercute o no en el patrón de activación de los módulos conductuales de la Red de Comportamiento.
- Optimizar la selección de acciones por parte de un agente mediante técnicas de aprendizaje.
- Analizar el desempeño de los algoritmos de Aprendizaje por Reforzamiento implementados.

Objetivos de los experimentos

- Modelar, mediante aprendizaje, la conducta de forrajeo de los monos aulladores, simulando su toma de decisiones frente a diversas opciones alimenticias.
- Maximizar la ganancia alimenticia del mono simulado, durante toda su vida, considerando que se encuentra en un ambiente desconocido y que debe seleccionar, de entre varias opciones, la mejor.
- Optimizar la alimentación del mono simulado en situaciones donde el alimento de su preferencia esté disponible y en situaciones donde no lo esté.

Capítulo 1

Redes de Comportamiento, un Mecanismo de Selección de Acción

1.1 Agente

Hasta mediados de 1980's, los investigadores de Inteligencia Artificial daban poca importancia a los temas relacionados al concepto de agente, pero posteriormente surgió un gran interés en torno a este tema y actualmente la noción de *agentes* es central a la Inteligencia Artificial. Incluso se ha llegado a definir a la Inteligencia Artificial [RyN95] como el estudio y la construcción de agentes racionales (donde un sistema es racional si hace lo correcto).

Pero hasta el momento, no se ha dicho qué es un agente. A fin de tomar una definición apropiada al presente trabajo, se exponen diversas conceptualizaciones dentro del campo de la Inteligencia Artificial.

Por un lado, existe la noción de agente propuesta por Russell y Norving [RyN95], quienes centran completamente su libro "Inteligencia Artificial: un enfoque moderno" en los agentes racionales, enunciando que un agente es algo que sensa y actúa, o bien, cualquier cosa que percibe su ambiente a través de sus sensores y actúa en él a través de sus efectores. Ya anteriormente, Kaelbling [Kae93] y Mataric [Mat94] definían un agente como un sistema capaz de percibir y actuar en un medio ambiente.

Maes [Mae95] enuncia que un agente autónomo "es un sistema computacional que habita algún ambiente dinámico complejo, sensa y actúa autónomamente en ese ambiente, y al hacerlo realiza un conjunto de metas o tareas para las que está diseñado".

Otros investigadores han optado por proponer ciertas propiedades básicas que un sistema debe tener para ser identificado como agente. Una de las recopilaciones de propiedades más difundida es la de Wooldridge y Jennings [Woo95], en la cual consideran agente a un sistema auto-contenido solucionador de problemas, autónomo, reactivo, pro-activo y con habilidad social, con lo que especifican que el agente debe operar sin intervención directa de los humanos, percibiendo su ambiente y respondiendo a él, siendo capaz de guiar sus acciones hacia sus metas y además, de interactuar con otros agentes.

Franklin y Graesser por su parte, a partir de una comparación de diversas definiciones, proponen que un agente es un sistema situado dentro de un ambiente, siendo a la vez parte de él,

percibiéndolo y actuando en el mismo, influenciando sus percepciones futuras y buscando lograr sus objetivos [Fra96].

Estas definiciones de agente coinciden en varios puntos. El agente 1) está situado en algún ambiente; 2) percibe su ambiente y actúa en él; 3) actúa para alcanzar metas; 4) actúa de manera que sus acciones actuales puedan influenciar sus percepciones posteriores (sus acciones influencian su ambiente), y; 4) actúa continuamente sobre algún periodo de tiempo.

Tales puntos ayudan a la construcción de una definición de agentes: "Un agente es un sistema situado en un ambiente, capaz de percibirlo y de actuar en él de manera continua e influenciándolo a la vez, para lograr sus objetivos".

Russell y Norving [RyN95] presentan el esquema general implicado en un agente (figura 1.1). El agente tiene un rol activo, originando acciones que afectan su ambiente, más que permitir pasivamente que el ambiente le afecte.

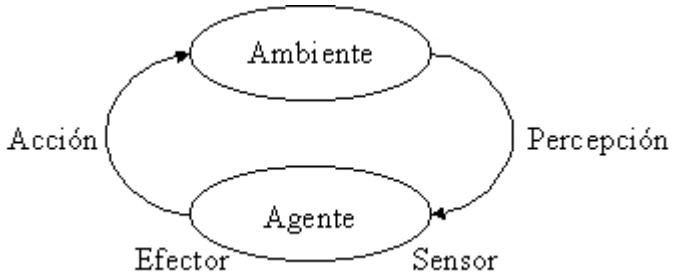

Figura 1.1: Descripción general de un agente

Para diseñar un agente hay que tener idea de las posibles percepciones y acciones, de las metas o medidas de desempeño que debe alcanzar el agente y de la clase de ambiente en el que operará:

Diseño de agente (tipo de agente) = Percepciones + Acciones + Metas + Ambiente

1.1.1. Ambiente

Hasta ahora se ha mencionado que todo agente está inmerso en un ambiente que es capaz de sensar y sobre el que puede actuar, pero no se ha mencionado con precisión lo que es el ambiente. En una forma sencilla, pero concluyente, el ambiente de un agente identifica a *todo aquello que le rodea* y la naturaleza de la conexión entre agente y ambiente siempre es la misma: las acciones son realizadas por el agente en el ambiente, el cual a su vez proporciona percepciones al agente.

No importa si el agente está situado en el mundo real o en un ambiente virtual; lo que interesa en relación a ellos es la secuencia de percepciones generadas por el ambiente, pues es a partir de ellas que el agente puede interactuar con su ambiente. A continuación se presentan diversos tipos de ambientes [RyN95]:

- Accesible / inaccesible. El ambiente es accesible cuando el aparato sensorial del agente da acceso al estado completo del ambiente; en caso contrario es inaccesible.

- Determinístico / no determinístico. El ambiente es determinístico si el próximo estado del ambiente es determinado completamente por el estado actual y las acciones seleccionadas por los agentes; si no es así, el ambiente es no determinístico.

- Episódico / no episódico. En un ambiente episódico, la experiencia del agente es dividida en "episodios", consistentes de la percepción del agente y de su actuación; sus acciones dependen sólo del episodio actual. Si no existen divisiones, es no episódico.

- Estático / dinámico. Si el ambiente puede cambiar mientras un agente está deliberando, entonces se dice que el ambiente es dinámico para el agente; de otra manera es estático. Si el ambiente no cambia con el paso del tiempo pero la calificación del desempeño del agente lo hace, entonces el ambiente es semidinámico.

- Discreto / continuo. Si hay un número limitado de percepciones y acciones distintas, claramente definidas, el ambiente es discreto. En caso contrario, es continuo.

1.1.2. Autonomía y adaptación

Los sistemas basados en el comportamiento, conocidos también como "agentes autónomos" [Mae94], tienen su inspiración en la etología, área de la biología que estudia el comportamiento de los animales en su hábitat natural. La IA basada en el comportamiento enfatiza el modelado y construcción de sistemas que interactúan directamente con el medio ambiente, que es lo que caracteriza a este tipo de sistemas; por lo general, se trata de un medio ambiente dinámico, complejo e impredecible, en el cual el agente busca satisfacer un conjunto de metas o motivaciones, variantes o no en el tiempo.

Maes [Mae94] enuncia los principales conceptos que interesan en este trabajo: "Un *agente* es un sistema que intenta satisfacer un conjunto de metas en un ambiente dinámico y complejo. Un agente está *situado* en el ambiente: puede sensar el mundo a través de sus sensores y actuar sobre él usando sus efectores ... Un agente es llamado *autónomo* si decide por sí mismo cómo relacionar los datos que sensa a instrucciones para sus efectores de forma tal que sus metas sean atendidas acertadamente. Se dice que un agente es *adaptativo* si es capaz de mejorar con el tiempo, esto es, si el agente llega a ser mejor alcanzando sus metas en base a la experiencia."

La autonomía es una característica deseable en los sistemas bajo el enfoque de la IA basada en el comportamiento. De acuerdo a McFarland, la autonomía implica libertad de control externo o, en otras palabras, auto-gobierno [McF95]. La razón por la cual los agentes autónomos no deben ser controlados por un agente externo es que el estado del agente autónomo no es completamente observable. Debe tenerse en cuenta que la autonomía requiere que el agente esté en funcionamiento continuo, repitiendo el proceso percepción-acción permanentemente (*persistencia*).

Por su parte, Russell y Norving [RyN95] mencionan que un sistema es autónomo en la medida en que su comportamiento es determinado por su propia experiencia, sin basarse en conocimiento empotrado al construir el agente (lo que lo haría inflexible), así como que la autonomía conlleva la capacidad de operar en una amplia variedad de ambientes, por adaptabilidad.

El principal problema a ser resuelto en la investigación de los agentes autónomos es el proponer una arquitectura para un agente autónomo que resulte en un agente que demuestre un comportamiento adaptativo, robusto y efectivo. Al decir *adaptativo* se hace referencia a un agente que mejora su habilidad para alcanzar metas en el tiempo. Un agente *robusto* es aquel que nunca colapsa completamente, pues en cambio presenta una degradación graciosa cuando fallan algunos componentes o suceden situaciones inesperadas. Se denomina como *efectivo* al agente que tiene éxito al alcanzar sus metas.

Existen dos subproblemas que se han intentado resolver: el problema de la selección de acción y el problema de aprender de la experiencia. Muchas de las arquitecturas de agentes propuestas sólo se han enfocado a uno de los dos subproblemas: ya sea que se combine una selección de acción simplista con un aprendizaje sofisticado o bien, se implemente una selección de acción sofisticada sin ninguna clase de aprendizaje. El presente trabajo estudia ambos problemas en un agente que debe decidir qué hacer mediante un mecanismo de selección de acción y aprendizaje (del tipo reforzado). A continuación se expondrán los dos subproblemas mencionados.

1.1.3. La selección de acción

En los sistemas basados en el comportamiento, el problema de la selección de acción radica en el proceso que un agente debe llevar al cabo en cada instante de tiempo a fin de satisfacer las múltiples metas que posee. Tal proceso consiste de seleccionar de entre un conjunto de posibles acciones a ejecutar, aquella acción que más se adecue a lo que se percibe en ese momento, para generar y mantener un comportamiento correcto y robusto del agente. Este problema puede plantearse con las siguientes interrogantes: ¿Cómo puede un agente decidir qué debe hacer inmediatamente para fomentar el progreso hacia sus múltiples metas variantes en el tiempo? ¿Cómo puede tratar con contingencias u oportunidades que puedan surgir? ¿Cómo puede arbitrar entre metas conflictivas? ¿Cómo puede reaccionar de manera oportuna?

En teoría es posible que, para un agente que tiene un conjunto fijo de metas y que vive en un ambiente determinístico o probabilístico, se pueda calcular la política óptima de selección de acción. Pero cuando se trata de agentes que se encuentran en ambientes más reales, es imposible determinar una política óptima, dado que se enfrentan diversos problemas que aumentan la dificultad para modelar, pues en principio se enfrenta un ambiente dinámico, no determinístico o no probabilístico y se tienen metas variantes en el tiempo, por lo cual se cuenta con información que posiblemente sea incompleta o incluso incorrecta. Además, no debe olvidarse el problema de la limitación de recursos, ya sea en memoria, computación o tiempo.

Los investigadores en el campo de los agentes autónomos están interesados en encontrar mecanismos de selección de acción que sean robustos y adaptativos, así como también en que los agentes alcancen sus metas dentro de los requerimientos y restricciones impuestos por el medio ambiente y las tareas a realizar. El mecanismo de selección de acción "ideal" deberá cubrir los siguientes requisitos [Mae94]:

- Favorecimiento de acciones que contribuyan a alcanzar metas, particularmente aquellas que representen un mayor progreso hacia las metas.
- Flexibilidad ante contingencias y oportunidades.

- Operación en tiempo real.
- Minimización de alternancias innecesarias entre acciones que contribuyan a metas diferentes.
- Mejoramiento en base a la experiencia.
- Mostrar una degradación graciosa cuando fallen los componentes o bien, sucedan cambios inesperados.
- No caer en ciclos o en situaciones de interbloqueos, o bien, en situaciones en que el agente demande metas inalcanzables.
- Ser bastante bueno para el ambiente y las tareas disponibles. Es decir, mientras el agente se dirija a alcanzar sus metas dentro de las restricciones impuestas por la situación en que se encuentre, la forma en que resuelve el problema es considerada como aceptable.

Maes [Mae94] presenta una clasificación de los modelos para la selección de acción que se han propuesto, dividiéndolos en tres clases:

(a) Redes heterárquicas, construidas a mano. Arquitecturas que requieren que el diseñador del agente resuelva el problema de la selección de acción desde el inicio para cada agente que se construya, es decir, el diseñador debe analizar cuidadosamente el ambiente y las tareas a manipular, para que proceda a diseñar un conjunto de módulos reflejos y la manera de combinar las salidas de esos módulos. Las metas en esta clase de agentes están implícitas, pudiendo ser múltiples y de naturaleza muy diversa. El mecanismo para solucionar conflictos (determinar qué módulos dirigirán a los efectores) está dado ya sea por un conjunto de alambrados de supresión/inhibición o bien, por un circuito lógico, mediante lo cual se asegura que a lo más un módulo controlará a un efector en todo momento. Ejemplos: la Arquitectura de Subsumción [Bro86] y la versión minimalista de la misma [Con90].

(b) Redes heterárquicas, compiladas. Arquitecturas que intentan facilitar la construcción de los agentes mediante la automatización del proceso de diseñar la solución de conflictos entre los módulos de competencia. Se requiere que el diseñador especifique en un determinado formalismo cuáles son las metas del agente, cómo pueden devenir unas metas en otras acciones o metas y cuáles son los diversos módulos/acción y sus condiciones y efectos esperados. Ejemplos de esta clase de arquitecturas lo constituyen las Redes de Comportamiento [Mae89, Mae90a] y el sistema Rex/Gapps [KyR90]

(c) Redes jerárquicas, construidas a mano. Propone organizar las acciones en una jerarquía que comprende desde "modos" o actividades de alto nivel a través de acciones compuestas de nivel medio hasta acciones primitivas detalladas. En realidad, sólo las acciones primitivas son ejecutables. Estos sistemas usan una clase de ordenamiento para la selección de acción en niveles de abstracción más altos para sesgar la selección de acciones más primitivas. Ejemplos de tales arquitecturas son Hamsterdam [Blu94] y el trabajo de Tyrell [Tyr93].

En la sección dos del presente capítulo se analizan algunos de los mecanismos de selección de acción que han sido desarrollados con el fin de solucionar el problema de la selección de acción.

1.1.4. Aprender de la experiencia

El problema del aprendizaje a partir de la experiencia puede definirse así: Sea un agente con (a) un conjunto de acciones o módulos de competencia, (b) cierta información sensada y (c) múltiples metas -variantes en el tiempo-, ¿cómo puede tal agente mejorar su desempeño basándose en su experiencia? ¿Cómo puede decidir cuándo "explotar" su actual mejor acción, versus la "exploración" de otras acciones para posiblemente descubrir formas de lograr sus metas? ¿Cómo puede incorporar la realimentación del mundo en las estructuras internas que generan su comportamiento? Por mejoría debe entenderse que el agente llegue a ser más exitoso en la satisfacción de sus metas o necesidades. Así, si se tiene una meta final, entonces el tiempo o el número de acciones promedio requerido (u otra medida de costo) para lograr la meta, decrementa con el tiempo. Si en cambio, se tiene una meta por maximización de reforzamiento, entonces el promedio del reforzamiento positivo recibido durante un cierto intervalo de tiempo incrementará con la experiencia.

La mayoría de las arquitecturas descuidan el problema de aprender por experiencia (excepto [Mae91b]), lo cual se observa porque sus agentes sólo son adaptativos en un sentido muy restringido, dado que son capaces de tratar con situaciones inesperadas, pero no aprenden por realimentación de su ambiente. No llegan a mejorar el logro de sus metas por medio de la experiencia.

Aprender a partir de la experiencia es una necesidad para cualquier agente que deba demostrar un comportamiento autónomo y robusto sobre largos periodos de tiempo. Dado que el aprendizaje por experiencia representa una fuente para mejorar el comportamiento, guarda estrecha relación con la capacidad de adaptación de los agentes. Es importante tener en cuenta que el verdadero comportamiento adaptativo es un proceso continuo, inherentemente dinámico.

Cualquier modelo por aprendizaje en agentes autónomos debe cubrir los siguientes puntos:

(a) El aprendizaje debe ser incremental.

(b) El aprendizaje debe estar sesgado a la adquisición de conocimiento relevante a las metas (no puede permitirse que el agente aprenda cualquier hecho que pueda ser aprendido).

(c) Debe ser capaz de enfrentar ambientes probabilísticos y ruidosos, sensores con fallas, etc.

(d) El aprendizaje debe ser no supervisado.

(e) De preferencia, que el modelo de aprendizaje proporcione al agente algún conocimiento inicial propio del sistema.

En el capítulo 2 se trata más a fondo el tema del aprendizaje, dado que éste es el tema central del presente trabajo.

1.2 Mecanismos de selección de acción

Como ya se mencionó, el dilema al que se enfrenta un agente al tener que decidir qué acción debe seleccionar en cada momento es conocido como el problema de la selección de acción. La especificación de la manera en que el agente debe seleccionar las acciones constituye lo que se denomina Mecanismo de Selección de Acción, el cual es un modelo computacional para el diseño

de agentes, cuya implementación producirá la elección de la acción más apropiada cuando se le proporciona al sistema una entrada conformada de diversos estímulos internos y/o externos. Los estímulos externos son percepciones del estado del mundo externo del agente. Los estímulos internos son percepciones del estado del mundo interno de la entidad (hambre, cansancio, etc). Las acciones son conductas externas ejecutadas por el agente en respuesta a la presencia de determinados estímulos internos y/o externos. Sólo una acción puede ser ejecutada en un tiempo específico. Un mecanismo de selección de acción combina los estímulos externos e internos y selecciona de entre todas las acciones potenciales que la entidad puede ejecutar en ese momento, aquella que resulte ser la más adecuada.

De forma breve, se puede decir que el problema de la selección de acción especifica el *qué*, en tanto que el mecanismo de selección de acción indica el *cómo*. A continuación se hace un breve análisis de los mecanismos de selección de acción más importantes dentro de los sistemas basados en el comportamiento.

1.2.1. Arquitectura de Subsumción

Arquitectura computacional propuesta por R. Brooks [Bro86], que surge ante la necesidad de controlar robots, con el propósito de programar agentes incorporados, situados e inteligentes. La arquitectura de subsumción (*SA*, por sus siglas en inglés) consiste de una serie de comportamientos, mismos que contituyen una red de máquinas de estados finitos fuertemente alambrada. La selección de acción consiste de comportamientos de niveles superiores dominando (o subsumiendo) la salida de comportamientos inferiores.

En esta arquitectura, Brooks define *niveles de competencia*, que son una especificación informal de una clase deseada de comportamiento en diversos niveles de abstracción (por ejemplo, evitar obstáculos, explorar el mundo, etc.). Un nivel de competencia más alto implica una clase deseada de comportamiento más específica (ver ejemplo en la figura 1.2). Brooks argumenta que adhiriéndose a los principios de la arquitectura de subsumción, se puede construir un sistema incrementalmente.

Cada nivel de competencia en la arquitectura de subsumción es implementado usando una máquina de estados finitos aumentada con elementos temporizadores y registros. En la arquitectura de subsumción, los módulos/comportamientos no pueden cooperar -cada capa es construida sobre su propio hardware totalmente desde el principio. El sistema de control está fuertemente alambrado directamente sobre la estructura de los comportamientos y sus interconexiones, por lo que no puede ser alterado sin rediseñar el sistema.

La funcionalidad de un robot basado en SA es vista como una propiedad *emergente* de la interacción de sus conductas y depende de las propiedades estáticas y dinámicas del ambiente.

La SA constituye un sustrato computacional que puede ser usado como medio para instanciar diferentes mecanismos de selección de acción, o como un medio de instanciamiento de diferentes técnicas para control motriz.

La SA no es ni una descripción computacional del problema de la selección de acción, ni una

descripción algorítmica de cómo seleccionar acciones, sino más bien es una técnica de implementación, la cual puede usarse para implementar un gran número de diferentes algoritmos. La arquitectura de subsumción no contiene una descripción de cómo tratar con los aspectos importantes del problema de selección de acción. En conclusión, es una técnica de implementación, más que un algoritmo o un mecanismo para la selección de acción.

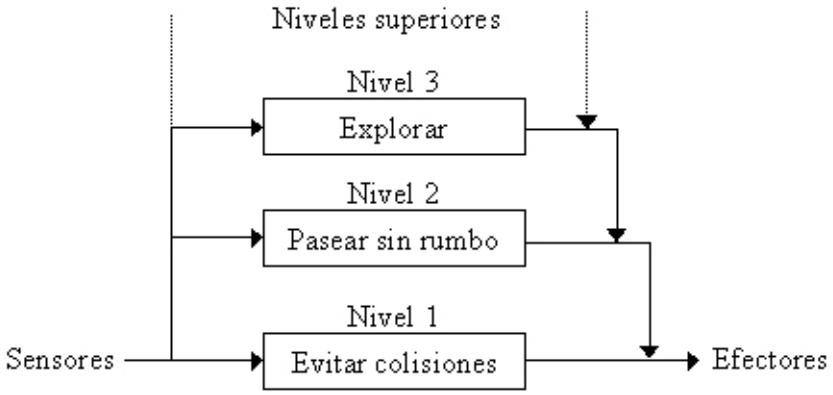

Figura 1.2: Arquitectura de Subsumción con una capa de niveles de competencia para un robot móvil

1.2.2. Redes de Comportamiento

Modelo propuesto por P. Maes [Mae89, Mae90a, Mae90b, Mae91a]. El sistema consiste de un conjunto de comportamientos o módulos de competencia que se encuentran conectados para formar una red. Este enfoque está inspirado en la Sociedad de la Mente de Minsky [Min86][1], y en realidad implementa una clase de análisis de medios-fines, donde un conjunto de comportamientos reduce la diferencia entre el estado actual y el estado meta del sistema.

En la red, cada comportamiento m_i es presentado como una tupla $(c_i, a_i, d_i, \alpha_i)$, describiendo: *1)* las precondiciones c_i bajo las cuales es ejecutable (esto es, puede ser aplicado), *2)* los efectos posteriores a la ejecución exitosa en forma de una lista añadir a_i y una lista borrar d_i, y *3)* el nivel de activación, α_i, el cual es una medida de la relevancia de la conducta. Cuando el nivel de activación de una conducta ejecutable excede un umbral específico, es seleccionada para habilitar su acción. Cuando una conducta es seleccionada ejecutará la acción más apropiada desde su punto de vista. El nivel de activación de los comportamientos está influenciado por la inyección y remoción externas de "energía de activación" de la red así como también por un intercambio interno de energía de activación entre los módulos dentro de la red. El intercambio de energía de activación toma lugar continuamente y en cada ciclo es seleccionada la conducta ejecutable con el nivel de activación más alto.

[1]Esta teoría sugiere la construcción de un sistema inteligente como una sociedad de agentes interactuantes, cada uno con su propia competencia específica. La idea es que los agentes cooperen en tal forma que la sociedad como un todo funcione apropiadamente. Maes plantea sus Redes de Comportamiento como sociedades de agentes, donde un agente es un módulo de competencia.

Mediante el intercambio de energía de activación, los comportamientos compiten y cooperan para seleccionar una acción (un comportamiento que ejecuta una acción) que es la más apropiada al estado actual del ambiente y a las metas del sistema. La forma en que la energía de activación es intercambiada entre las conductas favorece la selección de una secuencia de comportamientos que pueden transformar el estado actual del ambiente a un estado meta. Esta secuencia, sin embargo, no es determinada definitivamente y seguida después como en los planificadores de la IA tradicional. A diferencia de los planificadores tradicionales, este sistema no tiene una representación explícita de planes, ni implementa un algoritmo de búsqueda específico para construir planes. Más bien el sistema selecciona y ejecuta el próximo paso de la secuencia emergentemente, lo cual es influido no sólo por las metas del sistema sino también por el estado del ambiente en cada paso. Las acciones seleccionadas corresponden a las submetas del sistema, mismas que llevan al sistema a estar próximo a una de sus metas.

La idea central del modelo de Maes para la selección de acción es que diferentes tipos de ligas codifican varias relaciones (por ejemplo, relaciones consumatorias/apetitivas entre nodos, relaciones conflictivas entre nodos, relaciones para alcanzar metas entre nodos y metas, relaciones que contrarrestan metas entre nodos y metas y relaciones dependientes de la situación entre los sensores ambientales y los nodos), y con esta información fuertemente alambrada en el mecanismo, la excitación sólo puede alimentarse a partir de la situación y las metas actuales, y la actividad, después de varias iteraciones, vendrá a asentarse en el nodo que represente el comportamiento más apropiado. La figura 1.3 muestra un ejemplo de una Red de Comportamiento.

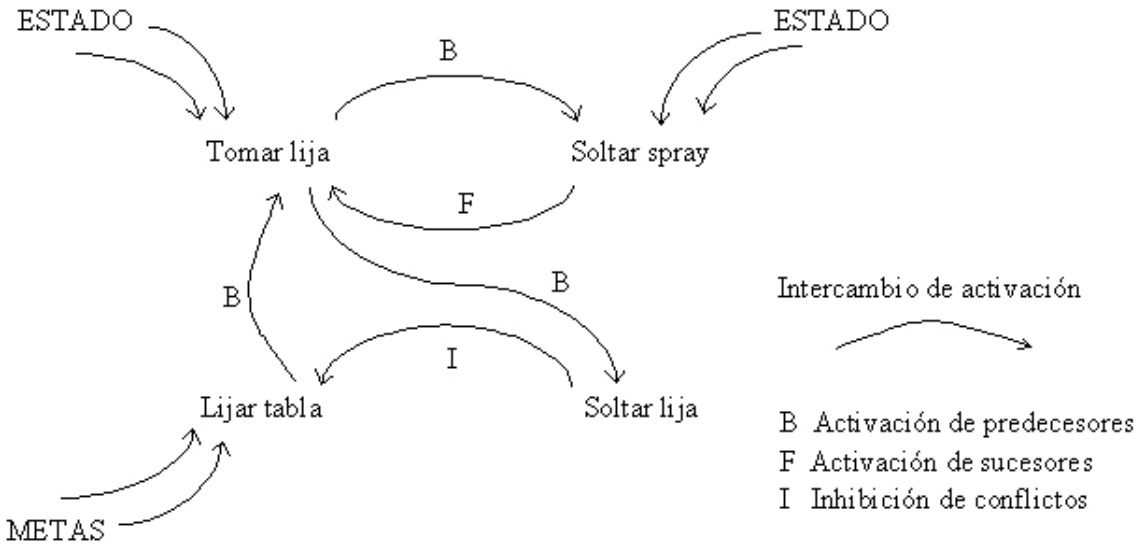

Figura 1.3: Ejemplo de una Red de Comportamiento que simula un robot con dos manos que tiene que lijar una tabla y pintarse a sí mismo (adaptado de [Mae90b]

El sistema está diseñado para manejar situaciones lo mismo que está orientado a metas, así el sistema será oportunístico y elegirá la secuencia más corta de acciones que puede llevar al sistema a un estado meta. Debido a su naturaleza distribuida, el sistema es tolerante a fallos, por lo que

el fracaso de un simple módulo de comportamiento no necesariamente conduce a la falla del sistema completo.

1.2.3. Hamsterdam

Modelo computacional de selección de acción [Blu94] inspirado en observaciones (basado en el modelo de Ludlow [Lud76]), en el cual se implementa un sistema del tipo "el ganador se lo lleva todo", consistente con el pensamiento etológico tradicional, pero permitiendo pérdida de actividad para expresar preferencias en forma de recomendaciones. Este enfoque se centra en la importancia de las organizaciones jerárquicas del comportamiento aunado a compartir información y contar con un flujo común con el cual expresar necesidades internas y oportunidades externas.

Las actividades son organizadas en jerarquías traslapadas no conexas, con las actividades más generales en la parte superior y las actividades más especificas en las hojas. Las actividades se representan como nodos en un árbol, pudiendo cada nodo tener 0 o más hijos. El mecanismo de selección de acción consiste en determinar el nodo hoja a activar en un momento dado, comenzando desde la raíz del árbol. Los hijos se inhiben mutuamente y sólo uno puede estar activo a la vez. Si se activa un nodo hoja, éste podrá ejecutar una o más acciones; en caso contrario, los hijos del nodo activado compiten por el control hasta alcanzar un nodo hoja (ver figura 1.4). Las acciones compiten en base a su valor, el cual resulta del monitoreo de variables endógenas y factores externos. Dentro de una colección de actividades mutuamente inhibidas, el sistema itera hasta hallar una solución estable en la cual una actividad tenga un valor positivo y el valor de las restantes actividades esté dentro de ciertos rangos.

Las jerarquías de actividad son construidas a mano, y los parámetros ajustados en base a comportamientos observados. Con su modelo, Blumberg demostró que es necesario incorporar mecanismos tomados del conocimiento etológico a fin de lograr que los animats se comporten de manera similar a los animales (un *animat* es un animal artificial [Wil85], ya sea simulado o incorporado en un robot, que debe sobrevivir y adaptarse en ambientes progresivamente cambiantes).

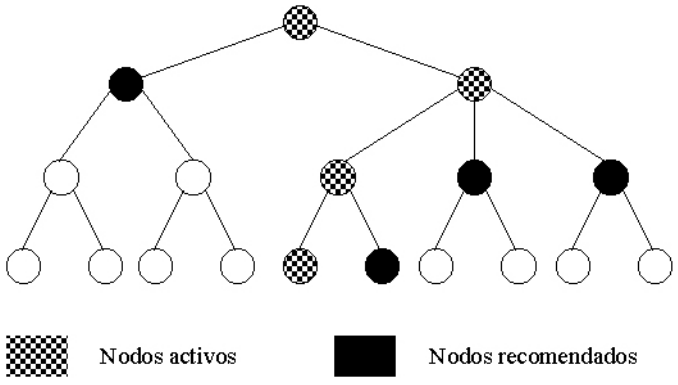

Figura 1.4: Selección de acción en Hamsterdam - Jerarquía de comportamientos.

1.3 Redes de Comportamiento

ABC [Gue97c], [Mon98], la herramienta para el diseño de agentes sobre la cual se desarrolla el presente trabajo, consta de una implementación del mecanismo de selección de acción denominado Redes de Comportamiento, propuesto por Maes [Mae89, Mae90a]. Por tal motivo, la presente sección está dedicada a presentar de manera amplia dicho modelo.

Un agente autónomo es visto como un conjunto de módulos de competencia (nodos de la red). En el enfoque de las Redes de Comportamiento, la selección de acción es modelada como una propiedad emergente de las dinámicas de activación/inhibición entre los módulos de competencia. La red de conductas es en sí una red distribuida, recurrente y no jerárquica. El conjunto de nodos que lo conforman representa las conductas; estos nodos pueden ser consumatorios (los que desencadenan una acción) o apetitivos (los que al dispararse, establecen condiciones para que otro, ya sea consumatorio o apetitivo, pueda dispararse). Existen dos flujos de entrada a la red, ya sea por los sensores del ambiente (estímulos externos) o por las motivaciones (generalmente derivadas de estímulos internos). Las motivaciones constituyen la energía interna potencial, ya que tiende a acumularse en el tiempo y se descarga por la ejecución de una conducta consumatoria; deben conectarse únicamente a nodos consumatorios. Los sensores del ambiente son evaluados binariamente (V o F), en tanto que las motivaciones son dadas en cantidades reales. La red exhibe un comportamiento guiado por metas.

1.3.1. Algoritmo

Formalmente, la Red de Comportamiento está constituida por:

- **Módulos de competencia**

  Un módulo de competencia m_i se describe mediante la tupla: $m_i = (c_i, a_i, d_i, \alpha_i)$

  La cual se conforma por los siguientes componentes:

c_i	lista de precondiciones.	a_i	lista de agregar.
d_i	lista de eliminar.	α_i	nivel de activación.

  La lista de precondiciones debe cubrirse en su totalidad al momento de activar al agente. Las listas de agregar y eliminar se utilizan para capturar los efectos esperados de la acción del agente. En base a esta definición, un agente es *ejecutable* en el tiempo t cuando todas sus precondiciones son verdaderas en t, con lo cual tal agente puede ser activado, lo que significa que podrá ejecutar algunas acciones sobre su ambiente.

- **Ligas**

  Los agentes están enlazados en una red por medio de ligas sucesoras ($\rightarrow$), predecesoras ($\leftarrow$) y conflictivas ($\nrightarrow$). Las ligas son usadas para propagar la activación entre los módulos y es precisamente mediante su acción sobre un agente (en sus 3 listas -precondiciones, añadir y eliminar), que se logra definir completamente la red.

  Sean dos módulos de competencia, $m_x = (c_x, a_x, d_x, \alpha_x)$ y $m_y = (c_y, a_y, d_y, \alpha_y)$, las ligas que entre ellos se establecen pueden ser de 3 tipos:

	Existe	para toda proposición
Liga sucesora de m_x a m_y	$m_x \rightarrow m_y$	$p \in a_x \cap c_y$
Liga predecesora de m_x a m_y	$m_x \leftarrow m_y$	$p \in c_x \cap a_y$
Liga conflictiva de m_x a m_y	$m_x \nleftrightarrow m_y$	$p \in c_x \cap d_y$

◇ **Activación**

Los módulos usan estas ligas para activar e inhibir a los demás módulos. A esto se le conoce como propagación de activación. Al transcurrir el tiempo, la energía de activación se acumula en los módulos que representan las mejores acciones a tomar, dadas la situación y las metas actuales. El nivel de activación de cada uno de esos nodos se compara contra un umbral (analizado más adelante), y cuando alguno de ellos logra sobrepasarlo y además se comprueba que tal módulo es ejecutable, entonces se activa ese módulo para que ejecute sus acciones en el mundo.

A cada momento, la propagación de la activación junto con la entrada de nueva energía de activación a la red son determinadas por el estado del ambiente y las metas globales del agente. De ahí que existan tres tipos de activaciones:

Activación por estado. Energía proveniente del ambiente hacia los módulos que aparean parcialmente con el estado actual.

Activación por metas. Incremento de la activación de los módulos que alcanzan metas globales. Existen dos tipos de metas: (a) temporales, a alcanzar una sola vez, y (b) permanentes, a alcanzar continuamente.

Inhibición de metas protegidas. Disminución de la activación por las metas globales que el agente ha alcanzado y que deben protegerse. Dichas metas protegidas remueven parte de la activación de aquellos módulos que podrían anularlas.

Además de la influencia del ambiente y de las metas, los módulos se activan e inhiben comúnmente entre ellos a través de sus ligas, por lo cual existen otras tres formas de propagación de activación:

Activación de sucesores Un módulo de competencia $m_x = (c_x, a_x, d_x, \alpha_x)$ ejecutable propaga activación hacia adelante, al incrementar el nivel de activación de aquellos sucesores m_y para los cuales la proposición compartida $p \in a_x \cap c_y$ es falsa.

Activación de predecesores Un módulo de competencia $m_x = (c_x, a_x, d_x, \alpha_x)$ no ejecutable propaga activación hacia atrás, al incrementar el nivel de activación de aquellos predecesores m_y para los cuales la proposición compartida $p \in c_x \cap a_y$ es falsa.

Inhibición de conflictos Un módulo de competencia $m_x = (c_x, a_x, d_x, \alpha_x)$ ejecutable o no, decrementa el nivel de activación de los módulos confictivos m_y para los cuales la proposición compartida $p \in c_x \cap d_y$ es verdadera, excepto para aquellas ligas para las cuales existe una liga conflictiva inversa que es más fuerte.

◇ **Parámetros**

La red requiere el uso de cinco parámetros, a fin de ajustar la dinámica de activación y, por ende, la selección de acción del agente.

θ Umbral de activación. Se reduce 10 % cada vez que ningún módulo puede ser seleccionado. Se restablece a su valor inicial cuando se ha seleccionado un módulo.

π Nivel promedio de activación.

ϕ Cantidad de energía de activación que inyecta en la red una proposición que se observa como verdadera.

γ Cantidad de energía de activación que una meta inyecta en la red.

δ Cantidad de energía de activación que una meta protegida sustrae de la red.

Como ya se mencionó, estos parámetros determinan la cantidad de activación que propagan los módulos, la cual puede darse en alguna de las siguientes formas:

- Para una proposición falsa en su lista de precondiciones, un módulo no ejecutable propaga α a sus predecesores.
- Para una proposición falsa en su lista de agregar, un módulo ejecutable propaga $\alpha(\phi/\gamma)$ a sus sucesores.
- Para una proposición verdadera en su lista de precondiciones un módulo decrementa $\alpha(\delta/\gamma)$ de los módulos conflictivos.

Algoritmo de las Redes de Comportamiento

El algoritmo global de la Red de Comportamiento realiza un ciclo infinito que abarca a todos los módulos de competencia:

Algoritmo 1 Algoritmo de las Redes de Comportamiento

1: Calcular el impacto del estado, las metas y las metas protegidas sobre el nivel de activación de un módulo.
2: Calcular la activación e inhibición que un módulo ejerce a través de sus ligas sucesoras, predecesoras y conflictivas sobre cada uno de los módulos relacionados.
3: Aplicar función de decaimiento, para asegurar que el nivel de activación global permanezca constante.
4: Activar el módulo que cumpla las siguientes tres condiciones:
(a) Ser ejecutable.
(b) Tener un nivel de activación que sobrepase al umbral.
(c) Tener el nivel de activación más alto entre todos los módulos que cubran los dos puntos anteriores.
Cuando 2 módulos de competencia satisfacen las tres condiciones (son igualmente fuertes), uno de ellos es seleccionado aleatoriamente. El nivel de activación del módulo activado se reinicializa a 0. Si ninguno de los módulos cumple las condiciones (a) y (b), el umbral es reducido en un 10 %.

1.3.2. Modelo matemático

A continuación se presenta la descripción matemática del algoritmo [Mae89].

◇ **Agente**

Como ya se estableció anteriormente (sección 1.3.1), un módulo de competencia m_i se

define como una tupla $m_i = (c_i, a_i, d_i, \alpha_i)$, siendo c_i, a_i y d_i los conjuntos de proposiciones de precondición, agregar y eliminar, respectivamente, y α el nivel de activación de m_i.

Un agente A se define como el conjunto de todos los módulos de competencia: $A = \{m_1, m_2, ..., m_n\}$.

◇ **Proposiciones**

Sea $p = (c_i \cup a_i \cup d_i)$ el conjunto de proposiciones que definen el módulo de competencia m_i.

Sea P el conjunto de precondiciones usadas para definir al agente A: $P = \{p_1 \cup p_2 \cup ... \cup p_n\}$.

Dado el agente A, sea:

$M(p) = \{m_i : m_i \in A \ \wedge \ p \in c_i, 1 \leq i \leq n\}$ el conjunto de todos los módulos de competencia m_i en A que tienen a la proposición p como precondición.

$A(p) = \{m_i : m_i \in A \wedge p \in a_i, 1 \leq i \leq n\}$ el conjunto de todos los módulos de competencia m_i en A que alcanzan la proposición p.

$D(p) = \{m_i : m_i \in A \wedge p \in d_i, 1 \leq i \leq n\}$ el conjunto de todos los módulos de competencia m_i en A que anulan la proposición p.

◇ **Percepción**

La función $S(t) = \{p : p \in P_A \ \wedge \ p$ es verdadera al tiempo $t\}$ retorna el conjunto de proposiciones que se observan verdaderas en el ambiente al tiempo t (el estado del ambiente como es percibido por el agente).

◇ **Metas globales del agente**

La función $G(t) = \{p : p \in P_A \ \wedge \ p$ es una meta del agente al tiempo $t\}$ regresa el conjunto de proposiciones que son metas del agente A en el tiempo t.

◇ **Metas globales alcanzadas**

La funcion $R(t) = \{S(t) \cap G(t)\}$ regresa el conjunto de metas del agente A que ya han sido alcanzadas en el tiempo t.

◇ **Módulos de competencia ejecutables**

La función $E(m_i, t)$ regresa 1 si el módulo de competencia m_i es ejecutable al tiempo t (esto es, si todas sus precondiciones son miembros de S(t)), y 0 en caso contrario.

◇ **Dinámica de Activación/Inhibición**

Activación por influencia del medio ambiente

La cantidad de energía que recibe un módulo m_i del medio ambiente, por correspondencia parcial con éste, es:

$$entrada\_por\_estado(m_i, t) = \sum_{\forall p \in P} \phi \frac{1}{|M(p)|} \frac{1}{|c_i|}$$

Activación por influencia de las metas

La cantidad de energía que recibe un módulo m_i, debido a su capacidad de realizar alguna meta, es:

$$entrada\_por\_metas(m_i, t) = \sum_{\forall p \in P} \gamma \frac{1}{|A(p)|} \frac{1}{|a_i|}$$

Inhibición por metas realizadas

La cantidad de energía que recibe un módulo m_i cuya acción puede afectar una meta previamente realizada es:

$$salida\_por\_metas\_realizadas(m_i, t) = \sum_{\forall p \in P} \delta \frac{1}{|D(p)|} \frac{1}{|d_i|}$$

Propagación hacia atrás

La cantidad de energía que un módulo m_i recibe de un módulo m_j a través de sus ligas de sucesor es:

$$factor\_bwp(m_i, m_j, t) = \begin{cases} \sum_{\forall p \in P} \alpha_j(t-1) \frac{1}{|A(p)|} \frac{1}{|a_i|} & \text{Si } E(m_i, t) = 0 \\ 0 & \text{Si } E(m_i, t) = 1 \end{cases}$$

donde $P = \{p : p \in a_i \cap c_j \wedge p \in S(t)\}$

La cantidad de energía en que un módulo m_i recibe propagación hacia atrás es:

$$bwp(m_i, t) = \sum_{\forall m \in A} factor\_bwp(m_i, m, t)$$

Propagación hacia adelante

La cantidad de energía que un módulo m_i recibe de un módulo m_j a través de sus ligas de predecesor es:

$$factor\_fwp(m_i, m_j, t) = \begin{cases} \sum_{\forall p \in P} \alpha_j(t-1) \frac{\sigma}{\gamma} \frac{1}{|A(p)|} \frac{1}{|a_i|} & \text{Si } E(m_i, t) = 1 \\ 0 & \text{Si } E(m_i, t) = 0 \end{cases}$$

donde $P = \{p : p \in c_i \cap a_j \wedge p \notin S(t)\}$

La cantidad de energía en que un módulo m_i recibe propagación hacia adelante es:

$$fwp(m_i, t) = \sum_{\forall m \in A} factor\_fwp(m_i, m, t)$$

Inhibición por conflicto

La cantidad de energía que es retirada de un módulo m_i por inhibición debida a conflictos

es:

$$factor\_confp(m_i, m_j, t) = \begin{cases} 0 \quad \text{Si } \alpha_i(t-1) > \alpha_j(t-1) \wedge (\exists p)(p \in S(t) \cap c_i \cap d_j) \\ max\left[\sum_{\forall p \in P} \alpha_j(t-1) \frac{1}{|D(p)|} \frac{1}{|d_i|}, \alpha_i(t-1)\right] \\ \quad \text{En cualquier otro caso} \end{cases}$$

donde $P = \{p : p \in d_i \cap c_j \wedge p \in S(t)\}$

La cantidad de energía que es retirada de un módulo m_i por inhibición debida a conflictos es:

$$confp(m_i, t) = \sum_{\forall m \in A \wedge m \neq m} factor\_confp(m_i, m, t)$$

◇ **Nivel de activación**

El nivel de activación del módulo m_i al tiempo t se define como:

$$\alpha(m_i, t) = \begin{cases} 0 & \text{Si } \ t = 0 \\ normaliza & \left[\begin{array}{ll} \alpha(m_i, t-1) & + \\ entrada\_por\_estado(m_i, t) & + \\ entrada\_por\_metas(m_i, t) & - \\ salida\_por\_metas\_realizadas(m_i, t) & + \\ fwp(m_i, t) + bwp(m_i, t) - confp(m_i, t) & \end{array}\right] \text{Si } \ t > 0 \end{cases}$$

donde *normaliza* es una función que se encarga de que la activación total del agente A_n permanezca constante en πn (por analogía con la "energía de activación").

◇ **Módulo de habilidad activo**

El módulo de competencia m_i que se activa al tiempo t está determinado por:

$$activo(t, m_i) = \begin{cases} 1 \quad \text{Si } \ \left\{\begin{array}{ll} \alpha(m_i, t) > \theta & \wedge \\ E(m_i, t) & \wedge \\ random(\{m_i | \alpha(m_i, t) \geq \alpha(m_{j \neq i}, t)\}) & \end{array}\right. \\ 0 \quad \text{En cualquier otro caso} \end{cases}$$

1.3.3. Limitaciones y críticas

La arquitectura de Redes de Comportamiento presenta unas limitaciones, algunas de las cuales son expuestas por Maes en sus artículos [Mae90a, Mae90b, Mae91a]: 1) es necesaria una calibración cuidadosa de los parámetros, ya que son dependientes del problema (la calibración es manual); 2) debido a que la red no mantiene registro de sus acciones, puede caer en ciclos de selección de acciones ([Mon98] introduce la dinámica del mundo para evitar los ciclos), y; 3) no usa retroalimentación del ambiente para mejorar la Red de Comportamiento (en [Mae91b] propone adaptar las ligas de los comportamientos en base a la experiencia).

Una de los principales críticos de las Redes de Comportamiento es Tyrrell, quien en su tesis

doctoral [Tyr93] realizó un análisis de varios mecanismos de selección de acción; él reconoce que el mecanismo de Maes es una mejoría a los enfoques previos porque basa sus cálculos tanto en motivaciones como en estímulos internos, abordando satisfactoriamente muchos aspectos del problema de selección de acción, pero también señala que algunas deficiencias en el diseño de las Redes de Comportamiento significan que éstas no se desempeñarán bien sobre problemas de selección de acción similares a los de los animales.

Uno de los puntos que observa en contra del mecanismo es que los sensores del ambiente están restringidos a tomar valores binarios (al considerar el superar un umbral), lo que hace que se pierda información, dado que muchas propiedades de los ambientes reales son continuas.

Además, Tyrrell [Tyr93, Tyr94] notó que hay perjuicio contra los módulos que reciben entrada de los sensores que también dan activación a otros módulos; esto porque Maes especifica que la activación que una condición verdadera tiene disponible para propagar a la red debiera ser dividida equitativamente entre todos los módulos que abastece.

Además de estas observaciones, se han señalado otras más [Nor94] con respecto al tamaño de la Red de Comportamiento, alegando que no está claro cómo se podría expandir el sistema a un conjunto de tareas más complejas de las que usó Maes, sin que el algoritmo llegue a ser computacionalmente costoso hasta el punto de volverse impráctico, pues la complejidad del algoritmo y el tiempo de reacción se incrementan con el número de comportamientos, así como el número de ligas entre ellos. Asimismo se acusa que la Red de Comportamiento es una representación estática de los métodos que un agente puede elegir para satisfacer un conjunto de posibles metas.

La propia Maes, en [Mae94], documento donde aborda esta arquitectura por última vez, concluye que la clase de agentes que pueden ser construidos está restringida, además de que en ocasiones es difícil especificar las metas y el comportamiento deseado de un agente, lo cuál es necesario hacer bien para tener una buena selección de acción.

Pero aún con las críticas que se le han hecho, no todo está en contra de las Redes de Comportamiento, pues han sido muy estudiadas y son uno de los mecanismos de selección de acción más conocidos y utilizados, sirviendo de inspiración a otros trabajos; incluso se han desarrollado propuestas para solucionar sus limitaciones (por ejemplo [Goe97] y [Sin02]).

Capítulo 2

Aprendizaje

2.1 ¿Qué es aprendizaje?

El aprendizaje es considerado una parte esencial de toda forma de inteligencia. En el ámbito computacional es uno de los aspectos más discutidos y deseados, pues no basta con hacer que las computadoras se comporten de acuerdo a lo que un programador les indica, sino que es deseable que por ellas mismas adquieran conocimiento en base a lo que experimentan, de manera similar a como lo hacen los humanos y los animales, con el objetivo de que el conocimiento adquirido sea utilizado para ampliar los comportamientos más allá de lo programado.

De esta necesidad es que surge toda una amplia investigación en Inteligencia Artificial que ha llegado a conformar el Aprendizaje de Máquina, campo concerniente con el desarrollo de teorías computacionales del aprendizaje y la construcción de sistemas de aprendizaje.

De los muchos trabajos de investigación en dicha área se han recopilado algunas definiciones al término "aprendizaje", las cuales se citan a continuación:

"El aprendizaje denota cambios en el sistema que son adaptativos en el sentido de que hacen posible que el sistema realice una misma tarea o tareas a partir de una misma población, más eficiente y efectivamente la próxima vez".

Simon [Sim83]

"Aprender es hacer cambios útiles en nuestras mentes".

Minsky [Min86]

"Aprender es construir o modificar representaciones de lo que está siendo experimentado".

Michalski [Mic86]

Definición operacional de aprendizaje: "Habilidad de ejecutar nuevas tareas que anteriormente no podían ejecutarse o de ejecutar mejor antigüas tareas (más rápido, con mayor precisión, etc.) como resultado de los cambios producidos por el proceso de aprendizaje".

Carbonell [Car89]

El aprendizaje es "mejorar automáticamente con la experiencia". Desde el punto de vista del

Aprendizaje de Máquina, el aprendizaje incluye "cualquier programa de computadora que mejore su desempeño en algunas tareas a través de la experiencia... Se dice que un programa de computadora aprende de la experiencia E con respecto a alguna clase de tareas T y medida de desempeño P, si su desempeño en las tareas T, medido por P, mejoran con la experiencia E".

Mitchell [Mit97]

"El aprendizaje es la mejoría del comportamiento de un sistema haciéndolo más apropiado para el ambiente en el cual el sistema está incorporado". "El problema de aprender a actuar es descubrir el mapeo de los estados perceptuales a las acciones". Aprendizaje es "la mejoría del comportamiento a través de la información incrementada acerca del ambiente".

Kaelbling [Kae93]

De todas estas definiciones queda clara la idea de que el aprendizaje implica cambiar en base a conocimientos adquiridos por medio de la experiencia y que esos cambios se reflejan en una mejora en la vida del aprendiz. El que una computadora sea capaz de aprender significa la posibilidad de tener sistemas inteligentes, autónomos, que sean capaces de adaptarse a situaciones ambientales por sí mismos, al ser capaces de adquirir conocimiento a través de la propia experiencia y utilizarlo de acuerdo a sus cambiantes necesidades. Estos motivos hacen del aprendizaje una de las metas más deseadas de los investigadores de IA.

2.1.1. ¿Por qué, qué y de dónde aprender?

Se ha hablado acerca de la necesidad de aprender que los sistemas computacionales tienen, pero no se ha especificado qué debe aprenderse y de dónde debe tomarse la experiencia para adquirir el conocimiento a aprender. Esta sección está dedicada a tales cuestiones.

El aprendizaje tiene dos propósitos universales que resumen el por qué se debe aprender [Mat94]. Estos propósitos radican en la utilidad del aprendizaje para:

- Adaptarse a cambios externos e internos.
- Simplificar la construcción interna de conocimiento.

Ahora bien, retomando la definición de *agente*, "sistema capaz de percibir y actuar en un medio ambiente" [Kae93] [Mat94], y la definición de *aprendizaje*, "mejoría del comportamiento de un sistema haciéndolo más apropiado para el ambiente en el cual el sistema está incorporado" [Kae93], se establece que el agente debe aprender a partir de la interacción con el ambiente que le rodea, el cual puede sensar y sobre el que puede actuar. Es precisamente de esta interacción de donde el agente debe tomar la experiencia, la cual le producirá el conocimiento a usar en sus posteriores acciones.

De lo anterior se tiene que el sistema, al experimentar en el medio que le rodea, debe observar y recabar información, es decir, debe aprender cómo trabaja el mundo y también debe aprender cuál es su tarea, o como [Kae93] enuncia: "el agente debe aprender a realizar una tarea en particular en un ambiente en particular".

Con lo anterior, al contestar la pregunta ¿qué aprender? diciendo: información del mundo y

cuál es la tarea (para llevarla al cabo), es necesario aclarar que hay infinidad de cosas que un agente puede aprender (aunque no muchas formas en que pueda lograrlo), por lo que debe establecerse el tipo de tarea de aprendizaje que el agente debe enfrentar. Una categoría en base a lo que es aprendido es la siguiente:

- Aprender conocimiento declarativo
  El aprendizaje de conocimiento declarativo es un área importante de la IA. Para agentes situados, los mapas del ambiente son el ejemplo más claro de conocimiento declarativo que se ha buscado que aprenda un agente. El construir y actualizar los mapas y modelos del mundo ha estado cercanamente relacionado a la acción en el mundo por parte de los agentes situados.
- Aprendizaje de control
  Basado en el control adaptativo y la teoría de control, este aprendizaje está orientado a sistemas dinámicos comúnmente involucrados en fábricas, donde deben monitorearse procesos y hacerse predicciones del comportamiento del sistema a fin de que se tenga un correcto desempeño, de acuerdo a los estándares establecidos.
- Aprender nuevos comportamientos
  El aprendizaje de nuevos comportamientos se refiere al problema de adquirir estrategias para alcanzar metas particulares. Es aprender "cómo" hacer algo.
- Aprender a seleccionar acciones/comportamientos
  Aprender "cuándo" hacer algo; decidir, de entre varias opciones, qué acción (o comportamiento) tomar en cada momento.

El presente trabajo está orientado al aprendizaje de selección de acciones que un agente debe realizar con el objetivo de mejorar al momento de escoger entre varias opciones, siendo esto crucial para su sobrevivencia.

Además, como se ha anticipado ya, el aprendizaje involucrado en esta investigación el es Aprendizaje por Reforzamiento (que se definirá en las siguientes secciones) y precisamente el aprendizaje de selección de acciones es por definición un "problema de Aprendizaje por Reforzamiento", debido a que está basado en correlacionar las acciones que elige y ejecuta el agente con la retroalimentación que recibe como resultado.

Habiendo establecido que el agente debe aprender una tarea a partir de su ambiente, ahora corresponde elegir la manera en que dicho aprendizaje debe llevarse al cabo, puesto que existen diferentes clases de aprendizaje. En la siguiente sección se dará una clasificación.

2.2 Tipos de Aprendizaje Autómatico

En base al tipo de retroalimentación disponible que el agente tiene, las tareas de aprendizaje pueden clasificarse en:

- **Aprendizaje Supervisado**
  Aprendizaje a partir de ejemplos - Se especifica lo que debe ser aprendido por cada ejemplo. El agente percibe las entradas y salidas esperadas al módulo de selección de acción y

se ajusta en base a las diferencias entre sus salidas y las deseadas (se conoce el comportamiento deseado). Por tanto, este tipo de aprendizaje requiere que un maestro especifique las respuestas correspondientes a las entradas.

- **Aprendizaje no Supervisado**
  Aprendizaje por observación - No existe ninguna retroalimentación del exterior (no se dan clasificaciones). El agente debe descubrir categorías y regularidades en los datos, es decir, busca patrones en las entradas y, por tanto, se limita a modelar la secuencia de percepciones. En este caso, no existe una respuesta explícita que sea la correcta.

- **Aprendizaje por Reforzamiento**
  En este tipo de aprendizaje se proporcionan entradas y en vez de comparar la salida contra un comportamiento deseado, se obtiene como retroalimentación una cierta valoración de la calidad de las acciones realizadas. Es decir, no se le indica al agente qué acciones debe elegir, pero en cambio recibe un valor que le indica la deseabilidad de la acción que ha elegido. El Aprendizaje por Reforzamiento actúa en función a premios/castigos, ajustando/aprendiendo las acciones a elegir. El propósito de estos sistemas es descubrir, mediante prueba y error, un mapeo desde las entradas a las salidas que maximice la recompensa (valoración positiva de acciones elegidas).

2.3 El Aprendizaje por Reforzamiento

En las siguientes secciones se expondrán aspectos concernientes al Aprendizaje por Reforzamiento, esenciales para la presente investigación, en virtud de que el módulo de aprendizaje propuesto para trabajar conjuntamente con las Redes de Comportamiento pertenece a este tipo de aprendizaje. Esta elección surge de que los métodos de Aprendizaje por Reforzamiento son los apropiados cuando se requiere que el sistema aprenda durante el tiempo de ejecución, sin que se posea de antemano ningún conocimiento de la organización y dinámicas del ambiente.

Se ha optado por trabajar con Aprendizaje por Reforzamiento dada la necesidad de que el agente modelado en este trabajo aprenda acerca de, a partir de, y mientras interactúa, mediante prueba y error, con un ambiente externo dinámico.

2.3.1. Concepto

El Aprendizaje por Reforzamiento es aprender qué hacer -cómo mapear sensaciones a acciones- a partir de la retroalimentación con el ambiente, relativa a cuán bien se actúa. Es decir, es aprender cómo comportarse de manera que se maximice una señal de recompensa numérica.

En base a esta definición se ha modificado la descripción general de agente mostrada en la figura 1.1, resultando un agente de Aprendizaje por Reforzamiento como el de la figura 2.1.

Los dos aspectos distintivos más importantes del Aprendizaje por Reforzamiento son:

- Búsqueda mediante prueba y error. No se le dice al aprendiz qué acciones tomar, sino más bien debe descubrir qué acciones producen la mayor recompensa al intentarlas.

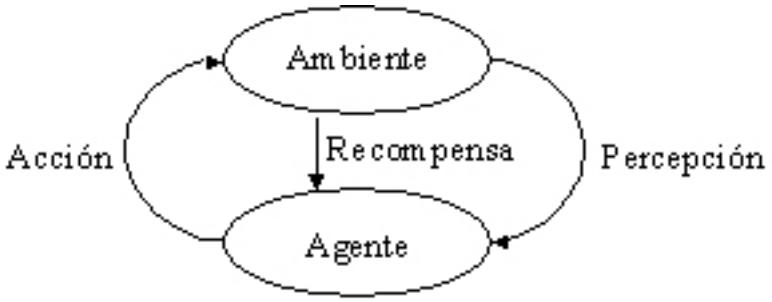

Figura 2.1: Agente ApR

- Recompensa retardada. En algunos casos, las acciones pueden afectar no sólo la recompensa inmediata, sino también la próxima sensación y, a través de esto, todas las recompensas subsecuentes.

El Aprendizaje por Reforzamiento no está definido para especificar algún método particular de aprendizaje o un conjunto de algoritmos de aprendizaje. Más bien, cualquier método que busque resolver un problema de Aprendizaje por Reforzamiento característico es considerado un método de Aprendizaje por Reforzamiento.

Existen dos estrategias principales para resolver problemas de Aprendizaje por Reforzamiento:

- Buscar en el espacio de comportamientos a fin de hallar uno que se ejecute bien en el ambiente. Este enfoque ha sido tomado por el trabajo en Algoritmos Genéticos y Programación Genética.
- Usar técnicas estadísticas y métodos de programación dinámica para estimar la utilidad de tomar acciones en estados del mundo.

La segunda estrategia toma ventaja de la estructura especial de los problemas de Aprendizaje por Reforzamiento que no está disponible en problemas de optimización en general (en ApR no se asume ningún conocimiento de la organización y dinámica del ambiente).

Los problemas de Aprendizaje por Reforzamiento involucran interactuar con un ambiente. El agente debe aprender acerca del ambiente y también debe descubrir cómo actuar óptimamente en ese ambiente. Por tanto, hay un componente estadístico (aprender acerca del ambiente) y un componente computacional (decidir cómo actuar).

Generalmente se espera que un aprendiz por reforzamiento se desempeñe pobremente en las etapas iniciales, pero que paulatinamente aprenda de la experiencia.

2.3.2. Historia del ApR

El Aprendizaje por Reforzamiento como campo de la Inteligencia Artificial se conforma a finales de 1980's. Anterior a esto, es posible encontrar los antecedentes que le dieron origen, así como

los trabajos pioneros en el área. Cabe destacar que es en los últimos años cuando este campo ha ganado importancia dentro de la IA hasta el grado de ser reconocido dentro de la clasificación de aprendizaje (ver sección anterior).

Como lo destacan Sutton y Barto [SyB98], los antecedentes principales del Aprendizaje por Reforzamiento pueden identificarse en la psicología del aprendizaje animal (misma que influencía igualmente a la escuela conductista de psicología) y en el control óptimo (programación dinámica):

- *Aprendizaje animal*. Basándose en el aprendizaje por prueba y error, Edward Thorndike formuló en 1911 la Ley del Efecto, planteando que un animal, al enfrentarse a una toma de decisiones en una situación determinada, al estar nuevamente en la misma situación elegirá aquello que le produzca mayor satisfacción (en base a sus respuestas anteriores), siendo muy poco probable que opte por aquello que le produzca insatisfacción. Esto se basa en la idea de que las acciones seguidas por buenos resultados tienen la tendencia de ser reseleccionadas, mientras que las malas acciones que llevan a la inconformidad del animal debilitan su tendencia a repetirse. Fue llamada Ley del Efecto por que precisamente describe el efecto de los eventos reforzados en la tendencia a seleccionar acciones.

  Los aspectos más importantes del aprendizaje por prueba y error son la selectividad y la asociatividad. La primera conlleva el probar acciones y seleccionarlas mediante la comparación de sus consecuencias; la segunda se refiere a que las acciones seleccionadas son asociadas a situaciones particulares. La Ley de Efecto cubre ambos aspectos.

  El trabajo de Thorndike fue considerado por Pavlov (1927) en sus experimentos de condicionamiento clásico y en general, el aprendizaje por estímulo-respuesta ha llegado a ser una metodología predominante para el estudio del comportamiento animal en psicología y biología.

- *Control óptimo*. Este término fue usado a finales de la década de 1950 para describir una técnica para minimizar una medida de desempeño de un sistema dinámico. Un enfoque al problema que esto representa fue desarrollado por Richard Bellman, quien en su Ecuación de Bellman (1957) define una ecuación para calcular la función de evaluación de un sistema dinámico. La clase de métodos para resolver problemas de control óptimo mediante la solución de esta ecuación son conocidos como Programación Dinámica; actualmente son fundamentales a la teoría y algoritmos del Aprendizaje por Reforzamiento moderno.

El deseo de que una computadora aprenda mediante prueba y error ya se veia en los trabajos de Turing (1950), pero es en 1954 cuando Minsky desarrolla el primer trabajo que involucra modelos computacionales de aprendizaje, en particular, aprendizaje por prueba y error. Por su parte, Samuel, en su juego de damas (1959) implementó un método de aprendizaje sin establecer conexiones al aprendizaje animal. Otros trabajos bajo el aprendizaje por prueba y error fueron: STeLLA, de John Andreae (1963); MENACE, de Donald Michie (1961, 1963), y; GLEE y BOXES, de Michie y Chambers (1968). Widrow, Maitra y Gupta (1973) desarrollan el “aprendizaje

con un crítico" al modificar el algoritmo LMS[1] con aprendizaje por reforzamiento. Estos fueron trabajos aislados enmedio del emergente campo del aprendizaje supervisado.

Es hasta la década de 1970 cuando, gracias a Harry Klopf, el Aprendizaje por Reforzamiento retoma al ya olvidado aprendizaje por prueba y error, a partir de lo cual se empieza a establecer la diferencia entre aprendizaje supervisado y aprendizaje por reforzamiento. En 1973 surgen, bajo la influencia del aprendizaje por prueba y error, los Autómatas de Aprendizaje (Tsetlin, 1973).

Junto con los trabajos de Klopf se da origen a una nueva área en el Aprendizaje por Reforzamiento: el aprendizaje por diferencia temporal, cuyos fundamentos se encuentran también en la psicología del aprendizaje animal, de donde toma el concepto de reforzador secundario (estímulo apareado con un reforzador primario, que llega a tomar propiedades de reforzamiento similares). La primera regla de aprendizaje por diferencia temporal fue propuesta por Ian Witten (1977), bajo el aprendizaje por prueba y error y el control óptimo.

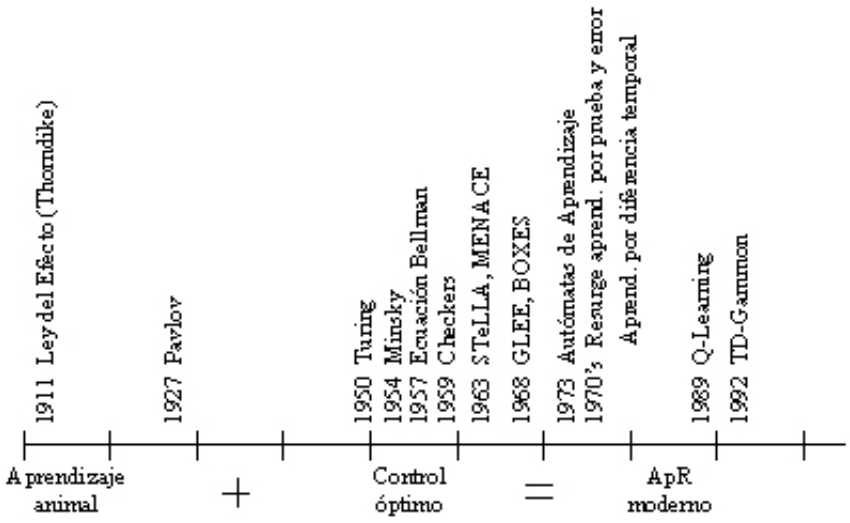

Figura 2.2: Breve historia del ApR

En 1989, Chris Watkins desarrolló el algoritmo Q-learning, con el cual conjuntó los tres hilos de investigación: aprendizaje por prueba y error, control óptimo y aprendizaje por diferencia temporal. En esa época el Aprendizaje por Reforzamiento cobra importancia dentro de la Inteligencia Artificial, en específico en el aprendizaje de máquina.

Un trabajo muy importante que debe mencionarse, por ser representativo del Aprendizaje por Reforzamiento, es el realizado en 1992 por Gerry Tesauro, quien usando técnicas de ApR desarrolló el TD-Gammon, el mejor juego computarizado de backgammon capaz de enfrentarse al campeón humano.

[1]El algoritmo LMS (Least Mean Squares) es una regla de entrenamiento que por cada ejemplo de entrenamiento observado ajusta los pesos en una pequeña cantidad en vías de reducir el error de ese ejemplo de entrenamiento.

A la fecha existe mucha investigación en esta área, con el desarrollo de muchos algoritmos y aplicaciones con resultados interesantes. La figura 2.2 presenta los puntos más destacados de la historia del ApR, ya que esta sección únicamente intenta dar a conocer los eventos más relevantes de esta área.

2.4 Modelo del Aprendizaje por Reforzamiento

En el modelo estándar del Aprendizaje por Reforzamiento, un agente está conectado a su ambiente via percepción y acción, como se bosqueja en la figura 2.3. Cabe destacar la correspondencia con el esquema general de agente (figura 1.1), estando ahora constituida la percepción por el estado del ambiente y una señal de reforzamiento que evalúa las decisiones tomadas.

El agente debe ser capaz de sensar su ambiente y de tomar decisiones que afecten el estado del ambiente, a fin de alcanzar una meta; debe ser capaz de aprender a partir de su propia experiencia (usándola para mejorar su desempeño).

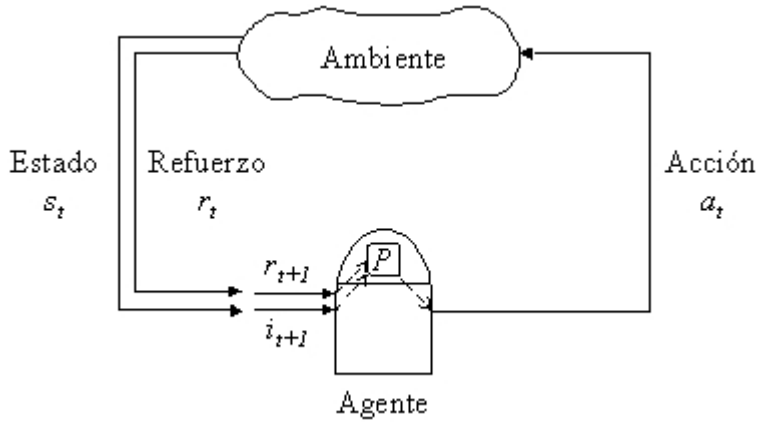

Figura 2.3: Modelo estándar del Aprendizaje por Reforzamiento

En cada momento el agente recibe como entrada i, alguna indicación del estado actual del ambiente, s; entonces el agente selecciona una acción, a, a generar como salida. La acción cambia el estado del ambiente, y el valor de esta transición de estado es comunicada al agente a través de una señal de reforzamiento escalar, r. El comportamiento del agente, P, debiera elegir acciones que tendieran a incrementar a largo plazo la suma de valores de la señal de reforzamiento. Esto puede aprenderlo a hacer mediante prueba y error sistemática, usando una amplia variedad de algoritmos.

El modelo formal del Aprendizaje por Reforzamiento consiste de:

- Un conjunto discreto S de estados del ambiente.
- Un conjunto discreto A de acciones del agente.

- Un conjunto de señales de reforzamiento escalar, r (típicamente {0,1}, o los números reales).
- Una función de entrada I, que determina la forma en que el agente ve el estado del ambiente.
- Una política π que mapea estados a acciones: $\pi : S \times A$

El objetivo único del agente es maximizar la cantidad de reforzamiento total recibida a largo plazo.

2.4.1. Elementos

Los elementos principales del Aprendizaje por Reforzamiento son, lógicamente, el agente y el ambiente del agente.

El *agente* es el aprendiz y tomador de decisiones. El *ambiente* es todo lo externo al agente, aquello con lo que interactúa.

El agente selecciona acciones y el ambiente responde a aquellas acciones, presentando nuevas situaciones al agente. El ambiente también da origen a recompensas, que el agente intenta maximizar durante toda su vida.

Además del agente y del ambiente, existen cuatro subelementos principales en un sistema de Aprendizaje por Reforzamiento: una política, una función de recompensa, una función de evaluación y un modelo del ambiente (opcional).

Política

Define la manera de comportarse del agente. Es un mapeo de los estados percibidos a las acciones a ser tomadas. La política es el núcleo de un agente de Aprendizaje por Reforzamiento en el sentido de que por si sola es suficiente para determinar el comportamiento.

Función de recompensa

Define cuáles son los buenos y los malos eventos para el agente. Mapea cada estado percibido (o par estado-acción) a un número único, una *recompensa*, indicando la deseabilidad de ese estado (debida a la acción tomada). Especifica lo que es bueno en un sentido inmediato.

Función de evaluación

Trabaja en base a los valores de los estados, especificando lo que es bueno a largo plazo. El valor de un estado es la cantidad total de recompensa que un agente puede esperar acumular a futuro, comenzando desde ese estado. Los valores, como predicciones de recompensas, indican la deseabilidad a largo plazo.

Para aclarar la diferencia entre recompensa y valor, considérese la siguiente situación: la acción tomada más recientemente ha producido una recompensa baja; no obstante, esa acción aún tiene un valor alto, debido a su buen desempeño histórico, lo que la hace deseable.

Es más difícil determinar valores que determinar recompensas, pero el único propósito de la estimación de valores es la obtención de más recompensa. Por otra parte, es con los valores con los que se trabaja al momento de tomar y evaluar decisiones. La selección de acciones está basada en juicios de valor.

Por todo esto, el componente más importante de casi todos los algoritmos de Aprendizaje por Reforzamiento es un método para estimar valores eficientemente.

Modelo del ambiente

El modelo de un ambiente es algo que imita el comportamiento de ese ambiente; por ejemplo, un modelo de predicción del próximo estado resultante y la proxima recompensa, dados un estado y acción actuales. Éste es un elemento opcional en los sistemas de Aprendizaje por Reforzamiento.

2.4.2. Algoritmo general

Un algoritmo de Aprendizaje por Reforzamiento es aquel que aprende un comportamiento apropiado para un agente en un mundo, mapeando entradas a acciones, y que requiere una entrada adicional: el valor de reforzamiento del estado del mundo para el agente.

El algoritmo general del Aprendizaje por Reforzamiento se presenta en el algoritmo 2. El aprendizaje está constituido por tres partes:

- s Estado interno - Guarda información acerca de las interacciones del agente con el mundo. El estado inicial es denotado por s_0

- e Función de evaluación - Mapea el estado interno y la entrada en una acción, seleccionando la acción que parece más útil para el agente en esa situación, en base a la información acerca del mundo almacenada en el estado interno. La utilidad de una acción puede deberse a dos situaciones: 1) tiene un alto valor de información, o; 2) el agente sabe poco acerca de su resultado.

- u Función de actualización - Mapea el estado interno del agente, su entrada, su acción y el valor de reforzamiento en un nuevo estado interno, ajustando el estado actual en base al reforzamiento resultante de ejecutar esa acción en esa entrada.

Algoritmo 2 Algoritmo General del Aprendizaje por Reforzamiento

1: $s = s_0$ ▷ Estado inicial
2: **loop**
3: i = entrada ▷ Lee entrada del ambiente
4: $a = e(s, i)$ ▷ Toma acción en base a la función evaluadora
5: r = reforzamiento ▷ Se determina el valor de la acción
6: $s = u(s, i, a, r)$ ▷ Actualiza el estado interno del agente
7: **end loop**

La figura 2.4 presenta la estructura interna del comportamiento del Aprendizaje por Reforzamiento; las líneas punteadas representan al estado inicial s_0, en cuyo caso la función evaluadora e actúa como mapa de acción libre de estado.

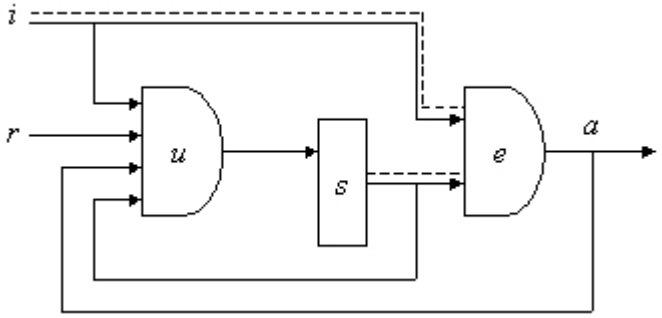

Figura 2.4: Esquematización del proceso de Aprendizaje por Reforzamiento

2.5 Equilibrio explotación-exploración

Inicialmente, el agente debe intentar una variedad de acciones y favorecer progresivamente a aquella que parezca ser la mejor. Como el objetivo del Aprendizaje por Reforzamiento es obtener gran cantidad de recompensa, el agente debe preferir las acciones que ha probado en el pasado y que han sido efectivas en la producción de recompensa. A la preferencia de acciones conocidas se le conoce como *explotación*, ya que es el aprovechamiento del conocimiento recabado por el agente en relación al buen desempeño de las acciones.

Un problema con este método de aprendizaje es que si un agente siempre se dedicara a explotar lo que sabe, puede suceder que posea un conocimiento erróneo acerca de las acciones y entonces esté favoreciendo a acciones sub-óptimas, ignorando totalmente el que existe una acción mejor.

Una manera de superar el inconveniente de la explotación pura radica en que el agente debe probar otras acciones de las que no tiene demasiado conocimiento o que parecen ser peor, lo que le lleva a ganar información acerca de sus valores de reforzamiento y en ocasiones le hace descubrir información completamente nueva de tales acciones, como por ejemplo, encontrar la acción óptima. A la elección de acciones desconocidas se le llama *exploración*.

Uno de los aspectos más interesantes del Aprendizaje por Reforzamiento es precisamente el equilibrar la explotación contra la exploración. Un agente debe decidir en qué momento debe incrementar el reforzamiento total y en qué momento puede optar por intentar descubrir nuevo conocimiento, en busca de una mejor acción.

Para el aprendiz por reforzamiento es muy importante actuar en base a información correcta, por tanto, la recolección de información le es absolutamente necesaria, pues con una mala decisión puede converger prematuramente a una acción sub-óptima.

La exploración del ambiente es una de las principales diferencias entre el aprendizaje supervisado y el Aprendizaje por Reforzamiento.

2.6 El Problema del Bandido

En la comunidad de estadística, el problema del Aprendizaje por Reforzamiento es analizado mediante el problema del "Bandido de los 2-brazos", denominado así por su similitud con las máquinas tragamonedas de juego/apuesta del mismo nombre, cuyo funcionamiento es el siguiente: una máquina consta de 2 brazos o palancas que, independientemente, cada vez que una de ellas es jalada, pueden o no pagar cierta cantidad de dinero. La recompensa dada por una palanca es igualmente independiente de lo que acontezca alrededor de la máquina y del jugador. El objetivo de jugar con tales máquinas es obtener la mayor cantidad de ganancia económica posible.

El jugador puede optar por elegir una palanca al azar o bien, desarrollar una estrategia para ganar la máxima paga con el tiempo, en base a la experiencia previa de jugadas (palanca jalada - pago), lo que definitivamente es mejor que optar por el azar. Un ejemplo de estrategia es la de "permanecer con un ganador, pero cambiar con un perdedor", indicando que se debe jalar una palanca hasta que se pierda, entonces cambiar a otra y permanecer jalando hasta que se pierda, y asi en lo sucesivo.

Una extensión al paradigma del Bandido de los 2-brazos es su generalización: el problema del "Bandido de los N-brazos", donde el problema de aprendizaje consiste de seleccionar entre N alternativas (dígase palancas o, en el lenguaje de aprendizaje, acciones).

En estos problemas se asume lo siguiente:

- Los eventos de jalar los brazos (palancas) son independientes.
- Los brazos no pagan nada o pagan una cantidad fija.
- La probabilidad de que cada brazo pague permanece constante durante todo el juego.
- El mundo elegirá las probabilidades en la peor forma para el jugador.

Dicho en otros términos, éste es un problema donde se tiene que seleccionar entre N acciones, obteniendo una recompensa por cada acción seleccionada, y mediante esto, aprender a escoger la mejor de ellas para maximizar la recompensa total.

Las situaciones planteadas en el problema del Bandido constituyen en sí un problema de selección de acción; es un escenario simplificado del problema de Aprendizaje por Reforzamiento en su versión no asociativa, que involucra aprender a actuar en una situación. Por tales motivos, el paradigma del Bandido ha sido ampliamente usado para analizar el problema de aprender mediante reforzamiento. De ahí que sirva de base para la presente investigación, cuyo objetivo es precisamente optimizar la selección de acciones por parte de un agente mediante técnicas de aprendizaje.

2.7 Algoritmos

A continuación se expondrán algunos de los algoritmos de Aprendizaje por Reforzamiento dentro de un escenario de retroalimentación puramente evaluativa, la cual indica cuán buena es una acción tomada, pero no si es la mejor o la peor acción posible.

En primer lugar, se presenta el algoritmo del Bandido, el cuál hace un análisis en base a la recompensa neta que se acumula al seleccionar las acciones. Posteriormente se exponen diversas técnicas con características que permiten clasificar a los algoritmos ApR. Los métodos que se presentan están clasificados en métodos de evaluación de acciones, estrategias de vectores de probabilidad, métodos de comparación de reforzamientos y técnicas de estimación de intervalos. Todos ellos trabajan sobre el problema de aprender a seleccionar acciones.

2.7.1. Algoritmo del Bandido

En relación al problema del Bandido (sección 2.6), en el ámbito computacional, el problema del "Bandido de los N-brazos"ha sido ampliamente estudiado, refiriéndose, como ya se comentó, al problema de enfrentarse a una elección entre N opciones (o acciones) diferentes. Después de cada decisión se recibe una recompensa numérica seleccionada desde una distribución de probabilidades estática que depende de la acción seleccionada. Igual que en el caso real, el objetivo es maximizar la recompensa total esperada sobre un periodo de tiempo.

El algoritmo básico es la versión para dos tipos de acciones, el problema del "Bandido de los 2-brazos", para el caso de decidir entre dos acciones, para lo cual se alterna entre ambas y se registra el número de éxitos de cada una; cuando el número de éxitos de una acción excede el número de éxitos de la otra acción por una cantidad k, se elige ganadora por siempre en el futuro.

2.7.2. Métodos de evaluación de acciones

Como ya se indicó con el problema del Bandido, el problema de aprender a seleccionar acciones, desde la perspectiva del Aprendizaje por Reforzamiento, implica obtener una medida numérica o recompensa por cada acción que es seleccionada, la cual indica la deseabilidad de esa acción.

En base a las recompensas recibidas, cada acción tiene una recompensa esperada o promedio, que se denomina el *valor* de la acción, denotado por $Q(a)$.

Si se conociera el valor real $Q^*(a)$ de cada acción, sería fácil resolver el problema del Bandido y en general, de la selección de acción, pues se escogería siempre a la acción que tuviera el valor más grande. Pero precisamente el problema que se enfrenta radica en que no es posible conocer los valores de las acciones con certeza y, por tanto, se debe recurrir a una estimación de los mismos.

Una manera de calcular el valor estimado de una acción, denotado por $Q_t(a)$, es promediar las recompensas recibidas (al tiempo t) cuando esa acción fue seleccionada, a lo que se le denomina el método del promedio de la muestra para la estimación de valores de acciones, dado que cada estimado es un simple promedio de la muestra de recompensas relativas.

Una vez explicado lo anterior, finalmente se puede señalar que los métodos de evaluación de acciones estan basados precisamente en la estimación de los valores de las acciones y en el uso de tales estimados al momento de llevar al cabo la selección de las acciones.

Dentro de estos métodos se presentan los algoritmos Greedy, ε-Greedy y Softmax, clasificados en estrategias codiciosas y aleatorias.

2.7.2.1. Estrategias codiciosas

Escogen los estimados de valores de acción más grandes, considerándolos como indicadores de las mejores acciones a seguir. Por tanto, estos métodos explotan el conocimiento que tienen para maximizar la recompensa inmediata, sin perder tiempo en probar acciones que consideran malas.

A su favor tienen el hecho de que, a medida que el número de veces que se ha seleccionado una acción a, tiende a infinito, por la Ley de los Grandes Números, su recompensa esperada $Q_t(a)$ converge a su valor real $Q^*(a)$.

Un inconveniente de estos algoritmos es que pueden caer prematuramente en la selección de valores no-óptimos.

El algoritmo del Bandido es clasificado como una estrategia codiciosa, pues una vez que ha evaluado todas las alternativas posibles, se dedica a seleccionar siempre a la que considera, es la mejor, ignorando totalmente a las demás (explota su conocimiento). La única cuestion es que los algoritmos básicos del Bandido hacen su selección evaluando la recompensa total recibida de cada acción, no su recompensa promedio (no aplican estrictamente el método de evaluación de acciones).

Algoritmo Greedy (Codicioso)

En cada momento existe al menos una acción que posee el valor estimado más grande. A esta acción se le denomina *codiciosa* - "greedy" en inglés.

Este algoritmo se basa en la regla más simple de selección de acción: en todo momento selecciona la acción (o una de las acciones) con el valor de acción estimado más grande, es decir,

selecciona la acción codiciosa a^*, para la cual $Q^*(a) = max_a Q_t(a)$

2.7.2.2. Estrategias aleatorias

Surgen como una mejoría a los métodos codiciosos, en particular al algoritmo Greedy, buscando evitar quedar atrapados en acciones sub-óptimas. La operación de tales estrategias es comportarse codiciosamente la mayor parte del tiempo, pero de vez en cuando, seleccionar una acción aleatoriamente, sin tomar en cuenta los estimados de los valores de las acciones.

Dada la azarosidad que ocasionalmente introducen estos métodos, además de explotar su conocimiento, realizan exploración para adquirir nueva información, con lo que mejoran sus posibilidades de encontrar la acción óptima.

Algoritmo ε-Greedy

La selección de acción sigue una estrategia codiciosa la mayor parte del tiempo, pero ocasionalmente, de acuerdo a una pequeña probabilidad ε, selecciona una acción al azar, de manera uniforme, independientemente de las estimaciones de las acciones.

La ventaja con que cuenta este algoritmo es que, en el límite, a medida que se incrementa el número de selecciones de acciones, toda acción será probada un número infinito de veces, garantizando así que, para toda a, el número de veces que una acción a se ha seleccionado tienda a infinito y, en consecuencia, todo valor estimado $Q_t(a)$ converja a su valor real $Q^*(a)$.

Éste es un método efectivo y popular de balance de explotación/exploración en el Aprendizaje por Reforzamiento, pero presenta el inconveniente de que al explorar, dado que selecciona uniformemente una acción al azar, es posible que igualmente escoja la peor acción o la segunda mejor acción, ya que todas son susceptibles de ser seleccionadas.

Algoritmo Softmax

Esta estrategia trata de superar el inconveniente de la uniformidad en la selección aleatoria del método ε-Greedy. Durante casi todo el tiempo se comporta de manera codiciosa, selecciónando la acción con el máximo valor estimado, pero cuando lo permite una probabilidad ε, selecciona una acción aleatoriamente.

A diferencia de ε-Greedy, la solución planteada por esta estrategia propone variar las probabilidades con que son seleccionadas las acciones, usando el valor estimado de las mismas $Q_t(a)$, mediante la aplicación de alguna variante de las distribuciones de Boltzmann o de Gibbs, tomadas de las técnicas de recocido simulado, aunque existen otras maneras de producir el efecto *softmax*. La distribución de Boltzmann es calculada con la fórmula 2.1.

$$P(a) = \frac{e^{Q_t(a)/T}}{\sum_{a' \in A} e^{Q_t(a')/T}} \tag{2.1}$$

Un inconveniente que presenta es que requiere la calibración del parámetro T. La exploración

softmax trabaja bien si la mejor acción está bien separada de las otras, pero padece un poco cuando los valores de las acciones son cercanos.

2.7.3. Estrategias basadas en vectores

Métodos clásicos del campo de aprendizaje automático en los que el estado interno del algoritmo de aprendizaje es representado como un vector de números no negativos que suman uno. El agente selecciona una acción probabilísticamente, con la probabilidad (p_n) de que seleccione la n-ésima acción igual al n-ésimo elemento del vector de estados.

El problema en estas estrategias radica en actualizar los valores en el vector de estados en base a la acción más reciente y su resultado.

Existen varios algoritmos pertenecientes a este enfoque, pero en particular se presentan dos de ellos, dado que se optó por su implementación en este trabajo.

Algoritmo LRP - Linear Reward-Penalty (Recompensa-Costo Lineal)

Su nombre se refiere a que la actualización que se realiza es lineal en las probabilidades y a que se ejecuta tanto en juegos de éxito como de fracaso. En su versión original, para dos acciones, la actualización del vector de probabilidades depende del valor de reforzamiento obtenido, el cual es de tipo booleano. Así, sucede que:

1. Una acción a con un $r = 1$ (éxito), incrementa su probabilidad p_a de ser ejecutada en $1 - p_a$.
2. Una acción a con un $r = 0$ (fracaso), incrementa la probabilidad $p_{a'}$ de ejecutar la otra acción a' en proporción a su probabilidad p_a actual.

Este algoritmo hace uso de dos parámetros, denotados por α y β, que controlan la cantidad de ajuste de éxito y fracaso. Tiene estados no absorbentes, lo que significa que siempre ejecuta la acción incorrecta con alguna probabilidad diferente de 0.

Algoritmo LR - Linear Reward (Recompensa Lineal)

Variante del algortimo LRP, caracterizado por no realizar ajustes al vector de probabilidades cuando se recibe un valor de reforzamiento de 0, por lo que sólo hace uso del parámetro α para el ajuste de éxito. Tiene estados absorbentes; por ejemplo, cuando una de las probabilidades se hace 0, nunca será tomada su acción asociada.

2.7.4. Métodos de comparación de reforzamientos

La idea principal en estos métodos radica en la interpretación de una recompensa en comparación con algún nivel estándar o de referencia, con la finalidad de que el agente pueda considerar a una recompensa como grande si supera a la recompensa de referencia o como pequeña en caso contrario. Por lo general, la recompensa de referencia es un promedio de recompensas previamente recibidas.

Estos métodos pueden ser muy eficientes, desempeñándose en ocasiones aún mejor que los métodos de evaluación de acción.

Algoritmo RC - Reinforcement Comparison (Comparación de Reforzamientos)

Algoritmo propuesto por Sutton [SyB98]. Para elegir entre las acciones, además de calcular la recompensa de referencia, mantiene una medida de la preferencia de cada acción, la cual modifica cada vez que se selecciona la acción, tomando en cuenta la diferencia entre la recompensa recibida y la recompensa de referencia, con lo que se implementa la idea de que altas recompensas incrementan la probabilidad de reseleccionar la acción tomada y bajas recompensas la decrementan. Finalmente, estas preferencias se usan para determinar las probabilidades de selección de las acciones.

2.7.5. Técnicas basadas en intervalos

Basadas en la técnica estadística de construcción de estimados de intervalos de confianza de cantidades, estos métodos proponen almacenar un estimado del reforzamiento esperado para cada acción, junto con alguna información acerca de cuán bueno es ese estimado. El tamaño del intervalo de confianza es una medida de la falta de información acerca de la cantidad que está siendo estimada.

El objetivo de estas estrategias es hacer más eficiente la exploración, planteando que esto se logra cuando se tiene información acerca de la certeza de los valores estimados de las acciones.

Algoritmo IE - Interval Estimation (Estimación de Intervalos de Confianza)

Este algoritmo fue propuesto por Kaelbling [Kae93]. El algoritmo codifica los estimados de las acciones junto con el grado de confianza que se tiene en tales estimados, el cual se calculada en base al número de intentos y la recompensa recibida de cada acción. La medida adicional que se obtiene es el límite superior de un intervalo de confianza, $ub(x, n)$, calculado mediante la ecuación 2.2:

$$ub(x,n) = \frac{\frac{x}{n} + \frac{z^2}{2n} + \frac{z}{\sqrt{n}}\sqrt{\left(\frac{x}{n}\right)\left(1-\frac{x}{n}\right) + \frac{z^2}{4n}}}{1+\frac{z^2}{n}} \tag{2.2}$$

donde, para la acción a, x_a es la suma de reforzamiento recibido, n_a es el número de intentos de a y z es un parámetro que controla el tamaño del intervalo de confianza.

Al inicio, cada una de las acciones tendrá un límite superior igual a 1, pero a medida que aumenta el número de intentos, los límites de confianza se ajustan, teniendo en cuenta que existen dos razones para que éstos puedan ser altos: hay poca información acerca de la acción, o; hay información de que la acción es buena.

La función de evaluación selecciona la acción con el límite superior más alto de reforzamien-

to esperado. Este método balancea el actuar para ganar información con el actuar para ganar reforzamiento.

Existe otro algoritmo similar a éste, el algoritmo de Lai; ambos métodos usan la noción de seleccionar la accción con el límite superior de confianza más alto, pero los límites de confianza del algoritmo de Lai son más complejos que los métodos estadísticos estándar usados en el algoritmo IE.

Capítulo 3
Antecedentes

A continuación se presentan los trabajos previos a esta propuesta de investigación, considerando en primer lugar a la herramienta ABC, una implementación de las Redes de Comportamiento, sobre la cual se hará una extensión; posteriormente se hablará de otra herramienta implementada, XLearn, inspiradora directa del uso del Aprendizaje por Reforzamiento en modelos etológicos simulados; asimismo, se expondrán los estudios etológicos que sirven de base para el desarrollo de los experimentos con la nueva implementación. Finalmente, se presentará el mejoramiento propuesto en el presente trabajo de tesis, cuyo desarrollo se da en el capítulo 4.

3.1 Herramienta ABC

En el año de 1997 se defendió en la Maestría en Inteligencia Artificial un trabajo de tesis desarrollado por Alejandro Guerra, quien se propuso realizar una implementación del algoritmo propuesto por Maes [Mae89], presentado en el capítulo 1. La tesis se titula "Agentes Autónomos y Selección de Acción, una perspectiva basada en Redes de Comportamiento" [Gue97c], y la implementación resultante es conocida como ABC.

ABC es una herramienta auxiliar en la experimentación con agentes basados en Redes de Comportamiento, conformada de un ambiente gráfico que permite el diseño y simulación de agentes basados en este mecanismo de selección de acción, proporcionando al usuario la capacidad de observar el desempeño del agente a fin de ajustar su comportamiento. ABC está programada en lenguaje Tcl/Tk, por lo cual es altamente portable a diversos ambientes: UNIX, Windows y MAC OS.

El motivo por el cual se eligió la propuesta de Maes [Mae89] para implementarse se basó precisamente en las propiedades del mecanismo: comportamiento guiado por metas, relevante a la situación del agente, capaz de adaptación, robusto, persistente, reactivo y rápido.

Mediante la interfaz gráfica de ABC se permite al usuario describir los módulos conductuales que conforman al agente a diseñar/simular, con lo cual se desliga al usuario final de detalles de programación. Las Redes de Comportamiento enfrentan un problema en relación a la elección adecuada del conjunto de valores para los diversos parámetros necesarios en este algoritmo, de los cuales dependerá en cierto grado, el que un agente se comporte de manera correcta. Como ya se mencionó, mediante las capacidades gráficas de ABC es posible que el usuario observe

el comportamiento del agente que se modela, con lo cual obtiene información (al momento de ejecución) que le será de ayuda para un posible ajuste de parámetros. El comportamiento de un agente (la competencia entre los módulos de habilidad) puede ser monitoreado por dos vias:

1 Mediante un informe dinámico que lista las conductas que conforman al agente, junto con información que refleja el estado actual de las mismas, y/o

2 Por una gráfica de activación sobre el tiempo, la cual ofrece información relativa al nivel de activación de cada conducta sobre el tiempo, visualizando las secuencias de acciones que conforman el comportamiento del agente.

Dado que una de las condiciones para que un agente sea autónomo es que se desenvuelva en un medio ambiente dinámico, ABC permite realizar cambios en la definición del medio y del agente mismo, con el objetivo de que el agente modifique su percepción del ambiente y se enfrente a nuevas situaciones, como cambio de metas o ambientes ruidosos.

La ventana principal de ABC (figura 3.1) presenta la información que describe al agente a modelar y su medio; está conformada de diversas secciones, básicamente: percepción, simulación y control/operación del simulador. La percepción del agente abarca el mundo y las metas del agente, por lo cual se informa acerca de las listas de proposiciones que se observan verdaderas en el medio ambiente en un momento dado, así como de las metas que el agente ya realizó y las que debe alcanzar.

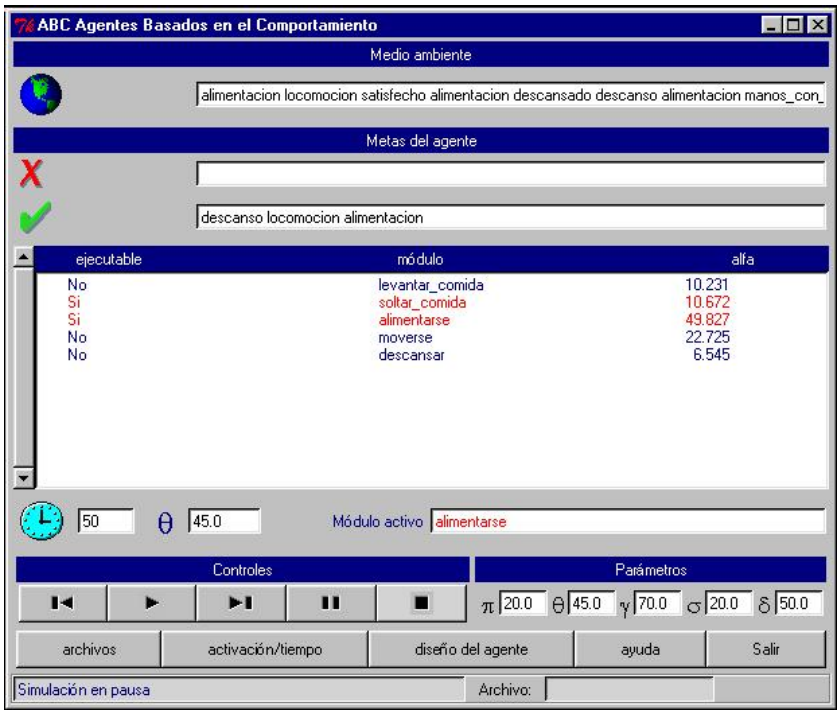

Figura 3.1: Ventana principal de ABC (primera versión)

El área de simulación presenta los módulos conductuales que definen al agente, informando si son o no ejecutables y su nivel de activación; también se muestra el ciclo de reloj en el que se encuentra la simulación, el valor del umbral de activación (θ) y el módulo que se encuentra activo. La operación de ABC se realiza mediante botones que permiten iniciar la ejecución, ya sea paso a paso o de forma contínua. Además cuenta con botones para accesar a otros componentes de ABC, tales como la administración de archivos, la gráfica de activación en el tiempo y el módulo de diseño de agentes. Los parámetros globales de la red pueden ser modificados en cualquier momento de la ejecución, dado que están presentes en esta ventana.

La gráfica de activación en el tiempo forma parte de una ventana (figura 3.2) que contiene información relativa al comportamiento del agente que puede ser analizada a fin de configurar los parámetros de la red para obtener un comportamiento adecuado por parte del agente; tal gráfica se conforma de la lista de módulos conductuales que componen al agente, presentando una gráfica de barras cuyo eje X lo conforma el tiempo transcurrido desde el inicio de la simulación, etiquetado en series de 10 ciclos de simulación.

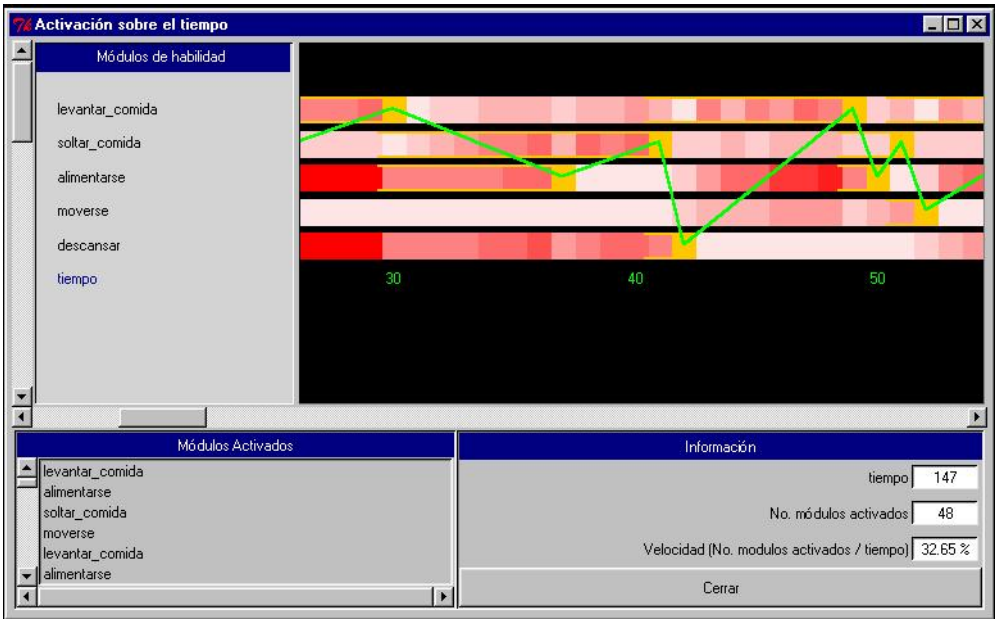

Figura 3.2: Gráfica de activación en el tiempo

Cada módulo tiene una barra, misma que varía en tonalidades blancas a rojas de acuerdo a su nivel de activación: un cuadro blanco representa un nivel de activación de 0, un cuadro rojo significa que el nivel de activación se encuentra 100 % por encima del umbral de activación. Un cuadro con una linea amarilla arriba y abajo de él indica que el módulo al que pertenece es ejecutable. Un cuadro amarillo señala que el módulo está activo en ese tiempo. La línea verde sobre los cuadros delinea la secuencia de activación de los módulos. En la ventana también se listan todas las conductas que han sido activadas durante la simulación, así como también se muestra el ciclo de tiempo en el cual se graficó, el total de módulos activados y la velocidad del agente.

Primeros experimentos

Guerra [Gue97c] realizó una serie de experimentos a fin de probar la funcionalidad de ABC. Para ello eligió tres ejemplos, dos de los cuales son problemas planteados anteriormente en el campo de la IA (el robot de Charniak y el Mundo de los Cubos) y el tercero fue tomado en base a un proyecto de investigación etológico a fin de emular la conducta de forrajeo del mono aullador (*Alouatta palliata*).

Robot de Charniak

El primer ejemplo consiste en la definición de un agente que emule el comportamiento del robot de Charniak [Cha85], esto es, plantear la existencia de un robot de dos manos cuyas metas son lijar una tabla y pintarse a sí mismo. El medio ambiente está constituido por una mesa de trabajo, una pintura en spray, una lija y una tabla. Existe una restricción en este problema: al momento en que el robot se pinta a sí mismo, queda fuera de operación (no puede realizar más acciones). Así, la red diseñada para modelar tal agente comprendió módulos conductuales referentes a las acciones básicas del robot: tomar objeto, soltar objeto, apoyarse en mesa, lijar tabla, pintarse a sí mismo.

Resultados: La Red de Comportamiento diseñada para simular al robot de Charniak demostró un buen comportamiento, acorde a lo que se esperaba de ella; este ejemplo mostró ser tolerante a fallas, al ser expuesto a situaciones que pudieron quebrantar su desempeño.

El Mundo de los Cubos

El segundo experimento tomó como base un problema clásico de planeación: el Mundo de los Cubos. Este mundo consiste de un número de bloques cuadrados, todos del mismo tamaño, colocados sobre una superficie plana. Los cubos pueden estar apilados, pero sólo un bloque puede estar acomodado directamente sobre otro. Existe un brazo robótico que puede manipular los cubos, pudiendo levantar un bloque y moverlo a otra posición, ya sea sobre la superficie plana o encima de algún otro cubo; únicamente se le permite levantar un bloque a la vez. La meta es construir una o más pilas de bloques, especificando cuáles bloques estarán por encima de qué otros bloques. Por ejemplo, una meta estaría dada por la construcción de dos pilas, una con el cubo A encima del cubo B, y la otra con C sobre D.

En particular, para este experimento se construyó un ambiente gráfico auxiliar en la manipulación del medio ambiente. La meta final era construir una o más pilas de bloques, a partir de cubos ubicados o no en pilas ya existentes, de acuerdo a especificaciones particulares de sobreposición de bloques, dadas en la red.

Resultados: El agente situado en el mundo de los cubos fue capaz de resolver diversas configuraciones, a petición del usuario. La red demostró tolerancia a fallas cuando se le introdujo ruido o cambios imprevistos al tiempo de ejecución de la simulación.

Proyecto Monots

El último experimento para demostrar la funcionalidad de ABC consistió en relacionar a esta herramienta con un problema del campo de la etología (estudio del comportamiento animal), debido a que el mecanismo de selección de acción en estudio (Redes de Comportamiento) presenta una clara influencia etológica. El proyecto Monots [Mar96a, Mar96b, Gar97, Gue97a, Gue97b, Gue98] fue el elegido para simular, dado que es un proyecto que estudia la conducta de forrajeo del mono aullador (*Alouatta palliata*) desde la perspectiva de generación de modelos que ofrezcan un marco teórico posible para explicar las operaciones experimentales. Con ABC aplicado a este problema se pretendió que el simulador fuera de ayuda en el enriquecimiento de un modelo de forrajeo basado en la simplificación de las Redes de Comportamiento.

La Red de Comportamiento utilizada para generar la conducta de forrajeo del mono aullador fue muy sencilla, ya que no contemplaba el establecimiento de la relación entre las motivaciones y la conducta de forrajeo en el mono. Precisamente, esto es propuesto como trabajo futuro.

Resultados: El comportamiento del agente exhibió un patrón interesante en donde el mono se alimenta, descansa y luego se mueve de forma coherente. Se observó la necesidad de las motivaciones, a fin de contrarrestar una sub-valoración de la competencia entre los módulos conductuales. Se observa que el comportamiento del agente cae en un ciclo ante la ausencia de factores que obligen al mono a contender con cambios en su medio ambiente.

Como nota importante cabe destacar que los parámetros utilizados en cada simulación fueron seleccionados después de una serie de experimentos, según los comportamientos obtenidos del agente modelado. Finalmente, se eligieron para cada experimento, aquellos parámetros con los que se observó un buen desempeño del simulador.

3.1.1. Extensión

Como continuación del trabajo anteriormente presentado, en 1998 Fernando Montes [Mon98] realiza la extensión de ABC, misma que se presenta bajo el trabajo de tesis titulado "Un Modelo de Conductas Animales basado en el Mecanismo de Selección de Acción de Maes".

Como se mencionó en el apartado previo, el experimento del comportamiento de forrajeo del mono aullador tiene su fundamento en estudios etológicos [Rod94], a partir de las observaciones recabadas en la isla de Agaltepec en el lago de Catemaco, Ver., Mex., zona de confinamiento de un grupo de *Alouattas palliatas*, conocidos comúnmente como monos aulladores, los cuales forman parte de un programa conservacionista de dicha especie. Como resultados de tales observaciones se obtuvo una relación estadística de las variables relevantes. Con ABC aplicado a este problema se busca que el simulador sea de ayuda en el enriquecimiento de un modelo de forrajeo basado en la simplificación de las Redes de Comportamiento [Gue97a, Gue97b, Gar97].

El objetivo de esta extensión a la herramienta ABC es el diseño de un modelo computacional que capture la estrategia de forrajeo del mono aullador y que al comparar los resultados de la simulación con los datos reales, se obtenga un comportamiento similar. Dado que ya existía la primera versión de ABC con el experimento de una red sencilla de modelado de la conducta de

forrajeo del mono aullador, se busca extender ABC para poder modelar conductas compuestas en animales.

Modificaciones

La herramienta ABC tuvo que ser modificada a fin de que la conducta de forrajeo simulada presentara una buena aproximación a la conducta del mono real. En la primera versión, las metas eran definidas como *metas temporales*, asumiéndose que todas tenian un peso asociado igual a 1. Por su lado, las motivaciones participaban en el cálculo de activación con un mismo peso. Fue necesario modificar el uso de metas, permitiéndose el uso de *metas permanentes* y *metas temporales*; las metas pueden ser usadas como motivaciones al asociarles un peso, mismo que representa la fuerza de la motivación. El peso en cuestión debe ser definido al inicio de la simulación, a fin de que las motivaciones participen en el cálculo de la activación de un módulo conductual. De acuerdo al algoritmo de Maes [Mae89], a las conductas que satisfacen motivaciones se les conoce como consumatorias, y su activación reduce la fuerza de activación asociada. Las conductas que contribuyen a que las consumatorias sean ejecutables se conocen como conductas apetitivas.

Dentro de la definición de módulos de comportamiento, se agregó una parte de acción que permite la continua modificación de los pesos asociados a las metas, a fin de lograr que el agente exhiba un mejor desempeño. Con esto, la nueva versión de ABC contempla la posibilidad de definir condiciones variables en el mundo del agente, junto con la definición de su negación explícita; también se establecieron intervalos de tiempo sobre los cuales cambiar las proposiciones variables en el mundo. Conjuntamente, es posible definir el valor de un incremento (a sumar o restar) de los pesos asociados a las motivaciones de las conductas, lo cual permite que exista una mejor calibración. Con esta versión es posible consultar datos referentes a los porcentajes de tiempo dedicado a cada conducta, el tiempo promedio (en ciclos de simulación) dedicado a cada actividad, e incluso, una matriz de transiciones entre conductas.

Experimentos

Se realizaron una serie de experimentos enfocados a determinar la correcta implementación de las motivaciones a la Red de Comportamiento, mismas que no existían anteriormente. Básicamente, fueron cuatro los experimentos, los cuales se explican brevemente a continuación.

Experimento 1
Modelo manejado por motivaciones

Red en la cual no existen ligas de ningún tipo. Se diseñó con el propósito de mostrar la forma en que los pesos agregados pueden influir sobre las motivaciones para lograr que el mono presente un patrón de conductas estable, que respete los tiempos que el mono real dedica a las conductas descanso, alimentación y locomoción. Las motivaciones respectivas son: cansancio, hambre e inquietud. Cada conducta está asociada con una precondición (siempre presente). Las motivaciones correspondientes deben ser tomadas en cuenta contínuamente como metas que el mono debe satisfacer.

Experimento 2
Modelo que emplea un mundo dinámico, sin modificación de las motivaciones

El mundo es dinámico y cambia con cierto intervalo de tiempo las proposiciones: comida_presente y sombra_presente por comida_ausente y sombra_ausente respectivamente (y viceversa). Se agregan las conductas buscar_comida y buscar_sombra. No se modifican los pesos asociados a las motivaciones, aún cuando sí se establecen los pesos iniciales de las mismas.

Experimento 3
Modelo que emplea un mundo dinámico, modificando las motivaciones

Combinación de los experimentos 1 y 2. Se establece un mundo dinámico para las tres conductas básicas y las dos de búsqueda. Después de la activación de cada módulo de comportamiento, se modifican los pesos asociados a las motivaciones de las conductas básicas. Resultado: el patrón de conductas ya no es uniforme y se conserva a pesar de las modificaciones de los pesos y de que el mundo está cambiando.

Experimento 4
Modelo que emplea un mundo dinámico con una generación aleatoria con distribución normal

La Red de Comportamiento es similar a la del experimento 3, pero estableciendo un mundo dinámico. Aquí se modifican los pesos y las condiciones de sombra. La comida aparece con una distribución normal con media de 100 y desviación estándar de 10; los pesos asociados a la motivación hambre se modifican incrementándolos si la comida es suficiente (más de 100 unidades de comida) o disminuyéndolos (100 o menos unidades de comida), con mayor o menor peso.

3.1.2. Alcance

Las Redes de Comportamiento son un mecanismo de selección de acción descentralizado y dinámicamente reconfigurable, donde la selección de acción es modelada como una propiedad emergente de las dinámicas de activación/inhibición entre los módulos de competencia. La Red de Comportamiento tiene las características de ser distribuida, recurrente y no jerárquica. Debido a su naturaleza distribuida, el sistema es tolerante a fallos; posee comportamiento orientado a metas, adaptabilidad, persistencia en el curso de acción, prevención de conflictos y rapidez.

Por sus características, este mecanismo ofrece grandes ventajas para la programación de agentes autónomos operando en ambientes dinámicos, mismos que se enfrentan a la toma de decisión de la próxima acción necesaria para cumplir múltiples metas variables en el tiempo de manera eficiente, es decir, enfrentando y superando cambios no previstos en el medio ambiente y en sus metas a alcanzar, o peor aún, fallas en sus sensores y en las acciones a aplicar.

Los resultados obtenidos con ABC han sido prometedores, por lo cual se ha considerado que es una herramienta de utilidad en la enseñanza de sistemas basados en el comportamiento, dado las facilidades que ofrece para la experimentación en el tiempo de ejecución, ofreciendo al usuario la visualización del comportamiento del agente simulado, y permitiéndole, a su criterio, manipular la definición del agente y/o del medio ambiente. En general, ABC ofrece grandes facilidades para

la construcción de agentes autónomos.

Queda de manifiesto que, mediante las Redes de Comportamiento, es posible modelar conductas animales reales observadas en la naturaleza. La complejidad en la estructura de una red de módulos conductuales y en la dinámica del ambiente, permiten explicar el surgimiento de patrones en el comportamiento animal observable en la naturaleza.

El modelado con el algoritmo de Maes permite ascender desde conductas muy elementales hasta comportamientos relativamente complejos que se derivan de la interacciones entre conductas más simples. Por tal motivo, los modelos de Redes de Comportamiento permiten estudiar la interacción entre un agente y un medio que puede ser dinámico.

3.2 Herramienta XLearn

Desde el punto de vista computacional, uno de los trabajos que motivaron esta investigación fue la implementación de una herramienta de simulación basada en aprendizaje denominada "XLearn: Un paquete para la simulación de Aprendizaje por Reforzamiento" [Mar00b], desarrollada en un curso de Aprendizaje impartido en la Maestría en Inteligencia Artificial de la Universidad Veracruzana. Dicho trabajo fue desarrollado por los alumnos bajo el asesoramiento del Dr. Manuel Martínez, como un paquete de apoyo en labores pedagógicas en relación al Aprendizaje Automático.

El objetivo de XLearn fue simular el comportamiento de una abeja en un jardín al seleccionar su alimento de entre varios tipos de flores, cada una con una probabilidad desconocida de proporcionar miel (recompensa). La abeja tenía permitido un número fijo n de visitas a cualquiera de las flores en su ambiente. La recompensa era binaria (tener o no miel). El objetivo era que la abeja obtuviera la mayor cantidad de miel durante las n visitas a las flores. Los algoritmos implementados en XLearn fueron: Greedy, ε-Greedy, Softmax, LR, LRP e Intervalos de Confianza; la simulación era estática, sin navegación de la abeja y la distribución de las flores siempre era la misma; las visitas eran determinadas por generación aleatoria entre el número de flores.

Como parte de los resultados se calculaba el factor XLearn [Mar00a], una medida de la eficiencia de los algoritmos, indicadora de la bondad en la estimación de la probabilidad de que la abeja encontrara miel en una flor. Este coeficiente se calculó, para cada tipo de flor f, con la siguiente fórmula:

$$XLearn_f = \frac{\frac{NumRecompensas}{NumVisitas}}{Probabilidad} \tag{3.1}$$

Donde:

$NumRecompensas$	Número de veces que hubo miel en flores tipo f.
$NumVisitas$	Número de visitas hechas a las flores f.
$Probabilidad$	Probabilidad de encontrar miel en una flor del tipo f.

Cuando el factor XLearn se aproxima más a 1, mejor es la estimación de la probabilidad. Este coeficiente da una medida de la capacidad del agente para discriminar entre los distintos estados del mundo.

La herramienta obtuvo buenos resultados y se estableció como un medio didáctico para una mayor comprensión del Aprendizaje por Reforzamiento, al permitir experimentar con diversos algoritmos ApR.

3.3 Estudios etológicos

Al igual que en las versiones previas de ABC, nuevamente se considera el trabajo etológico de [Rod94] con monos aulladores (*Alouatta Palliata*) de la isla de Agaltepec, del municipio de Catemaco, Veracruz. Rodríguez describe algunos aspectos de las estrategias de forrajeo de un grupo de estos primates bajo condiciones de semi-libertad, obteniendo datos estadísticos a partir de 1500 horas de observación. Además, presenta un modelo estocástico que simula el comportamiento del mono, describiendo tres conductas (descanso, alimentación y locomoción) y la forma en que cada una de ellas cambia a las otras, requiriendo como parámetros los tiempos promedio de permanencia en cada conducta y las probabilidades de transición (probabilidad de cambiar de una conducta a otra dependiendo de la presente) de la tropa de los monos aulladores.

Adicionalmente, este trabajo está basado en los estudios etológicos realizados por la bióloga Cristina Domingo Balcells [Dom04]. Domingo se enfoca de manera particular a la *plasticidad conductual* (capacidad de modular la conducta en función de las circunstancias del medio) de la misma especie de primates.

La plasticidad conductual se define como la potencialidad de emitir respuestas flexibles, modificar la conducta y por consiguiente, responder ante situaciones o condiciones cambiantes. De ahí su gran importancia para la adaptación de los animales ante los cambios del medio respecto a la disponibilidad y localización de las fuentes alimenticias y en general, del ambiente.

El objetivo de Domingo era analizar la respuesta conductual plástica de un grupo de monos aulladores ante variaciones en la disponibilidad de recursos alimenticios dentro de un espacio restringido.

El grupo de estudio fue una población de seis monos en cautiverio, en las instalaciones situadas en la estación biológica de Pipiapan, dentro del Parque de Flora y Fauna Silvestre Tropical (PAFFASIT), de la Universidad Veracruzana, localizada en la región de Los Tuxtlas, en el municipio de Catemaco, Veracruz (México).

El estudio consistió en someter al grupo a dos condiciones experimentales de distribución del alimento en el encierro:

- Fase control - se suministraron ramas frescas de especies silvestres colocadas en el centro del encierro y cuatro tipos de fruta cultivada presentadas en cuatro comederos de posición fija y equidistantes entre sí; cada comedero contenía las cuatro frutas. Duración: 180 horas.

- Fase experimental - se situó una única fruta por comedero al azar. Duración: 96,22 horas.

La cantidad total suministrada en ambas fases permanecía constante. El objetivo del experimento era identificar los alimentos preferidos.

Se predijo que el grupo de estudio ajustaría su comportamiento individual en función del cambio a corto plazo en la disposición del alimento. Tales ajustes conductuales comprenden cambios a nivel individual, como son: modificaciones del patrón de actividades, tiempo dedicado a la búsqueda y consumo de alimentos, locomoción y descanso, así como cambios en la dieta de los animales, en relación al tipo de alimento y cantidad consumida.

En sus estrategias de forrajeo, los monos aulladores presentan adaptaciones a la distribución espacial y temporal de los recursos alimenticios. Según explica Domingo, los herbívoros generalistas, como el individuo de estudio, pueden seleccionar un menú adecuado de plantas con la ayuda de sus habilidades de percepción y su experiencia previa con los alimentos; probablemente, la elección selectiva del alimento no se base únicamente en la calidad nutrimental del mismo, sino en otros factores, como la disponibilidad de los recursos, la capacidad de localizarlos y el grado de accesibilidad de los mismos.

La hipótesis del trabajo de Domingo era que "en respuesta a la distribución de alimento de alta calidad (fruta) suministrado a un grupo de monos aulladores en cautiverio, se producirán ajustes conductuales plásticos por parte de los animales".

Los ajustes que se esperaban a nivel individual eran cambios en el patrón de actividades (el tiempo dedicado a la búsqueda y consumo de alimentos, la locomoción y el descanso), así como cambios en la dieta, en relación al tipo de alimento y cantidad consumida por cada individuo.

En cuanto a los cambios en el patrón diario de actividades, se esperaba que variando aleatoriamente la ubicación espacial de las frutas, hubiera un aumento de las actividades de locomoción y búsqueda de alimento, dado que los individuos ya no podrían predecir la ubicación del alimento preferido, lo que les obligaría a realizar un mayor rastreo del área.

En relación a la alimentación, forrajeo y estructura espacial, se esperaba que durante la fase control, en que todas las estaciones alimenticias contendrían una mezcla de todos los alimentos, los animales tenderían a distribuirse ampliamente, de manera aleatoria, pero al variar al azar la posición de cada tipo de alimento en las estaciones alimenticias, se esperaba que ocurriera una exploración por parte del individuo hasta encontrar la ubicación del alimento de preferencia. Asimismo, se previó un acceso caótico al alimento inicial y un aprendizaje rápido de la posición espacial de los alimentos.

Estas hipótesis fueron corroboradas con el estudio etológico, teniendo como resultados la variación en la duración de la actividad de búsqueda de alimento, lo cual indica la selectividad de los individuos a la hora de alimentarse, es decir, los monos establecieron preferencias alimenticias. Otros datos obtenidos fueron en relación a las jerarquías grupales y la alimentación de los individuos: el mono de mayor jerarquía siempre comió el alimento de preferencia, debido a que mientras hubiera en existencia, siempre tuvo acceso a él. Por el contrario, los monos de

menor jerarquía, ante la inexistencia o no disponibilidad del alimento de preferencia, tuvieron que comer de las otras opciones alimenticias, observándose también la preferencia jerarquizada de alimentos.

La importancia de esta propuesta de investigación radica en que los estudios etológicos son difíciles de realizar, además de que toman mucho tiempo, pero una de las principales limitaciones de estos estudios es la disponibilidad del individuo de investigación, es decir, del mono aullador, ya que estos animales actualmente se encuentran en peligro de extinción, además de que con frecuencia enferman en cautiverio o presentan otro tipo de complicaciones a la adaptabilidad al encierro.

Por tanto, la simulación del comportamiento de estos animales es ideal para experimentar, siendo un apoyo a los estudios reales de los especialistas.

3.4 Mejoramiento propuesto

En el primer capítulo se planteó la construcción de arquitecturas de agentes que enfrentan el problema de la selección de acción, destacándose ahí un problema que pocas veces es atendido: aprender de la experiencia. El aprendizaje en base a la experiencia tiene como objetivo lograr que el agente sea capaz de mejorar su desempeño, por lo cual es algo deseable y necesario en el modelado de agentes que se aspira, tengan un comportamiento adaptativo y robusto.

Por tanto, hacer que un mecanismo de selección de acción se combine con aprendizaje es la propuesta de esta investigación. A diferencia de la forma en que otras arquitecturas puedan plantear el aprendizaje, aquí se propone aprender acerca de la toma de decisiones de una conducta, la cual forma parte del repertorio de conductas controladas por el mecanismo de selección de acción. Así, el agente a modelar cubre ambos aspectos, al seleccionar conductas de entre un repertorio mediante un mecanismo y aprender, por realimentación con el ambiente, a seleccionar acciones que sson de interés a una conducta.

Esta idea tuvo sus origenes al ver el modelado de abejas con la herramienta XLearn, pensando en hacer un módulo de aprendizaje parecido para los monos aulladores. Posteriormente, tomando en cuenta que la herramienta ABC en su última etapa habia simulado a este animal, se planeó entonces modelar la conducta de forrajeo mediante técnicas de aprendizaje, uniendo así los problemas de selección de acciones y aprendizaje.

Como no existe en la literatura algo similar, esta combinación conlleva el investigar los posibles efectos que el aprendizaje pueda tener en el mecanismo de selección de acciones.

De manera específica, se planea la introducción de un módulo de aprendizaje a la herramienta ABC, buscando que los agentes sean capaces de aprender en base a su experiencia en un medio ambiente dinámico, a fin de mejorar los resultados ya obtenidos con la herramienta. El aprendizaje que se utiliza es el Aprendizaje por Reforzamiento (ApR). El módulo de ApR se activará al estar el agente en una conducta específica, que implique toma de decisiones y sobre-

vivencia. Para este fin se ha considerado la implementación de varios algoritmos de Aprendizaje por Reforzamiento: Bandido, Greedy, ε-Greedy, Softmax, LRP, LR, Estimación de Intervalos y Comparación de Reforzamiento.

Con esto se busca dar seguimiento al trabajo iniciado por Guerra [Gue97c] y ampliado por Montes [Mon98]. Con la nueva extensión de ABC se pretende continuar el estudio del comportamiento de forrajeo del mono aullador, analizando los cambios que el aprendizaje genere en la Red de Comportamiento y buscando resultados similares al comportamiento del animal real.

Además, como ABC es una herramienta genérica, que implica el modelado de agentes, y en base a esto, toda una gama de simulaciones (ya sea robots inteligentes u otros animales), según se desee experimentar, la herramienta final con aprendizaje podrá ser aplicada a otros experimentos, cuidando únicamente la naturaleza del problema a plantear por aprendizaje.

Capítulo 4

Módulo de Aprendizaje por Reforzamiento con Redes de Comportamiento

Este capítulo está dedicado a describir la implementación de la propuesta de investigación: la elaboración de un módulo de Aprendizaje por Reforzamiento actuando sobre un módulo de comportamiento particular perteneciente a un Mecanismo de Selección de Acción, constituido por las Redes de Comportamiento (la nueva herramienta ABC_ApR). Por tanto, se presenta un esquema general de la interacción de la Red de Comportamiento con el Aprendizaje por Reforzamiento, para posteriormente exponer la implementación de los algoritmos ApR, así como las variantes ambientales en función a la cantidad y distribución de las posibles acciones a ejecutar.

4.1 Red de Comportamiento y Aprendizaje por Reforzamiento

El modelo que se propone en este trabajo enlaza al mecanismo de selección de Redes de Comportamiento con el Aprendizaje por Reforzamiento mediante un subsistema adicional a la red que se activa cada vez que se selecciona el módulo conductual sobre el que se desea aprender, m_{ApR}. El aprendizaje actúa como una especialización de selección de acción del comportamiento a aprender. La gráfica 4.1 presenta el esquema general de la forma en que interactúan ambos procesos de selección de acción.

Cuando el módulo de aprendizaje se activa, se seleccionará una opción op_i $(1 \leq i \leq N)$ mediante el mecanismo propio del algoritmo de aprendizaje que se esté aplicando. Estas opciones implican diferentes niveles de recompensa para el agente. La meta deseada es que, a medida que se aplique el aprendizaje, se identifique la opción que maximice la ganancia en la conducta m_{ApR} del agente modelado por la Red de Comportamiento.

Para ejemplificar la estructura de la herramienta ABC_ApR, considérese el diseño de una Red de Comportamiento que simule el comportamiento de un animal, por ejemplo, un mono. Los módulos conductuales m_i que definen al agente son las actividades de éste animal: descansar, explorar, alimentar, socializar, huir de depredadores, etc.; y, en concordancia a los experimentos realizados con ABC_ApR, relativos al forrajeo de los monos aulladores, sea m_{ApR} = alimentar la conducta sobre la que se aplique el aprendizaje.

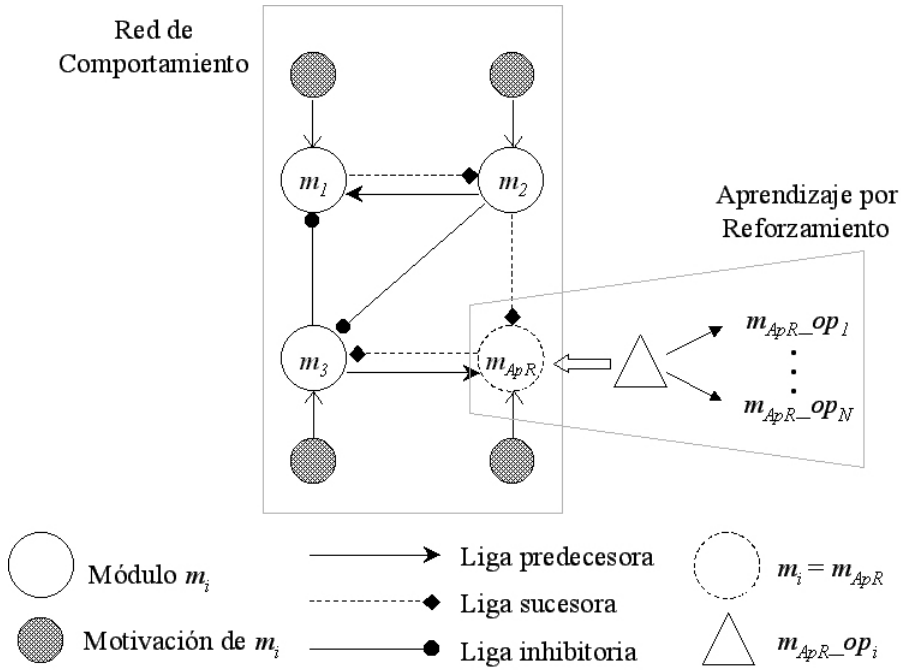

Figura 4.1: Modelo de Red de Comportamiento con Aprendizaje por Reforzamiento

Para alimentarse, el animal tiene opciones alimenticias -varios tipos de frutas y hojas- de calidad diversa, de entre las cuales existe un alimento preferencial (el de mayor calidad), según lo determinado por estudios etológicos (ver [Dom04] y sección 3.3). El animal determina el alimento preferencial en base a su experiencia en la ingesta de las diversas opciones alimenticias y posteriormente realiza una elección selectiva orientada a éste alimento. En términos de ABC_ApR, los diferentes tipos de alimentos conforman las opciones op_i y el alimento de preferencia es la opción óptima, $op*$, que posee la máxima probabilidad de recompensa para la conducta m_{ApR} del agente.

4.2 Módulo de aprendizaje

Para implementar el aprendizaje se optó por el Aprendizaje por Reforzamiento, dadas las características que posee y que lo hacen ideal para el tipo de problema que se enfrenta: un agente que se encuentra en un ambiente desconocido, cuyo objetivo es seleccionar, de entre varias opciones, la mejor, a fin de maximizar su recompensa total durante toda su vida.

A continuación se plantean las generalidades en torno al módulo de aprendizaje implementado:

- El aprendizaje actúa sobre una conducta específica m_{ApR} de la Red de Comportamiento que modela al agente simulado. Es deseable que esta conducta implique la optimización de selección entre sus opciones op_i.

- Los algoritmos implementados son Bandido, Greedy, ε-Greedy, Softmax, LRP, LR, Estimación de Intervalos y Comparación de Reforzamiento.

- Todos los algoritmos trabajan bajo el modelo del algoritmo general de aprendizaje, contando con una función de opción, una función de recompensa y una función de evaluación de acción, manteniendo actualizados los datos necesarios en cada iteración. La función de opción corresponde a la función de selección de acción del algoritmo mencionado (algoritmo 2, página 29)

- Todos los algoritmos trabajan bajo un marco común: el problema del Bandido, en su versión generalizada de N acciones. El usuario determina el valor de N.

- Las recompensas o valoraciones de la deseabilidad de las opciones elegidas, op_i, que el ambiente otorga al agente son de tipo estocástico, de acuerdo a una probabilidad asociada a cada opción al momento de definir la corrida de ABC_ApR. Las señales de reforzamiento generadas son booleanas $\{0, 1\}$.

- El módulo de aprendizaje cuenta con un ambiente de simulación 2-D, con el cual el agente interactúa confrontando las opciones a elegir. Cada vez que el módulo entra en actividad, el ambiente muestra la forma en que el agente adopta las opciones op_i, reflejando así sus decisiones para la conducta m_{ApR}.

- Desde sus versiones anteriores, ABC ha tenido un ambiente de tipo dinámico, determinado por las dinámicas de activación/inhibición entre los módulos de competencia de la Red de Comportamiento. Además de esto, ABC_ApR añade al ambiente un dinamismo propio de la distribución de las opciones op_i sujetas a aprendizaje. Así, el ambiente cambia cada 20 iteraciones de aprendizaje. En términos del experimento simulado del mono, ésta dinámica implica la redistribución del alimento en el mundo.

- Cada vez que una opción ha sido seleccionada por el algoritmo, antes de calcular sus recompensas real y esperada, el agente debe ver la disponibilidad de tal opción, por lo que debe realizar una búsqueda en el ambiente simulado. Si la opción seleccionada no se puede encontrar en el ambiente, debe tomar alguna otra existente en el mundo (ver sección 4.2.3).

- Para fines comparativos, se implementaron dos mecanismos bajo los que el agente selecciona acciones en su ambiente. El primero de ellos es el caso paraíso, donde el ambiente del agente únicamente contiene las situaciones que definen la opción óptima, op^*, por lo que sólo ésta puede ser percibida y, por tanto, seleccionada. El segundo es el caso sin aprendizaje, donde el agente puede percibir todas las opciones op_i en su ambiente, pero seleccionará la primera opción que detecte en su mundo, pues no se aplica ningún mecanismo de selección de opción y por tanto, no se aprende de lo experimentado.

- El módulo corre en función a la herramienta ABC, por lo que no es posible determinar el total de iteraciones que tendrá; éste surge de las motivaciones del agente en la Red de Comportamiento; el criterio de parada global son 1500 iteraciones de ABC.

4.2.1. Algoritmos de Aprendizaje por Reforzamiento

En el capítulo 2 se analizó lo relativo al Aprendizaje por Reforzamiento y se presentaron diversas técnicas a las que pertenecen los algoritmos usados en este trabajo. Antes de exponer las implementaciones de los algoritmos de aprendizaje de los que consta el módulo, se debe considerar la notación usada en las siguientes secciones:

Sea un agente A, $A=\{m_1, m_2, ..., m_n\}$, donde m_i $(1 \leq i \leq n)$ es un módulo de competencia de la Red de Comportamiento que modela al agente.

Sea $m_{ApR} \in A$, indicado por el usuario al diseñar la red para que sea objeto de aprendizaje. Para el módulo de competencia $m_{ApR} = (c_{ApR}, a_{ApR}, d_{ApR}, \alpha_{ApR})$ existe un conjunto de opciones OP del que m_{ApR} debe seleccionar una opción op_i al ser activado por la red. Esta selección de opción depende del algoritmo de aprendizaje aplicado.

$$OP = \{op_1, ..., op_N\}$$

$$\text{selección\_op(OP)} \rightarrow op$$

$$r_{op} = \begin{cases} 1 & \\ & \text{con probabilidad } PR(op),\ op = op_i \\ 0 & \end{cases}$$

N	Número de opciones del módulo m_{ApR}.
op	Opción seleccionada actualmente.
m_{ApR}_op	Opción ejecutada por m_{ApR}.
$PR(op_i)$	Probabilidad de recompensa de la opción op_i.
r_{op}	Recompensa recibida actualmente al ejecutar una opción op.
$R(op_i)$	Recompensas que ha dado la opción op_i.
$A(op_i)$	Número de veces que se ha seleccionado la opción op_i.
$Q(op_i)$	Valor estimado de la opción op_i.
$P(op_i)$	Probabilidad de seleccionar la opción op_i.

Se prescinde de usar el indicativo de tiempo t, debido a que todos los cálculos son realizados con los datos actuales en t.

4.2.1.1. Algoritmo del Bandido

De acuerdo a la naturaleza del algoritmo, como se explicó en la sección 2.7.1, el algoritmo cuenta con una etapa de predecisión, de tamaño k. Este algoritmo trabaja sobre las recompensas totales obtenidas, no sobre la recompensa estimada. Se usa la notación m y j para indicar que las opciones sobre las que se realiza la comparación son cualesquiera de la gama de opciones disponibles ordenadas en decremento por su recompensa recibida ($R_{op_m} > R_{op_n} > ... > R_{op_j}$).

En etapa de predecisión:

Prueba todas las opciones en orden y verifica en cada momento que, ordenadas de mayor a menor por sus recompensas totales obtenidas hasta ese momento, estén en etapa de predecisión: ($|R_{max}(op_m) - R_{min}(op_j)| < k$). En el momento en que $|R_{max}(op_m) - R_{min}(op_j)| >= k$ y existan más de dos opciones en etapa de predecisión, elimina a la opción op_j con la mínima recompensa recibida $R_{min}(op_j)$.

En etapa de decisión:

Cuando la comparación $|R_{max}(op_m) - R_{min}(op_j)| >= k$ únicamente se realiza entre dos opciones (ya se han eliminados las otras), se toma la decisión de que por siempre se seleccionará la opción con la máxima recompensa recibida $R_{max}(op_m)$:

$$op = R_{max}(op_m)$$

Después de que se ha seleccionado una opción, se obtiene su recompensa mediante $PR(op)$ y para fines de presentación de resultados, se calcula su recompensa esperada (ver ecuación 4.1).

4.2.1.2. Algoritmo Greedy

La selección de acción se da en dos etapas:

Etapa de predecisión

Al iniciar la corrida, el algoritmo cumple una etapa de predecisión, la cual consiste de probar todas las acciones en orden ($op_a, op_b, .., op_N$), hasta completar un cierto número k (según sea necesario, se reinicia el ciclo de prueba: op_m, op_N, op_a, op_b,...). Esto con el fin de recabar información acerca de las opciones. Por tal motivo, la selección de las opciones es determinada por el orden que posean.

Etapa codiciosa

Una vez que se ha superado la etapa de predecisión, el método Greedy entra en actividad. Realiza la selección de una acción de manera codiciosa, es decir, elige la opción que posea el valor de estimación más alto, para lo cual únicamente busca la mayor recompensa acumulada:

$$op = max_{op} Q(op_i)$$

En caso de haber más de una opción que tenga el máximo valor, se elige aleatoriamente entre ellas:

$$op = random(\{op_i \mid Q(op_i) = Q(op_{j \neq i})\})$$

Posterior a la selección de acción se obtiene la recompensa de la opción elegida, r_{op}. Finalmente, se calcula la recompensa esperada de la opción ejecutada, mediante la ecuación 4.1. La estimación del valor de la opción, o recompensa esperada, es calculada mediante el promedio de la muestra de las recompensas observadas, en su implementación incremental. Así, para cuando una opción $op = op_i$ se ha ejecutado, recibiendo una recompensa r_{op}, se calcula su valor mediante:

$$Q(op_i) = Q(op_i) + \frac{r_{op} - Q(op_i)}{A(op_i)} \tag{4.1}$$

La implementación incremental se debe a la conveniencia de no ir guardando la información relativa a todas las recompensas recibidas hasta un tiempo t.

4.2.1.3. Algoritmo ε-Greedy

Selecciona la mayor parte del tiempo codiciosamente, pero con una baja probabilidad ε se dedica a explorar las demás alternativas, de manera aleatoria, uniformemente. El valor del parámetro ε es proporcionado por el usuario al inicio de la corrida.

Con probabilidad ε
$op = random(\{op_i \mid 1 \leq i \leq N\})$

Caso contrario
$op = max_{op} Q(op_i)$

Posterior a la selección de acción, obtiene la recompensa que la opción elegida le otorga, en base a lo que el ambiente determine por medio de la probabilidad de recompensa asociada a la opción, $PR(op)$. Finalmente, se calcula el valor estimado o recompensa promedio, que es mediante el promedio incremental de la muestra (ecuación 4.1).

4.2.1.4. Algoritmo Softmax

Debido a que este algoritmo pertenece a las estrategias aleatorias, se dedica una parte del tiempo a actividades exploratorias, pero a diferencia de el algoritmo ε-Greedy, la selección aleatoria no es uniforme, sino mediante las probabilidades obtenidas por la distribución de Bolztmann, cuya ecuación (ya presentada en 2.1) es utilizada de la siguiente manera:

Con probabilidad ε
$op = op_i$ con

$$P(op_i) = \frac{e^{Q(op_i)/T}}{\sum_{j \neq i} e^{Q(op_j)/T}}$$

Caso contrario
$op = max_{op} Q(op_i)$

El parámetro T es conocido como *Temperatura* y es el que controla el grado de exploración, pues a grandes valores provoca que exista más exploración del ambiente, pero a medida que tiende a 0 la estrategia de selección de acción iguala a la estrategia codiciosa.

Una vez que se ha tomado la decisión de qué opción probar, se obtiene la recompensa por haber ejecutado esa opción. Al igual que con los tres algoritmos previos, se calcula la recompensa esperada mediante la ecuación 4.1.

4.2.1.5. Algoritmo LRP

Este algoritmo se enfoca a la representación de la deseabilidad de una acción mediante un vector de probabilidades (denotado por Q). Cuenta con una etapa de predecisión en la cual se prueban todas las opciones con el fin de recabar información relativa a cada una, obteniendo su recompensa r_{op} y realizando los ajustes al vector de probabilidades de la siguiente manera:

Para $op = op_i$,

si $r_{op} = 1$:

$$Q(op_{j\neq i}) = (1-\alpha) * Q(op_j)$$
$$Q(op_i) = Q(op_i) + \alpha(1 - Q(op_i))$$

si $r_{op} = 0$:

$$Q(op_{j\neq i}) = \frac{\beta}{N-1} + (1-\beta) * Q(op_j)$$
$$Q(op_i) = (1-\beta) * Q(op_i)$$

Fuera de la etapa de predecisión, la política de selección de acción es establecida aleatoriamente de acuerdo a las probabilidades que posee cada opción, por tanto

> Selecciona una opción op de acuerdo a el vector de probabilidades:
> $op = op_i$ con probabilidad $Q(op_i)$

El algoritmo es una de las versiones que existen (tomado de [Uns97]) y funciona para N acciones.

4.2.1.6. Algoritmo LR

Cuenta con dos etapas, una etapa de predecisión para probar todas las acciones y una de aplicación de una política aleatoria de selección de acción:

> Selecciona una opción op considerando su probabilidad en el vector Q:
> $op = op_i$ con probabilidad $Q(op_i)$

Posterior a la elección de una opción op se obtiene su recompensa r_{op} y para la actualización del vector de probabilidades, a diferencia de LRP, únicamente se ajusta el vector ante opciones exitosas, sin penalizar en ningún sentido por el hecho de que no se generen recompensas ($r_{op} = 0$). El vector es actualizado de la siguiente manera:

Sea $op = op_i$ y $r_{op} = 1$:

$$Q(op_{j\neq i}) = (1-\alpha) * Q(op_j)$$
$$Q(op_i) = Q(op_i) + \alpha(1 - Q(op_i))$$

4.2.1.7. Algoritmo RC

Esta técnica hace uso de una recompensa de referencia, rp, y de una medida de preferencia de cada opción, $Pr(op_i)$, a fin de calcular las probabilidades $Q(op_i)$ de selección de las acciones.

rp - Promedio incremental de todas las recompensas recibidas.

$$rp = rp + \alpha(r_{op} - rp)$$

$Pr(op_i)$ - Preferencia de la opción seleccionada $op = op_i$.

$$Pr(op_i) = Pr(op_i) + \beta(r_{op} - rp)$$

Calcula la probabilidad de seleccionar la opción op_i, $Q(op_i)$, mediante:

$$Q(op_i) = \frac{e^{Pr(op_i)}}{\sum_{n=1}^{N} e^{Pr(op_n)}}$$

Al principio aplica un proceso de recolecta de información, mediante la selección ordenada de las opciones. Posterior a esa etapa, aplica una política ε-greedy, decrementando ε con el transcurso del tiempo, a fin de seleccionar la mejor acción la mayor parte del tiempo.

4.2.1.8. Algoritmo IE

Inicialmente el algoritmo entra en una etapa de recolección de información, probando cada una de las opciones op_i. Posterior a esto, cada vez que se va a seleccionar una acción aplica una política ε-greedy, con un valor de ε grande que es decrementado con el tiempo, para favorecer a la política greedy.

Cada vez que se ejecuta una opción op se obtiene la recompensa r_{op} de acuerdo a su probabilidad de recompensa asociada. Como $op = op_i$, se analiza la deseabilidad de op_i, mediante el cálculo del límite superior del intervalo de confianza, y se almacena en $Q(op_i)$, que representa la probabilidad de seleccionar a la opción op_i. Los límites superiores del intervalo de confianza son obtenidos mediante la ecuación 2.2, que nuevamente se presenta a continuación:

$$Q(op_i) = \frac{\frac{R(op_i)}{N(op_i)} + \frac{z^2}{2N(op_i)} + \frac{z}{\sqrt{N(op_i)}}\sqrt{\left(\frac{R(op_i)}{N(op_i)}\right)\left(1 - \frac{R(op_i)}{N(op_i)}\right) + \frac{z^2}{4N(op_i)}}}{1 + \frac{z^2}{N(op_i)}}$$

4.2.1.9. Comparativos

Selección óptima

El ambiente sólo cuenta con la opción óptima, $op*$, la de máxima probabilidad de recompensa, $op* = max_{op}PR(op_i)$. Por tanto, el agente siempre la percibe y selecciona, lo cual representa el mundo ideal, o paraíso. El agente no toma decisiones sobre el tipo de opción que debe tomar, únicamente ejecuta la que detecta primero en su mundo.

A pesar de que con este ambiente óptimo el agente siempre selecciona la mejor de todas las acciones posibles que los demás algoritmos buscan, la recompensa puede o no ser satisfactoria (debido a la aleatoriedad). Para fines de obtener resultados comparativos, se calcula la recompensa esperada mediante la ecuación 4.1.

Selección sin aprendizaje

El ambiente contiene todos los tipos de opciones que m_{ApR} puede ejecutar, pero no se realiza ninguna selección de acción mediante técnicas de aprendizaje; por el contrario, el agente únicamente se limita a que, una vez que la Red de Comportamiento activa el módulo m_{ApR}, opta por seleccionar la opción op_i que detecta primero en el ambiente, sin considerar la posible ganancia que le pueda redituar (si es la mejor o la peor), o si existe otra opción op_j que puede ejecutar y que era mejor que la que ha seleccionado.

Al igual que todos los algoritmos, una vez ejecutada la opción, se obtiene su recompensa por medio de $PR(op_i)$ y se calcula la recompensa promedio con la ecuación 4.1.

4.2.2. Ambiente

El módulo de aprendizaje consta de un ambiente de dos dimensiones anexado a la derecha de la pantalla principal de la herramienta ABC, como se muestra en la figura 4.2.

El agente está representado por un círculo grande color café, en tanto que los diversos tipos de opciones de m_{ApR} se representan por pequeños círculos de colores (amarillo, naranja, verde o rojo, según sea el caso).

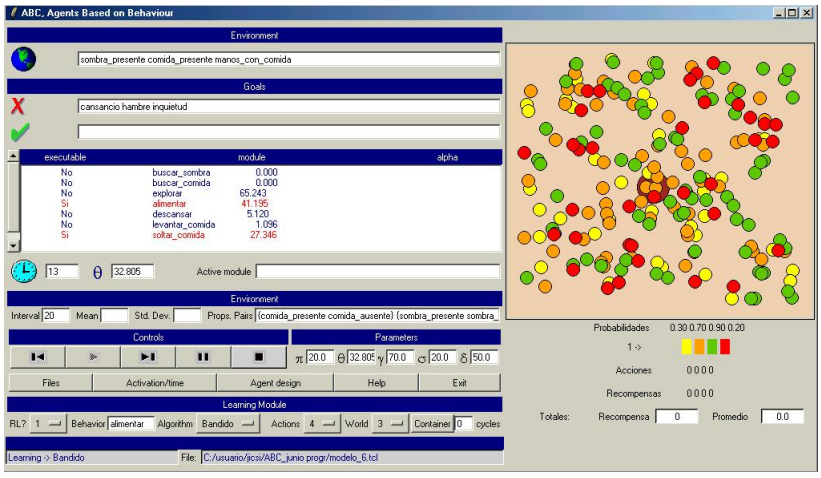

Figura 4.2: Ambiente del módulo de aprendizaje de ABC

Al comienzo, el agente se ubica en el centro de la pantalla que representa al ambiente, para iniciar su navegación de acuerdo a los requerimientos del módulo de aprendizaje. Las opciones op_i se distribuyen en el ambiente en posiciones aleatorias, de acuerdo al mundo seleccionado (ver sección 4.4).

En la parte inferior externa del ambiente simulado se presentan diversos datos de las acciones: sus probabilidades de recompensa, su color de identificación, la opción seleccionada y obtenida,

el número de veces que se ha seleccionado cada opción op_i y la recompensa generada, así como la recompensa acumulada y la recompensa promedio total, junto con los valores de los parámetros en caso de que el algoritmo requiera alguno.

Cuando el módulo m_{ApR} es activado por la Red de Comportamiento, se ejecuta el módulo de aprendizaje y en el ambiente, el agente inmediatamente busca a su alrededor la opción *op* que le indica su aprendizaje (o la más próxima, en cualquiera de los dos comparativos). Para esto, el agente cuenta con una zona de percepción dentro de la que detecta las opciones a su alrededor; al detectar la opción de su elección, modifica su dirección de navegación y se dirige hacia ella y una vez que está a su alcance (dentro de su área circular), la ejecuta, con lo cual desaparece del ambiente.

En caso de que la opción elegida no se no encuentre a su alcance, el agente navega por el ambiente en busca de ella, siguiendo la dirección que tenía antes de requerir la búsqueda, hasta encontrarla o modificar su selección (ver sección 4.2.3). Al llegar a los límites del mundo, el agente gira en direcciones aleatorias para continuar su navegación.

El ambiente simulado de ABC_ApR presenta una dinámica al cambiar cada 20 iteraciones de aprendizaje la posición de las opciones op_i, en forma aleatoria. En tanto no se dispare el módulo de aprendizaje, el agente navega aleatoriamente por el ambiente.

4.2.3. Impacto del ambiente en la selección de acciones

Cada vez que el agente ha seleccionado una opción de acuerdo a la naturaleza del algoritmo de aprendizaje en ejecución, es necesario corroborar la existencia de situaciones donde sea posible percibir la opción deseada, *op*, en el ambiente, para lo cual, mediante navegación, se ejecuta su búsqueda. En el caso ideal, la opción *op* será percibida y el agente procederá a ejecutarla, realizando posteriormente los ajustes de la deseabilidad o conveniencia de ese tipo de acción.

Por el contrario, puede darse el caso de que, si durante un cierto tiempo le fuera imposible al agente encontrar la opción elegida, entonces deberá seleccionar una opción de otro tipo para su ejecución, ante la escasez de situaciones donde se perciba la opción deseada.

Al suceder lo anterior, el agente tiene que modificar su selección de acción, desechando lo que el aprendizaje le indicó. Así, el agente seleccionará en orden jerarquizado (por recompensa) de entre los tipos de opciones diferentes a la opción deseada, o bien, de acuerdo a la opción que tenga más cercana. Una vez que se modifica la selección de acción, se inicia la búsqueda de la nueva opción, hasta que una opción es ejecutada. Posterior a esto, se procede a ver la recompensa que otorga el nuevo tipo de opción y se actualiza el aprendizaje.

Con esto, se evita que el agente caiga en un bloqueo permanente en espera de que se perciba una opción inexistente en el ambiente.

El tiempo de búsqueda de la opción $tiempo_{busqueda\_op}$ es considerado como parte del desempeño del agente durante el aprendizaje. La manera en que se registra es la siguiente:

$tiempo_{busqueda\_op} = 1$	Si el agente ejecuta la opción indicada por el módulo de aprendizaje
$tiempo_{busqueda\_op} += 1$	Cada vez que el agente no encuentra situaciones que le permitan ejecutar un cierto tipo de opción y se ve en la necesidad de buscar una opción diferente.

4.3 Impacto del aprendizaje en la Red de Comportamiento

El resultado obtenido por el aprendizaje actuando sobre una conducta específica de la Red de Comportamiento tiene que ser tomado en cuenta cada vez que el módulo de aprendizaje concluye una iteración. La forma en que se planeó la afectación a la Red de Comportamiento fue en relación al nivel de activación α de la conducta m_{ApR} sobre la que se modela el aprendizaje. El algoritmo 3 muestra las modificaciones al nivel de activación en la Red de Comportamiento cuando hay aprendizaje y cuando no lo hay.

Algoritmo 3 Enlace de Aprendizaje y Red de Comportamiento

```
1: loop
2:     Red de Comportamiento ⇒ m_i                          ▷ Selecciona m_i
3:     if m_i = m_ApR then
4:         Aprendizaje por Reforzamiento ⇒ op                ▷ Selecciona op y obtiene r_op
5:         if op = op* & r_op = 1 then
6:             m_ApR(α) = 0
7:         else if r_op = 1 then
8:             m_ApR(α) = m_ApR(α) * 0,5 * tiempo_busqueda_op
9:             intervalo_dinamica = X
10:        else
11:            m_ApR(α) = m_ApR(α) * 1,5 * tiempo_busqueda_op
12:            intervalo_dinamica = Y
13:        end if
14:    else
15:        m_i(α) = 0                                        ▷ Por defecto
16:    end if
17: end loop
```

Cuando no se aplica aprendizaje, la Red de Comportamiento aplica a sus módulos seleccionados un restablecimiento a cero en sus niveles de activación, para que reinicien el acumulamiento de energía a fin de entrar nuevamente en competencia con los demás módulos.

Cuando se aplica aprendizaje, al hacer la modificación del nivel de activación se consideran el tipo de opción seleccionada durante el aprendizaje, la recompensa obtenida y el tiempo utilizado en hallar la opción. El ajuste a α puede ser de tres tipos:

- Si se seleccionó la opción óptima, $op*$, y se obtuvo recompensa, aplica el decremento común de la Red de Comportamiento, es decir, el nivel de activación de m_{ApR} es cero.
- Si la opción seleccionada no es la óptima ($op \neq op*$), pero se obtuvo recompensa, el nivel de activación de m_{ApR} es decrementado en un 50 % y multiplicado por el tiempo que le

tomó encontrar la opción indicada por el módulo. Además, se modifica el intervalo que controla la dinámica del mundo (lapso de tiempo en que se cambian las proposiciones variables en el mundo, en la Red de Comportamiento).

- En caso de que la opción, óptima o no-óptima, no haya obtenido recompensa ($r_{op} = 0$), el nivel de activación es incrementado en un 50 % de su valor además de multiplicarlo por el tiempo de búsqueda que empleó en hallar la opción. Modifica también el valor del intervalo en el que se va a cambiar el mundo.

Cuando la opción solicitada por el módulo de aprendizaje es encontrada (sin necesidad de cambios por no percibirla), el tiempo de búsqueda no afecta.

La modificación del valor del intervalo con que se cambian las proposiciones variables en el mundo introduce más dinámica en las Redes de Comportamiento y, con esto, se acentúa el impacto del aprendizaje en la Red de Comportamiento modelada. La cantidad en que se varía el intervalo es constante, únicamente se determinó que $Y > X$, para ejercer penalización cuando no se recibe recompensa.

Estos ajustes indican que el obtener recompensa es lo correcto y que las opciones óptimas son las deseadas; por tanto, ejerce cierto castigo al elegir una opción no-óptima y al no recibir recompensa, y aún más, al emplear mucho tiempo en hallar una de ellas. En estos casos, el nivel de activación de m_{ApR} refleja la insatisfacción al aprender y la urgencia por seleccionar al módulo m_{ApR} en el corto plazo.

4.4 Variaciones ambientales

Una vez que se implementaron los algoritmos de aprendizaje y los comparativos, se buscó tener diferentes marcos de experimentación de los mismos, por lo que se optó por hacer variaciones en relación a la cantidad total de las diversas opciones op_i que forman parte del ambiente del agente y que éste puede percibir, así como en la forma en que tales opciones ocupan una posición dentro del ambiente simulado.

En el último caso, que en adelante se denominará variación por *distribución*, las opciones *op* serán ubicadas ya sea en cualquier lugar del ambiente, o bien, en zonas específicas del mismo (lo cual tiene inspiración etológica en un intento por emular parches alimenticios). En tanto, en la variante por *cantidad*, el ambiente tendrá escasez o abundancia de las opciones *op*.

4.4.1. Cantidad de acciones

Se pensó en hacer que el agente se enfrentara a situaciones donde hubiera opciones en abundancia en el ambiente, así como donde hubiera escasez de ellas. Por tal motivo se hizo variar la cantidad de toda la gama de opciones *op* que el agente podría percibir en su mundo al momento de seleccionar una opción para el módulo m_{ApR}, lo que da como resultado que existan ambientes de tipo escaso (con pocas opciones) y ambientes de tipo abundante (con muchas opciones).

4.4.1.1. Ambiente escaso

Bajo este mundo, las situaciones donde el agente pueda percibir las opciones están limitadas, debido a que es mínima la cantidad de opciones que forman parte del ambiente. Con este tipo de mundo se planea estudiar el comportamiento del agente ante la falta de opciones. La figura 4.3 ejemplifica la cantidad de opciones accesibles para el agente en un ambiente de este tipo.

4.4.1.2. Ambiente abundante

Representa un ambiente al que pertenecen más opciones en comparación a un ambiente escaso, es decir, donde hay abundancia de opciones; se espera que este tipo de mundo contribuya a que el agente maximice su recompensa promedio al poder percibir y por tanto, ejecutar, las opciones que desea. La figura 4.4 representa la forma en que el agente tiene grandes posibilidades de percibir opciones en este tipo de ambiente.

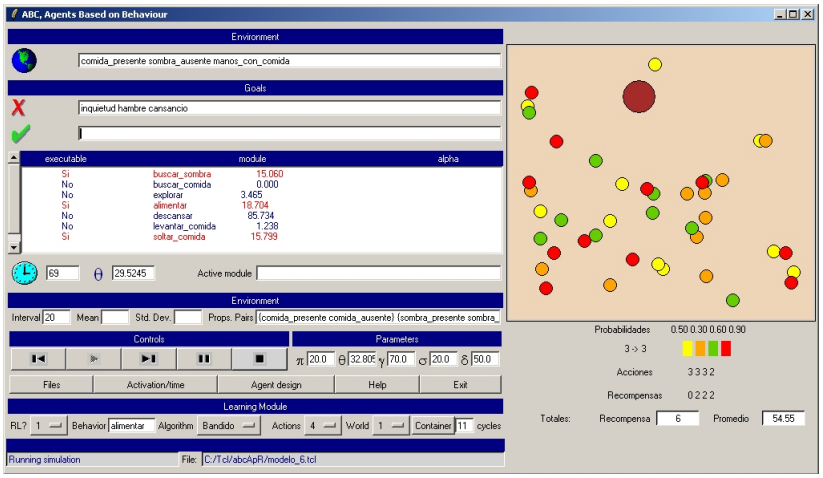

Figura 4.3: Mundo de distribución aleatoria con cantidad escasa

4.4.2. Distribución de acciones

Considerando los experimentos realizados con la herramienta ABC_ApR, surge esta variación del ambiente que rodea al agente, misma que comprende las distintas formas en que las opciones *op* que satisfacen al módulo m_{ApR} ocupan una posición dentro del mundo simulado. Por un lado, en semejanza a la distribución de los alimentos en la naturaleza, existe una distribución de opciones en posiciones aleatorias de todo el ambiente; por otra parte, en una simulación de experimentos etológicos, se considera que el alimento puede ser distribuido por medio de parches (lugares específicos donde los diversos alimentos son colocados), lo que mapeado a la distribución de las opciones resulta en que éstas sean ubicadas únicamente en ciertas zonas del ambiente simulado, donde tienden a acumularse.

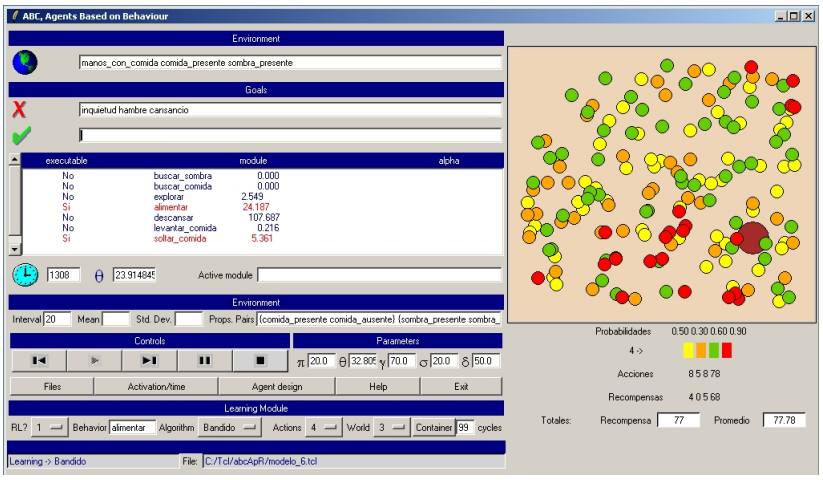

Figura 4.4: Mundo de distribución aleatoria con cantidad abundante

4.4.2.1. Aleatoria

Es el caso más sencillo de la ubicación de las opciones en el ambiente, consistiendo simplemente de poner cada opción en una posición al azar, dentro del mundo simulado que rodea al agente. Las figuras presentadas en la sección anterior (4.4.1) muestran gráficamente la disposición de las opciones bajo este criterio, presentando los casos de un ambiente escaso (figura 4.3) y uno abundante (figura 4.4).

4.4.2.2. Por zonas

Esta variación en la distribución de opciones se hizo a fin de que en los experimentos realizados con la herramienta implementada se pudiera hacer una simulación de estudios etológicos con parches alimenticios.

Bajo la inspiración del trabajo de Domingo [Dom04], (reseñado en la sección 3.3), se presentan las opciones en tres zonas, definidas perfectamente en el mundo; las opciones son generadas de acuerdo a dos fases:

- Fase primaria- Las tres zonas contienen todos los tipos de opciones.
- Fase secundaria - Sólo una zona presenta la opción óptima, en tanto que las otras dos presentan un surtido de las opciones restantes (a diferencia del experimento de Domingo, en el que cada parche contenía un único tipo de alimento, lo cual se cumple en parte para el caso de dos opciones, $N = 2$).

Al inicio, se presentan las zonas en fase primaria, para que el agente pueda tener acceso en un mismo lugar a todos los tipos de opciones posibles. Posteriormente se distribuyen las opciones en la fase secundaria, que se presenta cada vez que sea necesario, dependiendo de las iteraciones del módulo de aprendizaje.

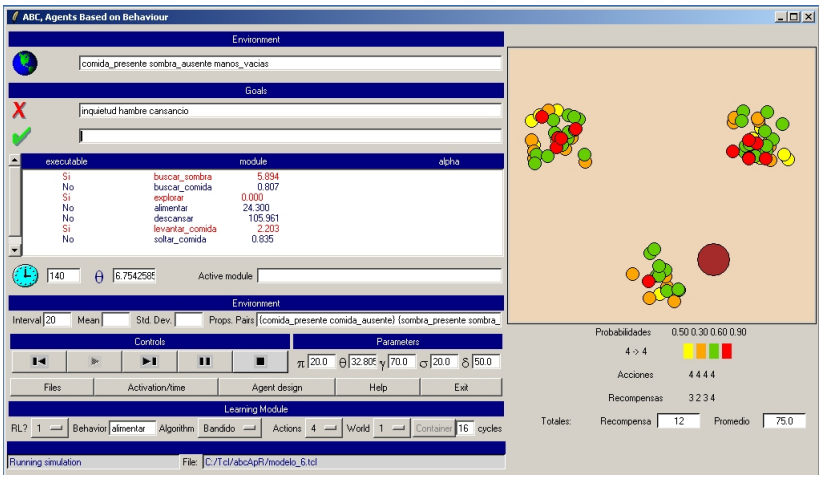

Figura 4.5: Mundo de distribución por zonas con cantidad escasa

La figura 4.5 muestra el mundo del agente en su versión "por zonas", para un caso escaso y en fase primaria, pues todas las zonas contienen una mezcla de todo tipo de opción. Por su parte, la figura 4.6 presenta el ambiente de zonas con cantidad abundante de opciones, en una fase secundaria, donde una de los zonas (determinada aleatoriamente) posee únicamente la opción óptima.

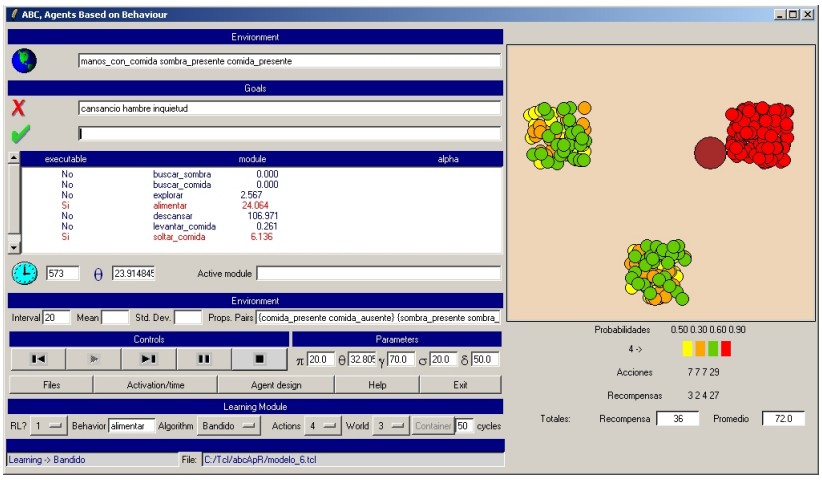

Figura 4.6: Mundo de distribución por zonas con cantidad abundante

4.4.3. Datos y gráficas obtenidas

Al ejecutar la herramienta ABC se obtienen diversos datos y gráficas, que muestran el desempeño de la Red de Comportamiento y del módulo de aprendizaje. En relación a la Red de

Comportamiento, se genera una gráfica de activación en el tiempo, que muestra cómo se fueron seleccionando los módulos y durante cuánto tiempo estuvieron activos, así como su acumulamiento de energía de activación (sección 3.1). También se obtienen el porcentaje, el tiempo promedio y la matriz de probabilidades de transición para cada módulo de la Red de Comportamiento (sección 3.1.1). Además, se guardan en archivo los datos para poder generar una gráfica de la proporción de tiempo dedicada a cada conducta, por bloques del total de ciclos de ABC (tomado de estudios etológicos, donde se conoce como la gráfica del Patrón Diario de Actividades, PDA).

Como resultados del aprendizaje, se presentan las curvas de aprendizaje y de convergencia, las cuales representan el desempeño de los algoritmos en cuanto a la recompensa total obtenida y a la ejecución de la opción óptima. La curva de aprendizaje es una gráfica de los valores de reforzamiento esperado contra el tiempo, que muestra una tasa de mejoría del desempeño. Por su parte, la curva de convergencia muestra la forma en que se obtiene la opción óptima a través de las iteraciones del módulo de aprendizaje.

Asimismo, se conserva un listado de las decisiones tomadas en cada iteración por el algoritmo de aprendizaje ejecutado. Los datos incluyen la recompensa y el número de selecciones de cada opción, la recompensa promedio, la opción solicitada por el módulo de aprendizaje y la que se obtiene en el ambiente, junto con la iteración de convergencia al óptimo; asimismo, se registran los datos globales de la corrida, que incluyen las recompensas acumulada y promedio obtenidas en la corrida, el factor XLearn (modificado de la ecuación 3.1, sección 3.2, calculado sobre los datos finales, no sobre cada tipo de opción), los porcentajes de solicitud de la opción óptima, de su obtención y de la selección de opciones no óptimas ("sobrevivencia").

Todos estos resultados son almacenados en un archivo .txt para su posterior consulta y en un archivo .pdf se conservan las curvas y la gráfica de activación en el tiempo. Además, en un archivo .txt adicional se conservan los datos de generación de las gráficas ya mencionadas, así como del tiempo de búsqueda de las opciones (no graficado).

Capítulo 5
Simulaciones

Debido al interés en el modelado del comportamiento de los monos aulladores, hecho con la herramienta ABC de [Gue97c] y [Mon98], el objetivo de la presente investigación es modelar, mediante aprendizaje, la conducta de forrajeo de estos animales, por medio de la simulación de su toma de decisiones frente a diversas opciones alimenticias.

Para lograr esto, en primer lugar se analizaron los datos estadísticos reales de las investigaciones etológicas de [Rod94] y [Dom04], vistos en la sección 3.3. Para simular el comportamiento del animal, se diseñaron Redes de Comportamiento que se aproximaran a estos resultados, las cuales constituyen los dos modelos computacionales con los que se experimenta con ABC_ApR. Sobre estos modelos se aplicaron los diversos algoritmos de aprendizaje a la conducta de alimentación.

Las Redes de Comportamiento diseñadas comprenden conductas básicas simplificadas y son muy sencillas, pero resultan suficientes para la obtención de simulaciones que se aproximen a datos reales. Así, para las dos redes diseñadas se calibró experimentalmente con respecto al intervalo de tiempo de la dinámica del mundo, para cambiar las proposiciones verdaderas del mundo simulado, y se ajustaron los pesos de las motivaciones de las conductas (la parte de acción del módulo de comportamiento de [Mon98]).

Por tanto, en este capítulo se expondrán, en primer lugar, los datos obtenidos de cada estudio etológico junto con los datos generados por el modelo computacional respectivo. Posteriormente se presentará un ejemplo de cada modelo con aprendizaje y finalmente se analizarán los resultados obtenidos bajo los diversos experimentos, presentando las observaciones más importantes en cuanto al desempeño de los algoritmos de aprendizaje y de la Red de Comportamiento con Aprendizaje por Reforzamiento.

5.1 Simulaciones sin aprendizaje

Esta sección presenta los datos obtenidos de las observaciones etológicas consideradas en la presente investigación, en base a los cuales se simula computacionalmente el comportamiento del mono aullador. Así, se muestran los resultados presentados por Rodríguez Luna [Rod94], para ser comparados con los obtenidos por el modelo de simulación ABC_RodríguezLuna. Lo mismo se hace para los datos de la investigación de Domingo Balcells [Dom04] y los resultados del

modelo ABC_DomingoBalcells.

Cabe hacer mención que en la investigación de Domingo se reportan gráficas del Patrón Diario de Actividades, PDA, individuales y grupales. Este tipo de gráficas muestran la proporción de tiempo dedicada a cada actividad del tiempo total diario, y sirven para comparar los patrones de cada conducta durante intervalos, dando una perspectiva del tiempo dedicado a cada conducta del total del tiempo de observación.

Este tipo de gráfica es una representación más compacta de la activación de las conductas, en comparación a la gráfica de activación en el tiempo generada por las versiones anteriores de la herramienta ABC (que presenta los niveles de la energía de activación de las conductas en cada iteración). Por este motivo, se optó por incluir entre los resultados obtenidos por ABC_ApR a los datos necesarios para generar gráficas tipo PDA, las cuáles se muestran para los modelos de simulación sin aprendizaje y sus ejemplos con aprendizaje.

5.1.1. Estudio etológico de Rodríguez Luna

El trabajo de Rodríguez Luna [Rod94] (ver sección 3.3, p. 45) presenta un modelo estocástico y los datos estadísticos producto de la observación del comportamiento de una tropa de monos aulladores. Estos últimos constituyen el Patrón Diario de Actividades (PDA), o la distribución de las principales actividades realizadas por los primates a lo largo del día.

Las conductas que se incluyen en el modelo son alimentación, descanso y exploración. Los datos estadísticos son: el porcentaje ocupado en cada conducta y los tiempos de espera estacionales -tiempos promedio de permanencia en una conducta-, presentados en la tabla 5.1, y; la matriz de probabilidades de transición, conteniendo las probabilidades de pasar de una conducta a otra dependiendo de la presente. Las tablas 5.2 y 5.3 muestran las matrices de probabilidades para las estaciones húmeda y seca, respectivamente.

Conducta	Porcentaje	Tiempo promedio EH	Tiempo promedio ES
explorar	15	12.93	10.38
alimentar	25	24.35	19.45
descansar	60	54.45	71.48

Cuadro 5.1: Porcentaje y tiempos promedio para estaciones húmeda (EH) y seca (ES).

Conducta	explorar	alimentar	descansar
explorar	0.00	0.57	0.43
alimentar	0.50	0.00	0.50
descansar	0.56	0.44	0.00

Cuadro 5.2: Matriz de probabilidades de transición para la estación húmeda.

Conducta	explorar	alimentar	descansar
explorar	0.00	0.56	0.44
alimentar	0.38	0.00	0.62
descansar	0.59	0.41	0.00

Cuadro 5.3: Matriz de probabilidades de transición para la estación seca.

5.1.1.1. Modelo ABC_RodríguezLuna

Sobre los datos presentados por Rodríguez Luna se buscó diseñar una Red de Comportamiento que se aproximara al patrón diario de actividades del mono aullador. El flujo de activación/inhibición de la Red de Comportamiento que modela el comportamiento del animal de acuerdo a los resultados de Rodríguez Luna se presenta en la gráfica 5.1.

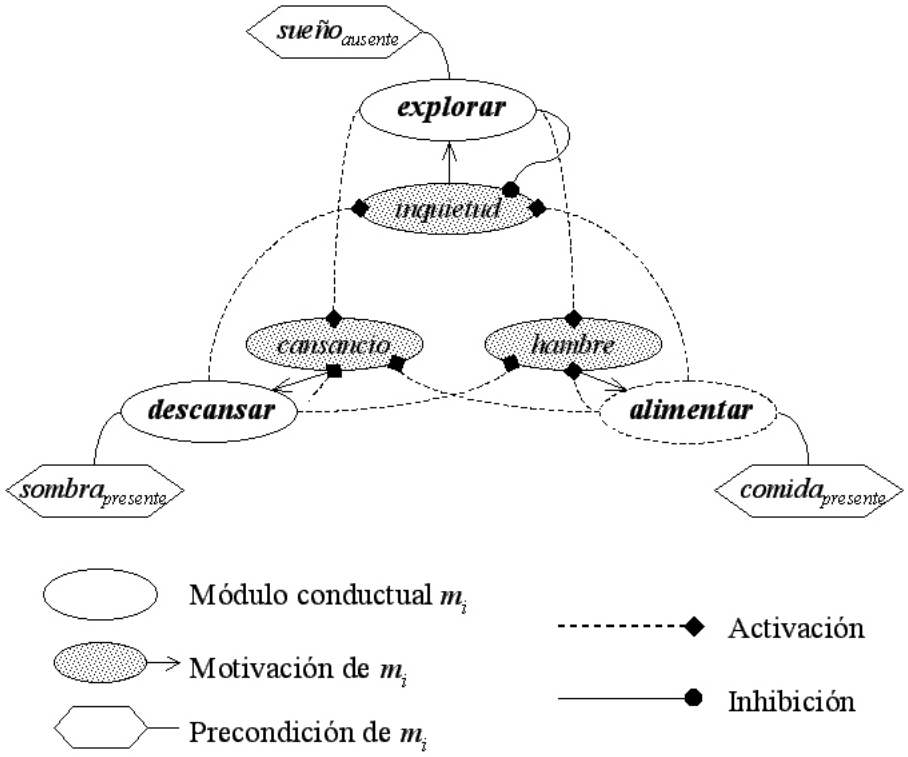

Figura 5.1: Modelo ABC_RodríguezLuna - Activación/inhibición de la Red de Comportamiento.

Esta simulación, que se ha nombrado modelo ABC_RodríguezLuna, arrojó los resultados que se muestran en los cuadros 5.4 y 5.5 con porcentaje y tiempo promedio, el primero, y matriz de transiciones, el segundo. Para fines comparativos, la tabla 5.4 presenta el porcentaje reportado por Rodríguez Luna.

Conducta	Porcentaje ABC_RodríguezLuna	Tiempo promedio (ciclo*4.662 minutos)	Porcentaje Rodríguez Luna
explorar	15.33	13.94	15
alimentar	24.73	22.47	25
descansar	59.93	54.45	60

Cuadro 5.4: Porcentaje y tiempos promedio del modelo ABC_RodríguezLuna. Notar aproximación del tiempo al reportado para la temporada húmeda de la tabla 5.1.

Conducta	explorar	alimentar	descansar
explorar	0.00	0.57	0.43
alimentar	0.44	0.00	0.56
descansar	0.57	0.43	0.00

Cuadro 5.5: Matriz de probabilidades de transición del modelo ABC_RodríguezLuna. Existe una aproximación a los datos de la tabla 5.2.

Estos resultados se aproximan bastante a los datos reales correspondientes a la estación húmeda que reporta Rodríguez (tablas 5.1 y 5.2). Los tiempos promedio de espera han sido calculados a partir de los ciclos promedio de ABC en que el agente permaneció en cada conducta; para igualarlos a los datos reales, se asume que un ciclo equivale a 4.662 minutos.

Un fragmento de la gráfica de activación en el tiempo generada por la Red de Comportamiento se presenta en la figura 5.2. Esta gráfica muestra los niveles de áctivación de cada conducta y, de ésta manera, cuándo se seleccionaron y por cuánto tiempo (ver sección 3.1 para una mayor explicación).

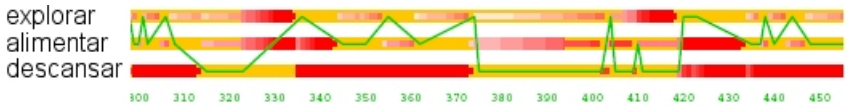

Figura 5.2: Segmento de gráfica de activación en el tiempo de la Red de Comportamiento del modelo ABC_RodríguezLuna

La figura 5.3 presenta la gráfica del Patrón de Actividades que esta simulación generó, la cuál presenta los patrones de activación de cada conducta por bloques de tiempo. El eje Y presenta el tiempo en que la conducta estuvo activa, en tanto que el eje X muestra los bloques de ciclos de ABC.

Se observa en el patrón de actividades que la conducta *descansar* se mantiene activa la mayor parte del tiempo, en tanto que la conducta *alimentar* es la segunda en tiempo de activación, y que cuando esta conducta ocupa más tiempo, la conducta *descansar* presenta un descenso en su tiempo de activación.

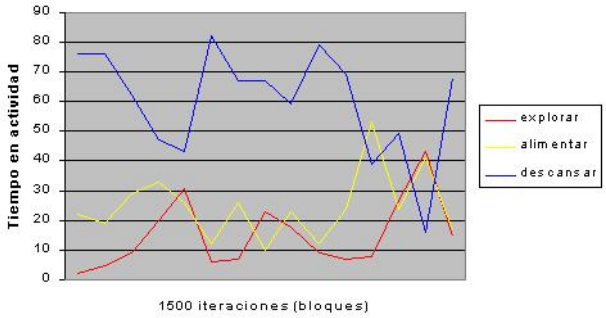

Figura 5.3: Patrón de actividades del modelo ABC_RodríguezLuna

En el apéndice (p. 99), se pueden consultar la codificación que estructura la Red de Comportamiento de este modelo (listado A.1) y la gráfica de activación en el tiempo completa (figura A.1).

5.1.2. Estudio etológico de Domingo Balcells

El estudio de Domingo [Dom04] (ver p. 45) trata del forrajeo del mono aullador bajo un contexto experimental que incluye el uso de parches alimenticios a fin de identificar el comportamiento del animal al buscar un alimento de preferencia, así como situaciones ambientales cambiantes. El trabajo de Domingo es de importancia a la presente investigación, debido a que el uso de parches alimenticios es el modelado que interesa en este trabajo, para la especialización de selección que se planea mediante aprendizaje.

El Patrón Diario de Actividades que reporta Domingo en base a sus observaciones está constituido únicamente por el porcentaje de tiempo que el animal invierte en las diversas actividades, las cuales son descanso, alimentación, búsqueda de alimento, exploración e interacciones sociales.

El cuadro 5.6 muestra los porcentajes presentados por Domingo, aclarando que, para fines del modelo computacional, la conducta de alimentación comprende también la conducta de búsqueda de alimento (11.09 + 2.41 = 13.50).

Conducta	Porcentaje
social	3.17
explorar	5.80
alimentar	13.50
descansar	77.53

Cuadro 5.6: Porcentajes reportados por Domingo.

A pesar de carecer de los tiempos de espera y de la matriz de transiciones, el trabajo de Domingo sí reporta gráficas que muestran el patrón de cada una de las actividades principales

en función de la proporción de tiempo dedicado a cada una de ellas. De ahí surgió la idea de obtener las gráficas PDA como parte de los resultados de la herramienta, para lo cual se almacena el tiempo en que cada conducta está activa, para poder graficar el patrón de comportamiento.

5.1.2.1. Modelo ABC_DomingoBalcells

El diseño de la Red de Comportamiento del modelo ABC_DomingoBalcells incluyó, además de las conductas descanso, alimentación y exploración, la conducta social. El flujo de activación/inhibición de la red diseñada para emular los estudios de Domingo Balcells se puede observar en la gráfica 5.4.

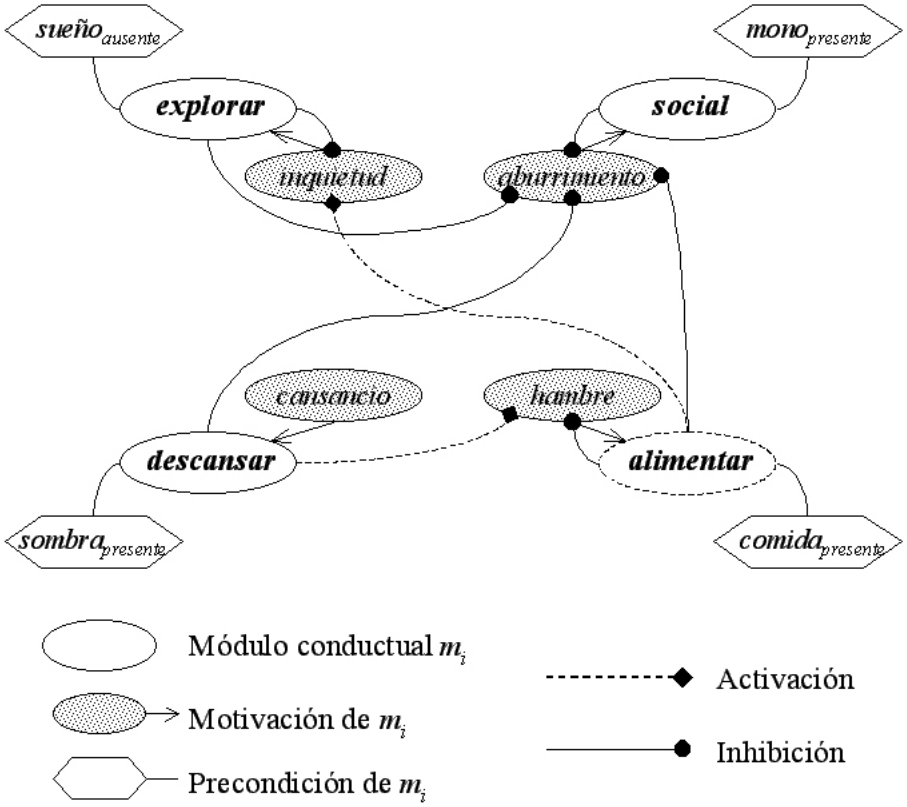

Figura 5.4: Modelo ABC_DomingoBalcells - Activación/inhibición de Red de Comportamiento.

Puesto que Domingo no reporta datos respecto al tiempo promedio y a la matriz de transiciones, no es posible calcular el tiempo promedio a partir de los ciclos promedio que arroja ABC. Sin embargo, teniendo en cuenta los buenos resultados obtenidos con el modelo ABC_RodríguezLuna, el hecho de que ABC_DomingoBalcells se aproxime a los porcentajes reportados por Domingo, nos permite considerar válidos todos los resultados para fines de análisis del desempeño de la herramienta.

Los cuadros 5.7 y 5.8 exhiben el patrón de comportamiento para el agente simulado bajo este modelo. Como se observa, existe una gran similitud en el porcentaje del agente simulado con el del animal (presentado nuevamente en la tabla 5.7).

Conducta	Porcentaje ABC_DomingoBalcells	Ciclos ABC promedio	Porcentaje Domingo Balcells
social	3.33	1.72	3.17
explorar	5.87	2.51	5.80
alimentar	13.53	3.03	13.50
descansar	77.27	22.29	77.53

Cuadro 5.7: Porcentaje y ciclos promedio del modelo ABC_DomingoBalcells.

Conducta	social	explorar	alimentar	descansar
social	0.00	0.07	0.45	0.48
explorar	0.06	0.00	0.71	0.23
alimentar	0.19	0.37	0.00	0.43
descansar	0.27	0.16	0.57	0.00

Cuadro 5.8: Matriz de probabilidades de transición del modelo ABC_DomingoBalcells.

La gráfica correspondiente al patrón de actividades, que exhibe el tiempo de actividad de cada módulo de la red, se puede consultar en la figura 5.5.

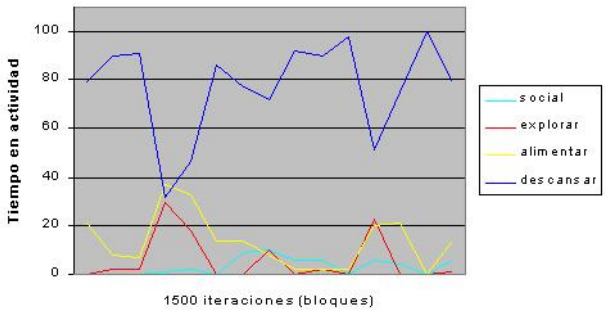

Figura 5.5: Patrón de actividades del modelo ABC_DomingoBalcells

La figura 5.6 exhibe un fragmento de la gráfica de activación en el tiempo generada por este modelo. En ella se representa la forma en que los módulos conductuales se fueron activando y durante cuánto tiempo, además de mostrar cómo se fue acumulando la energía de activación en cada conducta.

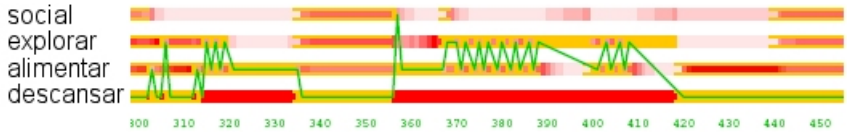

Figura 5.6: Segmento de gráfica de activación en el tiempo de la Red de Comportamiento del modelo ABC_DomingoBalcells

La codificación de este modelo computacional se presenta en el listado A.4 (p. 106) del apéndice, en donde también se incluye la gráfica de activación en el tiempo (sobre los 1500 ciclos de ABC), en la figura A.6.

5.2 Simulaciones con aprendizaje

Hasta ahora, se ha evidenciado la utilidad de las Redes de Comportamiento y, específicamente, de la herramienta ABC para el modelado de simulaciones de comportamientos reales observados en monos aulladores, presentado los resultados generados por tales modelos y remarcando su similitud con los reportados por las investigaciones etológicas, lo cuál hace que tales modelos computacionales sean considerados válidos en sus dinámicas de selección de conductas y, por tanto, estén bien fundamentados para fines de análisis del desempeño de la herramienta.

Con el diseño y obtención de Redes de Comportamiento que reportan un buen desempeño, se procede ahora a probar ABC_APR, aplicando el módulo de Aprendizaje por Reforzamiento a la toma de decisiones del mono simulado en su conducta *alimentar*, para lo cual se enfrentará a situaciones donde existan diversas opciones alimenticias (op_i), de las que se espera logre identificar la opción que le brinde mayor ganancia al largo plazo ($op*$), o en términos comúnes, que le brinde mejor satisfacción en su necesidad de alimentación. El hecho de que existan diversos tipos de opciones alimenticias permite enfrentar la especialización de selección de opciones, además de que su tipificación es esencial a la emulación de los experimentos de Domingo.

Así, en ABC_ApR se ejecutará cada modelo simulado bajo diversos experimentos, conformados por variaciones en el número del tipo de opciones alimenticias en el mundo y por diversos tipos de mundos simulados (por la cantidad de opciones, escaso o abundante; por su distribución, aleatorio o por zonas), considerando además diversas configuraciones de probabilidades de recompensa de cada tipo de opción.

El número de opciones alimenticias, N, con los que se trabajó varió en *2, 3* y *4* opciones; el diseño de la herramienta soporta un número mayor de opciones, según se desee probar.

Cada tipo de opción requiere tener asociada una probabilidad de recompensa, mediante la cual se determina si una opción otorga o no recompensa al ser ejecutada; esta probabilidad es independiente de las probabilidades de las otras opciones. Para los experimentos se establecieron cinco configuraciones de probabilidades de recompensa por cada variación de tipo de opción (con

$N = 2, 3, 4$). A tales configuraciones se les denominará en lo sucesivo *casos probabilísticos*. Los cuadros 5.9, 5.10 y 5.11 muestran las diversas probabilidades para 2, 3 y 4 opciones, respectivamente.

Caso	Opción 1	Opción 2
0	0.90	0.80
1	0.90	0.10
2	0.60	0.40
3	0.50	0.50
4	1.00	0.10

Cuadro 5.9: Casos probabilísticos de recompensa para $N = 2$.

Caso	Opción 1	Opción 2	Opción 3
0	0.80	0.90	0.60
1	0.70	0.90	0.20
2	0.30	0.60	0.10
3	0.50	0.50	0.50
4	0.10	1.00	0.10

Cuadro 5.10: Casos probabilísticos de recompensa para $N = 3$.

Caso	Opción 1	Opción 2	Opción 3	Opción 4
0	0.70	0.80	0.90	0.60
1	0.30	0.70	0.90	0.20
2	0.50	0.30	0.60	0.10
3	0.50	0.50	0.50	0.50
4	0.10	0.10	1.00	0.10

Cuadro 5.11: Casos probabilísticos de recompensa para $N = 4$.

Los casos 3 y 4 fueron propuestos para analizar el desempeño de los algoritmos ante situaciones extremas: una, el caso 4, donde se cree fácil la tarea de determinar la opción óptima; la otra situación, el caso 3, donde todas las opciones poseen la misma probabilidad de ser recompensadas. Además, cada uno de los casos probabilísticos guarda cierta similitud para las diversas opciones, esto con la intención de analizar la variabilidad en el desempeño en función del número de opciones.

Los parámetros requeridos por los diversos algoritmos de aprendizaje se presentan en el cuadro 5.12. Todos los algoritmos cuentan con una etapa de predecisión, igual al número de opciones, N, durante la cual probarán las diversas opciones sin aplicar su política de selección, pero sí aprenderán de la rentabilidad de las opciones.

Algoritmo	Parámetro	Valor	Parámetro	Valor
Bandido	k	$\{2, 3, 4\}$		
ε-Greedy	ε	0.1		
Softmax	T	100		
LRP	α	0.1	β	0.1
LRP	α	0.1		
IE	z	1.96		
RC	α	0.15	β	0.1

Cuadro 5.12: Parámetros de los algoritmos de aprendizaje y sus valores.

Los algoritmos IE y RC aplican su mecanismo de aprendizaje y registran así su decisión acerca de la deseabilidad de las opciones, para posteriormente aplicar una política ε-Greedy, donde el valor de ε es decrementado con el tiempo, para favorecer la selección de la que se considera, es la mejor opción. Así, en ambos casos, $\varepsilon\_inicial = 1.00$ y $\varepsilon\_final = 0.50$. El decremento de ε se da al llegar a las 50 iteraciones de aprendizaje.

A continuación se expondrán los ejemplos de dos corridas típicas, una por cada modelo simulado, presentando sus resultados y gráficas. Así mismo, en el apéndice (p. 99) se presentan los datos generales de todo el experimento al que la corrida presentada pertenece.

5.2.1. Modelo ABC_RodríguezLuna

Como ejemplo de este modelo computacional se ha seleccionado la corrida del algoritmo Softmax bajo las condiciones experimentales que se indican en el recuadro inmediato. Los resultados arrojados por la corrida también son presentados.

Algoritmo	No. Opción	Caso probabilístico	Tipo de mundo
Softmax	2	2 (0.60, 0.40)	Abundante aleatorio

RESULTADOS:

Recompensa promedio	66.36	
Recompensa acumulada	71	
Total de iteraciones	107	
Factor XLearn	1.11	
Iteración de convergencia	6	
Solicita óptima	105	98.13 %
Obtiene óptima	104	97.20 %
Sobrevivencia	3	2.80 %
Periodo de predecisión	1-5	
R_1/A_1	70/104	
R_2/A_2	1/3	

Como se puede observar, se trata de un caso probabilístico donde la probabilidad de la recompensa deseada es 0.60, habiendo logrado el algoritmo una recompensa promedio de 66.36 %, es decir, supera lo esperado. Su porcentaje de convergencia al óptimo es de 98.13 %, habiendo tomado la decisión de la mejor opción en la iteración 6, pero por sobrevivencia, es a partir de la iteración 89 que siempre ejecutó opciones óptimas.

El término *sobrevivencia* define ocasiones donde se consumen opciones no óptimas, ya sea porque así lo solicita el módulo de aprendizaje o por excepciones en el ambiente, por ejemplo, cuando la opción solicitada por el módulo de aprendizaje no es validada por la percepción del agente (no ve lo que quiere comer). Por periodo de predecisión se entiende el tiempo transcurrido desde la primera iteración de aprendizaje hasta el momento inmediato anterior a la primera consumación de opciones óptimas a solicitud del módulo de aprendizaje.

Otra perspectiva del desempeño obtenido en la corrida ejemplo es presentada por las curvas de aprendizaje y convergencia resultantes. En la curva de aprendizaje se grafican puntos que representan el reforzamiento promedio recibido sobre el total de iteraciones de aprendizaje, en tanto que la curva de convergencia se delinea por el acumulado (por bloques de iteraciones) de las veces que se ejecutó la opción óptima.

La curva de aprendizaje es mostrada en la figura 5.7. Es posible observar en la curva de aprendizaje cómo fue la maximización de la recompensa promedio, que incluso supera la recompensa que se esperaba (la línea recta representa la recompensa que genera la opción óptima, de ser seleccionada siempre).

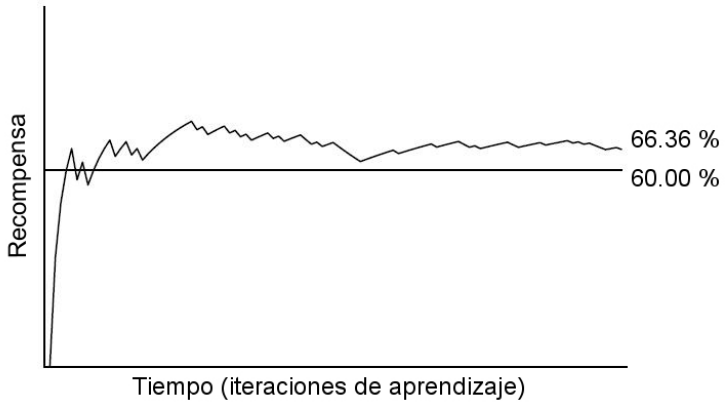

Figura 5.7: Curva de aprendizaje del ejemplo del modelo ABC_RodríguezLuna. La recompensa promedio recibida supera lo esperado.

La curva de convergencia se presenta en la figura 5.8. La línea horizontal discontinua representa el porcentaje de convergencia esperado, es decir, la convergencia óptima que se obtendría al seleccionar y ejecutar en todo momento la opción óptima, $op*$. Por otro lado, la línea sólida fue graficada con la cantidad acumulada por bloques de las veces que se ejecutó la opción óptima. La

corrida ejemplo exhibe una buena convergencia al óptimo en su curva , ya que la opción óptima se obtuvo casi desde el principio, aunque al final se observa una ligera baja en la convergencia, provocada por sobrevivencia, al no percibir en el ambiente a la opción óptima solicitada por el aprendizaje.

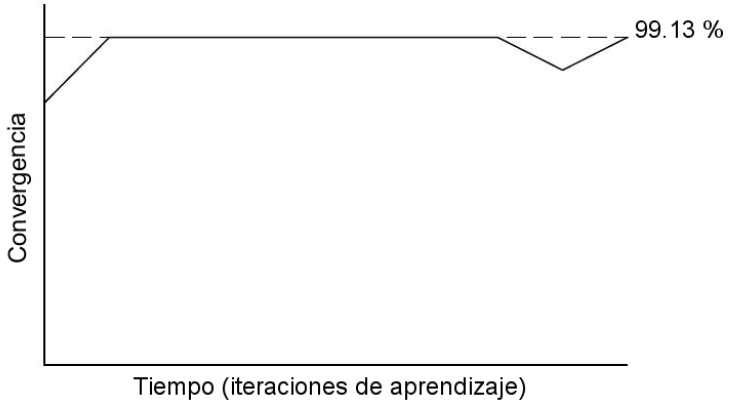

Figura 5.8: Curva de convergencia del ejemplo del modelo ABC_RodríguezLuna. Se obtuvo la opción óptima la mayor parte del tiempo.

Dentro del patrón de comportamiento obtenido, el porcentaje es dado en la tabla 5.13, en tanto que la matriz de probabilidades de transición se presenta en el cuadro 5.14. Comparando contra los datos reales, de las tablas 5.1 y 5.2 (p. 66), es posible observar que el comportamiento en esta corrida ejemplo se aproxima a ellos, siendo a la vez diferentes a los reportados por el modelo computacional sin aprendizaje (tablas 5.4 y 5.5, p. 68).

Conducta	Porcentaje	Ciclos promedio ABC	Tiempo (ciclo*4.662)
explorar	15.07	2.83	13.19
alimentar	26.20	4.73	22.05
descansar	58.73	11.59	54.03

Cuadro 5.13: Porcentaje y tiempos promedio del ejemplo del modelo ABC_RodríguezLuna.

Conducta	explorar	alimentar	descansar
explorar	0.00	0.55	0.45
alimentar	0.51	0.00	0.49
descansar	0.49	0.51	0.00

Cuadro 5.14: Matriz de probabilidades de transición del ejemplo del modelo ABC_RodríguezLuna.

La corrida completa se encuentra en el listado A.2 (p. 101) del apéndice, donde se muestra la forma en como el algoritmo hizo su selección de opciones para llegar a los resultados ya expuestos. Así mismo, en el apéndice se presenta la gráfica de activación en el tiempo (figura A.2) y la gráfica del patrón de comportamiento (figura A.3).

El apéndice también incluye la corrida general del experimento al que pertenece la corrida que se ha presentado como ejemplo. Se muestran los resultados globales del experimento (listado A.3, p. 104), junto con una gráfica acumulada de las curvas de aprendizaje (figura A.5) y las curvas de aprendizaje y convergencia de los ocho algoritmos ApR y los dos comparativos (figura A.4). Así mismo, se comentan brevemente los resultados de la corrida general.

5.2.2. Modelo ABC_DomingoBalcells

Para ejemplificar la simulación del modelo ABC_DomingoBalcells se seleccionó la corrida del algoritmo Greedy en un experimento que consta de 2 opciones, con el caso probabilístico No. 1 (0.90, 0.10) y un mundo abundante con distribución aleatoria. Los resultados obtenidos en esta corrida se muestran a continuación.

Algoritmo	No. Opción	Caso probabilístico	Tipo de mundo
Greedy	2	1 (0.90, 0.10)	Abundante aleatorio

RESULTADOS:

Recompensa promedio	92.71	
Recompensa acumulada	106	
Total de iteraciones	115	
Factor XLearn	1.02	
Iteración convergencia	3	
Solicita óptima	114	99.13 %
Obtiene óptima	114	99.13 %
Sobrevivencia	1	0.87 %
Periodo de sobrevivencia	1-2	
R_1/A_1	106/114	
R_2/A_2	0/1	

La recompensa promedio que se obtiene (92.71 %) es superior a la que se esperaba (90 %), lo cual demuestra que hubo buen aprendizaje y se recompensó en gran número de veces a las opciones seleccionadas. La curva de aprendizaje que arrojó la corrida se muestra en la figura 5.9. Como se puede observar en esta gráfica, la recompensa recibida por las decisiones del aprendizaje llega a ser alta y con el tiempo, supera la recompensa esperada.

Por otra parte, la convergencia al óptimo fue muy rápida y no hubo necesidad de ejecutar opciones no óptimas. La figura 5.10 muestra la curva de convergencia de la corrida ejemplo, la cual representa el buen desempeño del algoritmo en la identificación temprana de la opción óptima (la línea discontinua corresponde a la convergencia ideal, al ejecutar siempre la opción

óptima).

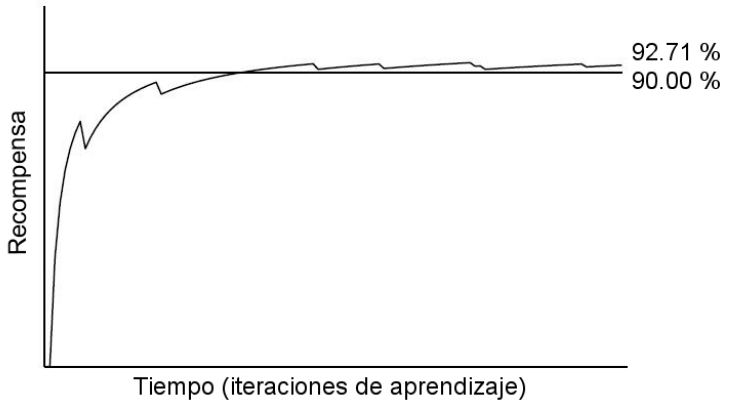

Figura 5.9: Curva de aprendizaje del ejemplo del modelo ABC_DomingoBalcells. La recompensa esperada (línea horizontal, 90 %) es superada por la recompensa promedio obtenida.

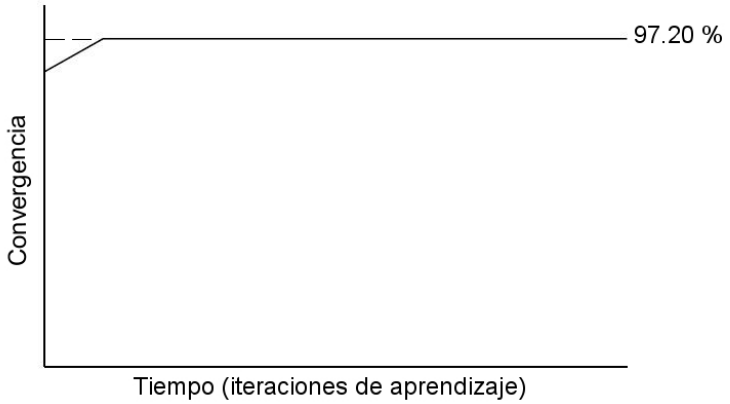

Figura 5.10: Curva de convergencia del ejemplo del modelo ABC_DomingoBalcells. La ejecución de la opción óptima predomina en toda la corrida ejemplo.

El porcentaje en que se activaron los módulos de la Red de Comportamiento, así como la matriz de transiciones que indica las probabilidades con las que se puede cambiar de un módulo a otros se muestran en las tablas 5.15, para el porcentaje, y 5.16, para la matriz.

En el apéndice se presenta la corrida completa en el listado A.5 (p. 108). Ahí mismo, la figura A.7 (p. 110) muestra la gráfica de activación en el tiempo generada y el patrón de comportamiento de todas las conductas se puede observar en la figura A.8.

Conducta	Ciclos promedio ABC	Porcentaje
social	1.70	3.40
explorar	2.05	5.60
alimentar	2.71	12.67
descansar	23.50	78.33

Cuadro 5.15: Porcentaje y ciclos promedio del ejemplo del modelo ABC_DomingoBalcells.

Conducta	social	explorar	alimentar	descansar
social	0.00	0.07	0.40	0.53
explorar	0.05	0.00	0.76	0.20
alimentar	0.20	0.44	0.00	0.36
descansar	0.29	0.16	0.55	0.00

Cuadro 5.16: Matriz de probabilidades de transición del ejemplo del modelo ABC_Domingo-Balcells.

La corrida general de la que es parte el ejemplo aquí presentado pertenece al experimento de 2 opciones, caso de probabilidades No. 1, mundo abundante aleatorio. El resumen de los resultados de todos los algoritmos y de los comparativos bajo estas condiciones se presenta en el listado A.6 del apéndice (p. 111), junto con las gráficas de aprendizaje y de convergencia (figura A.9) y una gráfica que incluye a todas las curvas de aprendizaje (figura A.10).

Como los dos ejemplos presentados tuvieron buena convergencia al óptimo y el ambiente les favoreció en la disponibilidad de las acciones solicitadas, casi no presentan variación en su tiempo de búsqueda. Para ilustrar una situación donde sí existe la modificación de la selección de acciones y donde el tiempo de búsqueda fluctúa, consultar el ejemplo de la gráfica A.11 del apéndice (p. 113).

5.3 Desempeño de los algoritmos de aprendizaje

Con los parámetros indicados en la sección anterior (tablas 5.9, 5.10, 5.11 y 5.12, pp. 73 y 74), para cada experimento se ejecutaron los algoritmos de Aprendizaje por Reforzamiento, junto con los mecanismos de selección implementados para fines comparativos (selección óptima, o paraíso, y selección sin aprendizaje), que se denominarán en lo sucesivo *comparativos no-ApR*; por tanto, los datos analizados de cada modelo constituyeron un colchón de pruebas de 60 corridas, cada una con los ocho algoritmos y los dos comparativos.

El número de iteraciones por corrida de algoritmo varía en función de las decisiones tomadas por la Red de Comportamiento. Por tanto, para analizar el desempeño de los algoritmos de aprendizaje se considera la recompensa promedio total recibida (junto con el factor XLearn de la corrida) y el porcentaje de solicitud de la opción óptima por parte de la política basada en aprendizaje.

El factor XLearn de la corrida surge de una modificación a la ecuación 3.1 (p. 44), que calcula el factor sobre cada tipo de acción para ser una estimación de la obtención de recompensa al seleccionar una acción específica. A diferencia de la ecuación original, el factor de la corrida es calculado sobre los datos finales de manera global (no sobre cada tipo de opción), para lo cual se calcula dividiendo la recompensa promedio de la corrida entre el porcentaje de la opción óptima.

A continuación se presenta el análisis del desempeño de los ocho algoritmos y los dos comparativos, dependiendo del número de opciones, del tipo de mundo simulado y de la configuración probabilística que determinó las recompensas de las opciones. Para los diversos análisis, se establecieron ordenamientos jerárquicos, de tipo descendente principalmente, según los diferentes rubros bajo examinación.

5.3.1. Por recompensa promedio y factor XLearn

Se analizó la recompensa promedio recibida en cada algoritmo ApR y comparativo no-ApR ejecutado, junto con el factor XLearn por corrida, el cual es un indicador del desempeño del algoritmo en la obtención de ganancia total (maximización de la recompensa promedio por corrida) mediante su especialización de selección de opciones.

Las tablas 5.17 y 5.18 muestran el ordenamiento jerárquico descendente de los algoritmos y compárativos, en base a estos dos indicadores de desempeño para los dos modelos probados.

Orden jerárquico	Algoritmo	% Recompensa promedio		Orden jerárquico	Algoritmo	% factor XLearn
1	Óptimo	11.729		1	Óptimo	11.555
2	Softmax	10.256		2	Softmax	10.258
3	ε-Greedy	10.155		3	ε-Greedy	10.175
4	IE	10.035		4	IE	10.025
5	Greedy	10.015		5	Bandido	10.010
6	Bandido	10.014		6	Greedy	9.937
7	RC	9.906		7	RC	9.908
8	LR	9.696		8	LR	9.743
9	LRP	9.564		9	LRP	9.596
10	SinApR	8.631		10	SinApR	8.793

Cuadro 5.17: Jerarquización de algoritmos y comparativos por recompensa promedio y XLearn para el modelo ABC_RodríguezLuna.

El cuadro 5.19 muestra el ordenamiento jerárquico global de todas las corridas de ambos modelos. Los algoritmos de Aprendizaje por Reforzamiento que obtienen mayor recompensa promedio son ε-Greedy y Softmax, pertenecientes ambos a las estrategias aleatorias, en tanto que la peor estrategia es la basada en vectores, pues los algoritmos LR y LRP no obtienen gran cantidad de

Orden jerárquico	Algoritmo	% Recompensa promedio		Orden jerárquico	Algoritmo	% factor XLearn
1	Óptimo	11.754		1	Óptimo	11.586
2	ε-Greedy	10.293		2	ε-Greedy	10.310
3	Softmax	10.194		3	Softmax	10.186
4	IE	10.104		4	IE	10.105
5	Greedy	10.087		5	Bandido	10.043
6	Bandido	10.052		6	Greedy	10.039
7	RC	9.939		7	RC	9.939
8	LR	9.668		8	LR	9.690
9	LRP	9.533		9	LRP	9.535
10	SinApR	8.376		10	SinApR	8.568

Cuadro 5.18: Jerarquización de algoritmos y comparativos por recompensa promedio y XLearn para el modelo ABC_DomingoBalcells.

Orden jerárquico	Algoritmo	% Recompensa promedio		Orden jerárquico	Algoritmo	% factor XLearn
1	Óptimo	11.742		1	Óptimo	11.571
2	Softmax	10.225		2	ε-Greedy	10.242
3	ε-Greedy	10.223		3	Softmax	10.122
4	IE	10.069		4	IE	10.065
5	Greedy	10.051		5	Bandido	10.026
6	Bandido	10.033		6	Greedy	9.988
7	RC	9.922		7	RC	9.923
8	LR	9.682		8	LR	9.716
9	LRP	9.549		9	LRP	9.566
10	SinApR	8.504		10	SinApR	8.681

Cuadro 5.19: Jerarquización global de algoritmos y comparativos por recompensa promedio y XLearn.

recompensa en su ejecución, pero logran más que el comparativo sin aprendizaje (SinApR). Como se esperaba, el comparativo Óptimo presenta el máximo porcentaje de recompensa recibida.

5.3.1.1. Desempeño en relación al número de opciones

Analizando el desempeño en ambos modelos, siempre se obtuvo mayor recompensa en las pruebas con dos opciones, posteriormente con tres y finalmente con cuatro opciones. Esto indica que entre menor sea el número de tipos de opciones entre las cuales decidir, se obtendrá mayor recompensa, pues la tarea de aprendizaje es más fácil para identificar a la mejor opción.

5.3.1.2. Desempeño en relación al tipo de mundo

El cuadro 5.20 muestra en qué tipos de mundo se obtuvo mayor recompensa promedio en cada uno de los modelos probados. Existen diferencias al jerarquizar el tipo de mundo, pues mientras el modelo de RodríguezLuna posiciona en primer lugar al mundo abundante con distribución por zonas y en segundo al mundo abundante aleatorio, el modelo de DomingoBalcells los clasifica intercambiados. La tabla 5.21 muestra la jerarquización global obtenida.

Orden jerárquico	ABC_RodríguezLuna	% Recomp. promedio	ABC_DomingoBalcells	% Recomp. promedio
1	M. abundante por zonas	27.17	M. abundante aleatorio	27.27
2	M. abundante aleatorio	27.10	M. abundante por zonas	26.99
3	Mundo escaso por zonas	23.77	Mundo escaso por zonas	23.76
4	Mundo escaso aleatorio	21.97	Mundo escaso aleatorio	21.98

Cuadro 5.20: Jerarquización de mundos en función a la recompensa promedio recibida - Modelos.

Orden jerárquico	Tipo de mundo	% Recomp. promedio
1	Mundo abundante aleatorio	27.18
2	Mundo abundante por zonas	27.08
3	Mundo escaso por zonas	23.76
4	Mundo escaso aleatorio	21.97

Cuadro 5.21: Jerarquización de los tipos de mundo en base a la recompensa promedio recibida.

En términos generales, la tendencia es que los mundos abundantes superan en recompensa promedio recibida a los mundos escasos. Por su parte, para ABC_RodríguezLuna los mundos con distribución por zonas son mejores al obtener recompensa que su respectivo mundo (escaso/abundante) con distribución aleatoria, es decir, la distribución de las opciones en el ambiente por medio de zonas favorece la maximización de la recompensa que se recibe; lo anterior se mantiene para ABC_DomingoBalcells en escasez, pero ante abundancia de opciones, la distribución aleatoria es mejor que las zonas alimenticias. Globalmente, se impone el patrón observado en ABC_DomingoBalcells.

5.3.1.3. Desempeño en relación al caso probabilístico

En función al factor XLearn, los diferentes casos probabilísticos se han ordenado jerárquicamente de mayor a menor, con la intención de mostrar cuál de ellos es el que favorece la maximización de la recompensa promedio por corrida. Los casos No. 3 y 0, para todos los números de opciones, siempre se posicionaron en el primero y segundo lugar, respectivamente; los demás casos presentaron variaciones en sus número de ordenamiento, para finalmente quedar organizados jerárquicamente como lo muestra la tabla 5.22.

Orden jerárquico	Caso probabilístico	% factor XLearn
1	3	22.89
2	0	21.63
3	1	19.37
4	2	18.79
5	4	17.32

Cuadro 5.22: Jerarquización de los casos probabilísticos en base al factor XLearn.

Con el caso No. 3 (0.50, ..., 0.50), donde todas las posibles opciones a seleccionar tienen asociada la misma probabilidad de recompensa, el seleccionar cualquier opción y la aleatoriedad al decidir el éxito favorecen el que se recompense gran cantidad de las opciones seleccionadas, superando en varias ocasiones la recompensa deseada (*50 %*).

El caso No. 4 (1.00, 0.10, ..., 0.10), por otra parte, es el que presenta dificultad al alcanzar la recompensa deseada (*100 %*), puesto que las corridas bajo este caso probabilístico exhiben un desempeño muy pobre en los mundos con cantidades escasas, lo cual repercute en los resultados globales. Este mismo patrón de bajo desempeño en la escasez lo muestra en menor grado el caso No. 1.

La maximización de recompensa de los casos No. 0 y 3 se debe a la poca diferencia que guarda la probabilidad de recompensa media de las opciones no óptimas, $\overline{PR}(op_i), op_i \neq op*$, con respecto a la probabilidad de recompensa de la opción óptima, $PR(op*)$ cuya relación se muestra en la tabla 5.23.

Caso probabilístico	$PR(_{op*})$	$\overline{PR}(_{op\neq op*})$ $N=2$	$\overline{PR}(_{op\neq op*})$ $N=3$	$\overline{PR}(_{op\neq op*})$ $N=4$	$PR(op*) - \overline{PR}(_{op\neq op*})$ $\forall N$
0	0.90	0.80	0.70	0.70	0.17
1	0.90	0.10	0.45	0.40	0.58
2	0.60	0.40	0.20	0.30	0.30
3	0.50	0.50	0.50	0.50	0.00
4	1.00	0.10	0.10	0.10	0.90

Cuadro 5.23: Diferencias entre la probabilidad de recompensa de la opción óptima $PR(op*)$ y las probabilidades medias no óptimas $\overline{PR}(op \neq op*)$.

La probabilidad de recompensa media de las opciones no óptimas es calculada de la siguiente forma

$$\overline{PR}(op_i) = \frac{\sum_{op_i \neq op*} PR(op_i)}{N-1}$$

Por ejemplo, para el caso probabilístico No. 1 de 4 opciones, las probabilidades de recompensa

de las opciones no óptimas son 0.30, 0.70 y 0.20, cuyo promedio nos da un valor de 0.40 para su probabilidad de recompensa media.

Por su parte, la probabilidad global de recompensa no óptima, $\overline{PR}(op \neq op*), \forall\, N$, es calculada al promediar, por cada caso probabilístico, las N probabilidades medias no óptimas. Finalmente, a la probabilidad de recompensa de la opción óptima se le resta la probabilidad global de recompensa no óptima, para obtener la distancia entre ambas.

Se considera que los casos probabilísticos con la menor distancia entre las probabilidades $PR(op*)$ y $\overline{PR}_{(op \neq op*)}, \forall\, N$, son los que mayor recompensa brindarán, y que aquellos con la mayor distancia son los peores en la maximización de la ganancia.

En base a esto, y considerando los datos de la tabla 5.23, es posible establecer un número de ordenamiento de los casos probabilísticos en función de la minimización de la distancia entre las probabilidades de recompensas de las opciones óptima y no óptimas. La tabla 5.24 presenta la jerarquización global de las configuraciones probabilísticas en cuanto a su cercanía con la probabilidad de recompensa óptima.

Orden jerárquico	Caso probabilístico	Distancia entre probabilidades
1	3	0.00
2	0	0.17
3	2	0.30
4	1	0.58
5	4	0.90

Cuadro 5.24: Jerarquización de los casos probabilísticos en base a la distancia entre probabilidades.

Se puede enunciar que, con algunas excepciones, entre menor sea la diferencia entre las probabilidades de recompensar óptima y no óptimamente, mayor será la maximización de recompensa promedio en situaciones donde sea necesario elegir opciones no óptimas por escasez de la opción óptima deseada.

Como se puede observar, existe similitud entre la jerarquización de la tabla 5.24 y la presentada en la tabla 5.22; se cree que la diferencia en los casos No. 1 y 2 se debe a que el caso No. 1 posee una probabilidad de recompensa de 0.90 en su opción óptima, en tanto el caso No. 2 tiene una de 0.60, viéndose el primero más favorecido con la obtención de recompensa.

5.3.2. Por solicitud del óptimo

Los algoritmos fueron jerarquizados en base al porcentaje de solicitud de la opción óptima determinada mediante aprendizaje. Debido a las características de las simulaciones, la opción que

se cree mejor no se encuentra siempre, por lo que debe modificarse la selección determinada por el aprendizaje para optar por otra opción, evitando así los interbloqueos.

El porcentaje de solicitud de la opción óptima representa la convergencia al óptimo de los algoritmos ApR. El desempeño de los algoritmos bajo este rubro para cada una de las simulaciones es mostrado en las tablas 5.25 y 5.26.

Orden jerárquico	Algoritmo	% Convergencia
1	ε-Greedy	13.96
2	IE	13.93
3	Bandido	13.34
4	Softmax	13.22
5	Greedy	13.00
6	RC	12.29
7	LR	10.48
8	LRP	9.78

Cuadro 5.25: Jerarquización de algoritmos por convergencia - modelo ABC_RodríguezLuna.

Orden jerárquico	Algoritmo	% Convergencia
1	IE	14.27
2	ε-Greedy	14.15
3	Softmax	13.45
4	Bandido	12.97
5	Greedy	12.80
6	RC	12.69
7	LR	10.10
8	LRP	9.57

Cuadro 5.26: Jerarquización de algoritmos por convergencia - modelo ABC_DomingoBalcells.

A diferencia del análisis de desempeño por recompensa promedio recibida y el factor XLearn, en este análisis no se tomaron en cuenta los comparativos Óptimo y SinApR, dado que el primero siempre converge al óptimo y el segundo, debido a que no existe aprendizaje, no guarda registro de convergencia.

Como se puede observar, existen discrepancias en las jerarquizaciones de los modelos, por lo que la tabla 5.27 muestra una jerarquía globalizada de la convergencia. El algoritmo por Intervalos de Confianza ahora es el que obtiene la mayor convergencia al óptimo, desplazando a segundo y tercer lugar a ε-Greedy y Softmax. Como en el análisis por obtención de recompensa, los algoritmos basados en vectores resultan ser los peores al momento de converger.

Orden jerárquico	Algoritmo	% Convergencia
1	IE	14.10
2	ε-Greedy	14.06
3	Softmax	13.34
4	Bandido	13.15
5	Greedy	12.90
6	RC	12.49
7	LR	10.29
8	LRP	9.67

Cuadro 5.27: Jerarquización global de algoritmos ApR por convergencia al óptimo.

5.3.2.1. Desempeño en relación al número de opciones

Los algoritmos, de manera general para ambos modelos simulados, se comportaron en la siguiente forma: cuando hubo sólo dos tipos de opciones sobre las cuáles decidir, se tuvo mayor convergencia al óptimo que cuando hubo tres o cuatro tipos; asimismo, con tres tipos de opciones hubo mayor convergencia que con cuatro. Esto muestra que es más fácil la convergencia mientras menor sea el número de opciones entre las cuáles seleccionar.

5.3.2.2. Desempeño en relación al tipo de mundo

Los porcentajes de convergencia al óptimo obtenidos por cada uno de los modelos, ordenados descendentemente, determinan las jerarquias que se pueden observar en los cuadros 5.28 y 5.29. De manera general, la jerarquización de los tipos de mundo es mostrada en la tabla 5.30.

Orden jerárquico	Tipo de mundo	% Convergencia
1	Mundo abundante aleatorio	25.61
2	Mundo escaso aleatorio	25.22
3	Mundo abundante por zonas	24.82
4	Mundo escaso por zonas	24.35

Cuadro 5.28: Porcentaje de convergencia según el tipo de mundo - modelo ABC_RodríguezLuna.

Orden jerárquico	Tipo de mundo	% Convergencia
1	Mundo abundante aleatorio	25.69
2	Mundo escaso aleatorio	25.19
3	Mundo escaso por zonas	24.91
4	Mundo abundante por zonas	24.21

Cuadro 5.29: Porcentaje de convergencia según el tipo de mundo - modelo ABC_Domingo-Balcells.

Orden jerárquico	Tipo de mundo	% Convergencia
1	Mundo abundante aleatorio	25.65
2	Mundo escaso aleatorio	25.20
3	Mundo escaso por zonas	24.63
4	Mundo abundante por zonas	24.52

Cuadro 5.30: Jerarquización de los tipos de mundo en función a la convergencia al óptimo - ambos modelos

Cabe mencionar que estos datos han sido determinados sin consideración del caso No. 3 (0.50,.., 0.50), en el cual no se puede diferenciar de entre todas las opciones una que sea mejor que las demás, pues todas ella tienen la misma probabilidad de recompensa asociada.

En general, bajo este indicador, los mundos aleatorios muestran una mayor convergencia que los mundos por zonas. Un punto a destacar es que esta medida de desempeño presenta una dependencia con la cantidad de opciones disponibles en el mundo, lo que hace que los mundos escasos favorezcan un buen reconocimiento de la opción óptima, pues a través de la dinámica del ambiente (que consiste en redistribuir las opciones cada determinadas épocas de aprendizaje), la escasez obliga a que el agente pruebe todas las opciones disponibles y con esto, le sea posible identificar a la opción óptima. Bajo estas condiciones, las experiencias de superviviencia son más comunes (el módulo de aprendizaje recomienda una opción que no se encuentra disponible en el ambiente), de forma que si el agente presentaba una tendencia a convergir hacia una opción subóptima, al experimentar otras opciones por sobrevivencia podrá corregir este error. En los mundos abundantes las experiencias de supervivencia son menos comunes, por lo que el agente confronta más fácilmente una repetición de opciones subóptimas.

5.3.2.3. Desempeño en relación al caso probabilístico

El orden jerárquico de los casos probabilísticos establecido en función a la convergencia exhibida por los algoritmos ApR se puede ver en la tabla 5.31; se analizó sobre los casos aplicados a los tres tipos de opciones probados. Para ambos modelos simulados se obtuvo la misma jerarquización.

Orden jerárquico	Caso probabilístico	% Convergencia
1	4	26.84
2	1	23.77
3	2	20.97
4	0	19.85
5	3	8.58

Cuadro 5.31: Jerarquización global de los casos probabilísticos en base al porcentaje de convergencia.

El caso No. 4 es el que se posiciona como el primero en converger, al tener el más alto porcentaje de solicitud de la opción óptima (con una probabilidad de recompensa asociada a la opción óptima de *1.00*). El peor desempeño en convergencia lo presenta el caso No. 3 (0.50, ..., 0.50).

Los niveles por convergencia 1, 2 y 5 se presentan por unanimidad en ambos modelos y para todos los mundos, en tanto, existe intercambio en los casos No. 0 y 2.

El ordenamiento en cuestión exhibe un desempeño similar al que se esperaba: los casos No. 4 y 1 se suponían fáciles de converger; el caso No. 2, medianamente fácil, y; los casos No. 0 y 3 se esperaban difíciles en la tarea de determinar la opción óptima.

Cabe notar que en la jerarquización de los casos probabilísticos mediante el factor XLearn (indicador de la recompensa promedio que toma en cuenta a la recompensa deseada), el caso No. 4 se ubicó en último lugar mediante la obtención de recompensa promedio, pero es primero en cuanto a convergencia al óptimo. Esto se debe a la afectación que los mundos escasos tienen al no encontrar la opción óptima solicitada y optar por una opción diferente a la convergida.

Además, bajo el factor XLearn, las jerarquías principales, 1 y 2, son para los casos que demostraron el más bajo desempeño en porcentaje de convergencia. En relación a esto se sugiere ver el comentario referente a la tabla 5.23 del desempeño de la recompensa en los casos probabilísticos (sección 5.3.1), donde se analiza el impacto de la diferencia entre la probabilidad de recompensa de la opción óptima y la probabilidad media de recompensa de las opciones no óptimas (notar que ahí se obtuvo una jerarquización inversa a la determinada por la convergencia).

5.3.3. Observaciones

5.3.3.1. Algoritmos con tendencia a obtener la misma recompensa

Considerando los resultados arrojados por las dos simulaciones, se observó una tendencia en ciertos algoritmos a obtener la misma recompensa, y en algunos casos, a comportarse de manera idéntica. En el cuadro 5.32 se presentan las combinaciones que sucedieron con más frecuencia en las corridas.

Combinación de algoritmos	Frecuencia
IE RC	20
Bandido Greedy	18
LRP LR	7
ε-Greedy IE RC	6
ε-Greedy RC	3
Softmax IE	3

Cuadro 5.32: Frecuencia de algoritmos con tendencia a obtener la misma recompensa.

La similitud entre IE y RC se debe a que ambos utilizan una política de selección de opción aleatoria adicional a su mecanismo de aprendizaje; debido a que tal política es ε-Greedy, en ocasiones ambos también son similares a este algoritmo (surgiendo incluso combinaciones). Al mostrar similitud, Bandido y Greedy demuestran que ambos pertenecen a las estrategias codiciosas, al igual que LRP y LR como estrategias basadas en vectores.

5.3.3.2. Algoritmos con tendencia a tener el mismo porcentaje de solicitud del óptimo

Analizando con respecto a los algoritmos que suelen ser idénticos en cuanto al porcentaje de solicitudes de la opción óptima, se obtuvo una lista de combinaciones, de las cuales la tabla 5.33 muestra las más frecuentes. 32 de las combinaciones se dieron con el modelo ABC_Rodríguez-Luna y 25 con el otro modelo. Cabe destacar que la mayoría de estas combinaciones se dio en experimentos con 2 opciones (41), en tanto que con 3 y 4 opciones, el número de repeticiones fue mínimo (9 y 7 respectivamente).

Combinación de algoritmos	Frecuencia
IE RC	17
Bandido Greedy	15
ε-Greedy IE RC	7
LRP LR	5
ε-Greedy IE	2
Bandido IE	2
Bandido Softmax	2

Cuadro 5.33: Frecuencia de algoritmos con tendencia a tener el mismo porcentaje de solicitud del óptimo.

5.3.3.3. Porcentaje de algoritmos que convergen a la recompensa deseada

Al analizar el total de las corridas, se destacaron aquellas donde se hubiese alcanzado o superado la recompensa deseada (la recompensa esperada de la opción óptima). Los algoritmos fueron entonces jerarquizados mediante el porcentaje de convergencia a la recompensa deseada, quedando como se muestra en el cuadro 5.34.

Cabe mencionar que en las jerarquías de los desempeños de algoritmos presentadas anteriormente, el algoritmo LR se ha ubicado penúltimo, al tener un desempeño pobre y sólo superar a LRP, pero bajo el criterio de convergencia a la recompensa deseada se clasifica en una mejor posición, lo que señala que este algoritmo es de convergencia deficiente y generalmente no obtiene bastante recompensa, pero cuando la logra, incluso puede superar la recompensa deseada (demuestra aquí superioridad ante los algoritmos del Bandido y RC).

Orden jerárquico	Algoritmo	% Convergencia a la recompensa deseada
1	Óptimo	33.33
2	Softmax	13.33
3	ε-Greedy	9.78
4	Greedy	8.89
5	IE	8.00
6	LR	6.67
7	Bandido	6.22
8	RC	5.33
9	LRP	4.89
10	SinApR	3.56

Cuadro 5.34: Jerarquización de algoritmos que convergieron a la recompensa deseada.

5.3.3.4. Porcentaje de algoritmos con la máxima recompensa por corrida

Considerando exclusivamente a los algoritmos que obtuvieron la máxima recompensa por corrida, se obtuvo una jerarquización de ellos mediante el ordenamiento descendente de sus porcentajes; esto nos indica cuáles algoritmos se comportaron con tendencias a maximizar la recompensa promedio. La tabla 5.35 presenta tal ordenamiento.

Orden jerárquico	Algoritmo	% Obtención de máxima recompensa
1	Greedy	22.86
2	Softmax	20.71
3	ε-Greedy	20.00
4	Bandido	12.14
5	RC	7.14
6	LR	6.43
6	IE	6.43
7	LRP	4.29

Cuadro 5.35: Jerarquización de algoritmos en base a la obtención de la máxima recompensa por corrida.

Cabe destacar que obtener la máxima recompensa no implica superar la recompensa deseada; esto se logra en 22 de 38 ocasiones bajo el modelo ABC_RodríguezLuna y en 25 veces de 35 con ABC_DomingoBalcells.

5.3.3.5. Comparación de porcentaje de sobrevivencia

Para los experimentos con aprendizaje del modelo de simulación ABC_DomingoBalcells, se seleccionó un algoritmo ApR por corrida (o dos algoritmos en situaciones donde presentaban un comportamiento idéntico) de entre los algoritmos que obtuvieron la máxima recompensa. Sobre esta colección, se analizó el porcentaje de sobrevivencia que tuvo cada algoritmo, para establecer una comparación con respecto a la sobrevivencia obtenida por el comparativo SinApR, a fin de observar si el aprendizaje (mediante la especialización de los comportamientos) representa una mejoría bajo este rubro y, de ser así, cuánto mejor es aprender a sólo comportarse sin conocimiento.

El porcentaje de sobrevivencia es un indicador de la selección y consumación de opciones no óptimas por parte del agente. Se seleccionó el comparativo SinApR debido a que no aplica ningún mecanismo de aprendizaje, por lo que el agente simulado sólo selecciona y consume las opciones que logra percibir en su entorno; por tanto, la cantidad de opciones no óptimas que consuma es independiente a todo mecanismo de aprendizaje. Por el contrario, los algoritmos de aprendizaje aplican políticas de selección en función a técnicas de aprendizaje, a fin de identificar la mejor opción.

En la tabla 5.36 se muestran los porcentajes de sobrevivencia de las corridas realizadas con ABC_DomingoBalcells. En promedio, el comparativo SinApR obtuvo un porcentaje de sobrevivencia de 59.34 %, en tanto que los mejores algoritmos ApR tuvieron una sobrevivencia de 38.43 %, habiendo entre ambos una diferencia de 20.91 %, lo cual indica que es mejor aplicar aprendizaje a fin de minimizar la selección de opciones no óptimas.

Porcentaje de sobrevivencia promedio	Algoritmos ApR	SinApR
	38.43	59.34

Cuadro 5.36: Sobrevivencia en los mejores algoritmos ApR y en el comparativo SinApR.

Por otro lado, con el propósito de analizar el nivel de sobrevivencia con respecto a los diferentes tipos de mundo, la tabla 5.37 muestra los porcentajes de sobrevivencia promedio de los algo-

Tipo de Mundo	% Sobrevivencia algoritmos ApR	% Sobrevivencia SinApR	Diferencia
M. escaso aleatorio	58.039	63.425	5.386
M. escaso por zonas	48.062	58.958	10.896
M. abundante aleatorio	23.167	56.175	33.008
M. abundante por zonas	24.463	58.810	34.347

Cuadro 5.37: Jerarquización de mundos en función al porcentaje de sobrevivencia promedio - Algoritmos ApR y SinApR.

ritmos ApR y de SinApR para cada tipo de mundo (escaso/abundante, aleatorio/por zonas), así como la diferencia entre ambos porcentajes.

Comparando los tipos de mundo mediante la sobrevivencia promedio, los mundos abundantes se destacan como los mejores para la minimización del consumo de opciones no óptimas con respecto a los mundos escasos. Pero, por otro lado, tanto en ambientes abundantes como en escasos, los mundos con distribución de opciones mediante zonas son mejores que aquellos distribuidos aleatoriamente; esto es más obvio en mundos de tipo escaso, en donde la diferencia entre los porcentajes de sobrevivencia para las zonas supera en más de un 100 % a la distribución aleatoria. Los experimentos inspirados en parches alimenticios, por tanto, demuestran ser provechosos para la explotación de lo aprendido con respecto a la identificación de la mejor opción.

Con lo anterior, queda de manifiesto que la aplicación de técnicas de Aprendizaje por Reforzamiento en búsqueda de una especialización de conducta, siempre es mejor al uso de la aleatoriedad en la toma de decisiones.

Conclusiones

Habiendo llevado al cabo la presente investigación se ha determinado un conjunto de conclusiones, mismo que se expone a continuación.

En primer lugar, se presenta la herramienta ABC_ApR en respuesta al objetivo principal del trabajo, que propone la implementación de un módulo de Aprendizaje por Reforzamiento que actua sobre uno de los módulos de una Red de Comportamiento. ABC_ApR añade técnicas de aprendizaje al mecanismo de selección de acción ya implementado en la herramienta ABC [Gue97c, Mon98], proponiendo que un módulo de la Red de Comportamiento sea especializado en su toma de decisiones mediante la aplicación de algoritmos de Aprendizaje por Reforzamiento (la figura 4.1 -p. 50- contiene la estructura general de la nueva herramienta).

Se ha propuesto la especialización de una conducta como una forma de selectividad en el momento en que un agente ejecuta tal conducta, en el sentido de que pueda decidir qué elegir de entre varias opciones disponibles para la conducta. Un ejemplo de esto es un agente simulado al momento de disparar una conducta de alimentación; sin especialización, únicamente se sabe que el agente comerá; con especialización, el agente puede decidir un tipo específico de alimento de entre varios que pueda consumir y, con el tiempo, comer aquél que considere mejor.

El hecho de que se aplique aprendizaje a una conducta tiene como objetivo conseguir su especialización, y es precisamente esta especialización de la conducta la que influencia la dinámica de la red.

En base a lo experimentado con ABC_ApR, se concluye que mediante los resultados arrojados por el módulo de aprendizaje, es posible influenciar la dinámica que, en base a su construcción inicial y al intercambio interno/externo de energías de activación, establece una Red de Comportamiento para la selección de conductas.

Como ya se sabe, para probar la herramienta se hicieron experimentos basados en información del campo etológico, en relación al estudio del comportamiento de un primate: el mono aullador. Los resultados obtenidos de las simulaciones se compararon con los datos reportados en los estudios etológicos de Rodríguez Luna [Rod94] y de Domingo Balcells [Dom04].

Desde la perspectiva de las simulaciones realizadas con ABC_ApR, se buscó que el mono simulado aprendiera acerca de las opciones alimenticias que su mundo contenía, a fin de especializarse

en la identificación e ingesta del mejor alimento, optimizando con esto su alimentación. Como resultado de este aprendizaje, el mono llega a ser selectivo en su toma de decisiones; un indicador de esta selectividad es el porcentaje de sobrevivencia presentado en la tabla 5.36 (p. 91) para una toma de decisiones aleatoria y una basada en técnicas ApR (hay menos sobrevivencia cuando se aplican técnicas de aprendizaje).

Por otro lado, el hecho de que la dinámica de la Red de Comportamiento se vea influenciada por decisiones basadas en aprendizaje no significa que se dé origen a sesgos en la selección, que favorezcan a la conducta sobre la que se lanza el módulo de aprendizaje; más bien, se re-equilibra constantemente la Red de Comportamiento, a fin de resolver las necesidades cambiantes del agente simulado.

En los experimentos realizados sobre dos simulaciones (ABC_RodríguezLuna y ABC_Domingo Balcells), la conducta *alimentar* conserva un porcentaje de activación de conducta aproximado al reportado en los respectivos estudios etológicos (ver tabla 5.1 -p. 66- para Rodríguez y tabla 5.6 -p. 69- para Domingo), pero al mismo tiempo cada experimento implica la optimización en la elección de la fuente alimenticia, dependiente de las características del mundo. Además, ante excepciones ambientales, el mono aplicó sobrevivencia mediante el consumo de opciones diferentes a la elegida por aprendizaje y, en general, presentó adaptabilidad a los diversos tipos de ambientes. Así, mediante una evaluación sencilla de los resultados que arroja el módulo de aprendizaje anexado, se abre toda una gama de exploración de variaciones del comportamiento de un agente simulado bajo un mismo modelo de Red.

En relación a otro de los objetivos de esta investigación, se presentan a continuación las conclusiones obtenidas mediante el análisis del desempeño de los algoritmos de Aprendizaje por Reforzamiento implementados.

Por un lado, en cuanto a la maximización de la recompensa esperada y la convergencia a la opción óptima, los algoritmos que demostraron ser mejores son ε-Greedy y Softmax (las dos estrategias aleatorias) y Estimación de Intervalos, que se ubica como el mejor algoritmo en cuanto a convergencia al óptimo (ver tablas 5.19 -p. 81- y 5.27 -p. 86).

En contraposición, las estrategias de aprendizaje con desempeño más pobre son las basadas en vectores: los algoritmos LRP y LR, de entre los cuales, LRP siempre resulta ser el peor bajo todos los desempeños, pero nunca se ubica por debajo de la situación donde se elige al azar (comparativo SinApR).

En ocasiones, los algoritmos tienden a comportarse de forma idéntica, ya sea porque obtienen el mismo porcentaje de solicitud de la opción óptima o porque igualan su recompensa promedio. Se presentan con más frecuencia cuatro combinaciones de algoritmos:

1) Estimación de Intervalos y Comparación de Reforzamiento.
2) Bandido y Greedy.

3) ε-Greedy, Estimación de Intervalos y Comparación de Reforzamiento.
4) LRP y LR.

Esta semejanza en el comportamiento se presenta ya sea porque los algoritmos pertenecen a una misma estrategia (*2, 4*), o bien, porque comparten la misma política de selección de acción (*1, 3*), que los hace identificarse con respecto a la estrategia de la política (tablas 5.32 y 5.33, p. 88).

Por otra parte, de entre todas las corridas, el algoritmo que converge más veces a la recompensa deseada es Softmax (tabla 5.34, p. 90).

A continuación se expondrán las conclusiones que derivan de las diversas configuraciones experimentales en cuanto al número de tipos de opciones de las cuales aprender, los ambientes en que el agente se encuentra inmerso y las probabilidades de recompensa asociadas a las diversas opciones.

En relación al número de opciones sobre las que se busca identificar la que sea de más beneficio para el agente, se establece que a menor número de opciones, las tareas de maximizar la recompensa esperada y de identificación de la opción óptima serán más fáciles (sin ignorar el hecho de que la identificación de la mejor opción depende fuertemente de las probabilidades de recompensa asociadas).

Cabe hacer mención que la obtención de recompensa promedio se ve afectada tanto por las probabilidades de recompensa asociadas, como por la cantidad y distribución de las opciones en el ambiente, por lo cual se observó que los mundos abundantes se muestran más propicios a lograr la maximización de la recompensa esperada que los mundos escasos (tabla 5.21, p. 82), ya que la abundancia de la opción óptima minimiza las excepciones por el ambiente (consumo de opciones no óptimas).

En los párrafos anteriores se ha mencionado la influencia de las probabilidades de recompensa asociadas a las opciones, acerca de lo cual se han formulado dos conclusiones, apoyadas en la diferencia existente entre la probabilidad de recompensa de la opción óptima y el promedio de las probabilidades de recompensa de las opciones restantes. Así, se ha observado que:

- A mayor distancia entre la probabilidad de recompensa de la opción óptima y el promedio de las probabilidades de recompensa de las opciones no-óptimas, "mayor será la convergencia al óptimo" (ver tabla 5.31 y comentario, p. 87).
- A menor diferencia entre la probabilidad de recompensa asociada a la opción óptima y la media de probabilidades de recompensa de las opciones no-óptimas, "mayor será la maximización de la recompensa promedio" en situaciones donde no se pueda acceder a la opción óptima. Se consideran excepciones ante buenas convergencias al óptimo, recompensas deseadas muy altas y disponibilidad de la opción óptima (ver tablas 5.22 -p. 83- y 5.24 -p. 84-, así como sus comentarios).

El primer punto establece que si al momento de ser recompensado a partir de diferentes tipos de

opciones, el beneficio obtenido de una de ellas (la opción óptima) se destaca de el de las otras, se logrará una rápida y correcta identificación de la mejor opción.

El segundo planteamiento establece el caso contrario, donde las probabilidades de recompensa de las opciones no se encuentran perfectamente diferenciadas, lo cual beneficia la maximización de recompensa aún en las situaciones donde el agente no haya elegido a la mejor opción.

En cuanto al porcentaje de solicitud de la opción óptima por parte del módulo de aprendizaje (que representa la convergencia al óptimo de los algoritmos), se tiene que la identificación correcta de la mejor opción se presenta con más frecuencia en ambientes aleatorios (tabla 5.30, p. 87), principalmente cuando existe abundancia en las opciones.

Por otro lado, se ha observado que los mundos escasos presentan una buena identificación de la opción óptima, lo cual se debe a que en estos mundos se presentan frecuentemente las excepciones por el ambiente (modificaciones a lo solicitado por falta de su percepción), que favorecen al agente al permitirle explorar más opciones, aún sin que así lo hubiera deseado; esto permite que en situaciones de convergencia a subóptimos, el agente pueda darse cuenta de la existencia de una mejor opción y por tanto, rectificar hacia la solicitud de ésta.

En este punto, habiendo expuesto lo relativo a la convergencia al óptimo y a la recompensa promedio obtenida, es de destacar la función ejercida por la cantidad de opciones disponibles en los dos tipos de mundo, pues se observa que en los ambientes escasos se favorece la convergencia a la opción óptima en mayor grado, en tanto que los mundos abundantes presentan las recompensas promedio más altas (nótese que una alta convergencia al óptimo no conlleva una alta recompensa promedio).

La rápida identificación de la opción óptima en los mundos escasos simulados es congruente con la etología, pues cuando el animal se enfrenta a escasez de alimento en la naturaleza, debe hacer una rápida y correcta identificación del alimento que le sea más benéfico energéticamente, a fin de establecerlo como su alimento de preferencia y aprovecharlo al máximo.

Finalmente, en cuanto al análisis del porcentaje de consumo de opciones no óptimas (o porcentaje de sobrevivencia) experimentado por el agente simulado bajo las diversas configuraciones ambientales a las que se enfrentó, debe destacarse la diferencia (20.91 %) entre el porcentaje de sobrevivencia promedio de los algoritmos de aprendizaje que lograron la máxima recompensa promedio por corrida y el de la selección aleatoria sin aprendizaje (comparativo SinApR), tanto de manera global (38.43 % para los algoritmos y 59.34 % para SinApR; tabla 5.36, p. 91) como desglosada para los diferentes mundos (tabla 5.37).

Desde una perspectiva global, los algoritmos de aprendizaje presentan una selección de la opción óptima más alta que la que se origina bajo circunstancias aleatorias, por lo cual siempre será mejor la toma de decisiones con aplicación de aprendizaje a aquella donde se tenga un comportamiento que no se base en el conocimiento de las opciones que el mundo ofrezca.

También en relación a la sobrevivencia, se puede concluir que los mundos abundantes y la distribución por zonas favorecen la minimización del consumo de opciones no óptimas. En especial, cabe destacar el hecho de que los mundos escasos distribuidos aleatoriamente enfrentan la tarea más difícil en cuanto a la disponibilidad de las opciones solicitadas y que la distribución por zonas en estos mundos viene a plantear una mejoría en su desempeño, lo cual aplica también en menor grado en mundos con opciones en abundancia.

Así, con lo expuesto para diferentes rubros, se considera que ha sido posible hacer un análisis de los algoritmos de Aprendizaje por Reforzamiento implementados, viendo además su desempeño ante excepciones por el ambiente, situaciones que generalmente no se experimentan.

En relación a esto, cabe mencionar que, bajo muchos experimentos, la capacidad de identificación de la opción óptima del agente fue mayor a su capacidad de maximización de recompensa, pero las características de los ambientes en los experimentos hace que la oportunidad de optimizar la recompensa se pierda.

Trabajo futuro

Dentro de las tareas a experimentar y/o mejorar se considera la implementación de un aprendizaje multiconductual, entendido como la aplicación de aprendizaje sobre varios módulos conductuales, a diferencia del único comportamiento que actualmente se especializa mediante técnicas de aprendizaje. Se considera que el aprendizaje multiconductual afectaría en mayor grado la dinámica de la red y, en consecuencia, los patrones de comportamiento que se obtengan mostrarían más variabilidad, obteniéndose así un conjunto de experimentaciones más amplio.

Otra tarea deseable es que el ambiente simulado refleje todas las conductas que conforman la Red de Comportamiento del agente, con lo que la simulación sería más completa. Con esto, la toma de decisiones, que implica las percepciones del estado del mundo, consideraría la existencia/inexistencia de situaciones en el ambiente que sean propicias para la activación de cada conducta (por ejemplo, para descansar el agente necesitaría estar en un lugar apropiado del ambiente para hacerlo). Con un ambiente simulado de este tipo se mostrarían las acciones ejecutadas por el agente en todo momento.

Por otro lado, es deseable experimentar con recompensas no booleanas, implicando que las opciones del módulo m_{ApR} no desaparezcan del ambiente al ser consumidas por el agente; esto además sería un indicador de la deseabilidad de la acción. En el caso de aprendizaje multiconductual, la consumación paulatina de recursos se aplicaría a toda conducta sujeta a aprendizaje.

Una extensión que sería un buen marco de experimentación lo constituye la versión de la herramienta para un sistema multi-agentes donde exista competencia y/o cooperación por los recursos del ambiente, lo que haría más real la investigación de animats. Computacionalmente, esto re-

quiere la modificación de ABC_ApR, que actualmente modela a un sólo agente, para que pueda soportar la co-existencia de varios agentes.

Se proponen dos tipos de aprendizaje para el sistema multi-agentes: 1) aprendizaje individual, resultante de la propia experiencia del agente (ya implementado) y, 2) aprendizaje grupal, implicando que a partir de información obtenida de todos los agentes, se establezca una información pública que sea utilizada por cada agente para reafirmar o modificar su aprendizaje individual. El aprendizaje grupal sería una forma de comunicación entre los agentes, pero además el sistema requeriría del establecimiento de otros tipos de interacciones que permitan la co-existencia de los agentes, considerando que éstos se verán inmersos en situaciones de competencia por recursos, o de cooperación para la realización de metas conjuntas.

Así mismo, el sistema deberá contar con mecanismos que coordinen la asignación de recursos ante situaciones conflictivas. Por ejemplo, ante la solicitud por parte de dos o más agentes de una opción percibida en el ambiente, con todos los agentes ubicados más o menos a la misma distancia con respecto a la opción solicitada, una solución basada en la etología sería el pre-establecimiento de jerarquías entre los agentes, a semejanza de las jerarquías sociales en las tropas de monos, con lo que se establecería una *competencia directa*, favoreciendo un acceso jerárquico a la opción; en cambio, si esta misma situación se diera con un agente ubicado muy cerca de la opción deseada y los demás a una distancia mayor, la solución sería asignar el recurso solicitado al agente más próximo, sin considerar su jerarquía social (*competencia por explotación*).

En el contexto de la simulación de los monos aulladores, un sistema multi-agentes sería de gran importancia, al permitir analizarlos en conductas de interacción social, que involucren relaciones directas entre los agentes. Por otro lado, un experimento interesante que implicaría aprendizaje grupal está en los mundos distribuidos en zonas, donde un mono podría hallar la zona óptima y, mediante información pública, guiar hacia ella a la tropa, lo cual es particularmente importante en mundos con escasez de alimento, donde un tiempo grande de exploración conduce al consumo de opciones no óptimas. Sin embargo, a semejanza de lo reportado por los etólogos [Dom04], esto implicaría una disminución en la ganancia alimenticia del individuo que, habiendo hallado un parche, deberá competir por los recursos alimenticios y compartirlos con el resto de la tropa.

Por otro lado, también en el contexto de simulación de monos aulladores, existen observaciones relativas a las rutas de forrajeo de estos animales en su hábitat, que implican la identificación del mismo mediante el etiquetado de todas las fuentes de alimentación (árboles) disponibles en el medio, para poder establecer los lugares donde los monos realizan sus actividades y hacer un seguimiento de sus desplazamientos y preferencias alimenticias [Arr99].

Estas rutas de forrajeo, en términos de ABC_ApR que ya plantea la selección de alimentos por parte del mono simulado, se considera viable de investigar, pudiendo implicar incluso el uso de un ambiente simulado más apegado a lo real, que comprenda una distribución de las fuentes alimenticias parecida a la del mundo del animal.

Apéndice A

Modelo ABC_RodríguezLuna

```
### Archivo..: /modelo_Luna.tcl

### Parametros
set PI 20.0
set TETA 15
set SIGMA 20.0
set GAMMA 70.0
set DELTA 50.0

### Proposiciones Variables en el Mundo
set PVM {{comida_ausente comida_presente} {suenio_ausente suenio_presente}
              {sombra_ausente sombra_presente}}
### Intervalo en el que se va a cambiar el Mundo
set interval 15
### Estado inicial del mundo
set PV {sombra_presente suenio_ausente comida_presente}
### Metas del agente
set M {}

### Modulos conductuales
define_mc descansar \
        {sombra_presente} \
         {cansancio} \
        {} \
        {global cansancio hambre inquietud interval
                set cansancio(peso) [expr [valor_campo cansancio peso]  *1.85 ]
                set hambre(peso) [expr [valor_campo hambre peso]  * 2.0 ]
                set inquietud(peso) [expr [valor_campo inquietud peso] * 2.1 ]
                set interval 12}

define_mc alimentar \
        {comida_presente} \
        {hambre} \
        {} \
        {global cansancio hambre inquietud interval
                set cansancio(peso) [expr [valor_campo cansancio peso]  * 2.0 ]
                set hambre(peso) [expr [valor_campo hambre peso]  * 1.03 ]
                set inquietud(peso) [expr [valor_campo inquietud peso] * 3.2 ]
                set interval 15}

define_mc explorar \
        {suenio_ausente} \
        {inquietud} \
        {} \
        {global cansancio hambre inquietud interval
                set cansancio(peso) [expr [valor_campo cansancio peso]  * 2.0 ]
                set hambre(peso) [expr [valor_campo hambre peso]  * 2.7 ]
                set inquietud(peso) [expr [valor_campo inquietud peso] * 0.5 ]
                set interval 15}

### Motivaciones         -> 0 Permanente, 1 Temporal
define_motivacion cansancio {0.60} {0}
define_motivacion hambre {0.25} {0}
define_motivacion inquietud {0.15} {0}

### Conducta sobre la que se lanza el modulo de aprendizaje
global OPTAPR; set OPTAPR alimentar
```

Listado A.1: Código del modelo ABC_RodríguezLuna

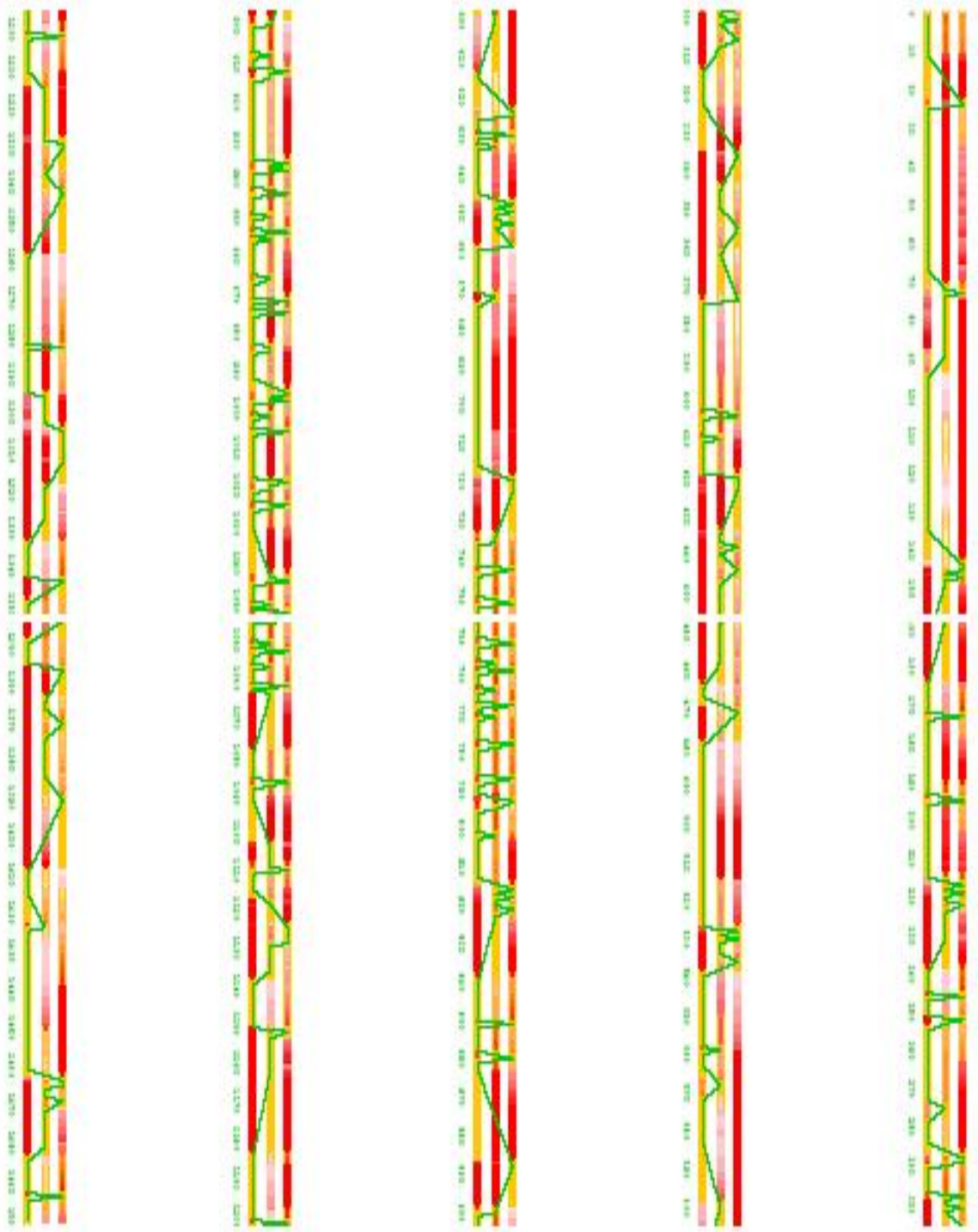

Figura A.1: Gráfica de activación en el tiempo del modelo ABC_RodríguezLuna

Ejemplo de ABC_RodríguezLuna - Algoritmo Softmax, mundo abundante aleatorio

Caso 2: 0.60 0.40 (optima: 1) Sem.Aleat. 0.1

It.	R/A_1	R/A_2	REC.	OPT.	Conv.	Sol/Obt	timeABC
1	(0/1)	(0/0)	0	1 *	0	0/0	(72)
2	(0/1)	(0/1)	0.0	0 *	2/2	1/1	(74)
3	(1/2)	(0/1)	33.33	1 *	2	0/0	(75)
4	(2/3)	(0/1)	50.0	1 *	2	0/0	(77)
5	(2/3)	(1/2)	60.0	0 *	5/5	1/1	(82)
6	(3/4)	(1/2)	66.67	1 *	5	0/0	(84)
7	(3/5)	(1/2)	57.14	1 *	5	0/0	(146)
8	(4/6)	(1/2)	62.5	1 *	5	0/0	(147)
9	(4/7)	(1/2)	55.56	1 *	5	0/0	(149)
10	(5/8)	(1/2)	60.0	1 *	5	0/0	(216)
11	(6/9)	(1/2)	63.64	1 *	5	0/0	(217)
12	(7/10)	(1/2)	66.67	1 *	5	0/0	(221)
13	(8/11)	(1/2)	69.23	1 *	5	0/0	(252)
14	(8/12)	(1/2)	64.29	1 *	5	0/0	(253)
15	(9/13)	(1/2)	66.67	1 *	5	0/0	(255)
16	(10/14)	(1/2)	68.75	1 *	5	0/0	(257)
17	(10/15)	(1/2)	64.71	1 *	5	0/0	(266)
18	(11/16)	(1/2)	66.67	1 *	5	0/0	(300)
19	(11/17)	(1/2)	63.16	1 *	5	0/0	(302)
20	(12/18)	(1/2)	65.0	1 *	5	0/0	(303)
21	(13/19)	(1/2)	66.67	1 *	5	0/0	(305)
22	(14/20)	(1/2)	68.18	1 *	5	0/0	(309)
23	(15/21)	(1/2)	69.57	1 *	5	0/0	(314)
24	(16/22)	(1/2)	70.83	1 *	5	0/0	(341)
25	(17/23)	(1/2)	72.0	1 *	5	0/0	(347)
26	(18/24)	(1/2)	73.08	1 *	5	0/0	(354)
27	(19/25)	(1/2)	74.07	1 *	5	0/0	(360)
28	(20/26)	(1/2)	75.0	1 *	5	0/0	(362)
29	(20/27)	(1/2)	72.41	1 *	5	0/0	(365)
30	(21/28)	(1/2)	73.33	1 *	5	0/0	(370)
31	(21/29)	(1/2)	70.97	1 *	5	0/0	(405)
32	(22/30)	(1/2)	71.88	1 *	5	0/0	(415)
33	(23/31)	(1/2)	72.73	1 *	5	0/0	(425)
34	(24/32)	(1/2)	73.53	1 *	5	0/0	(464)
35	(24/33)	(1/2)	71.43	1 *	5	0/0	(473)
36	(25/34)	(1/2)	72.22	1 *	5	0/0	(495)
37	(25/35)	(1/2)	70.27	1 *	5	0/0	(496)
38	(26/36)	(1/2)	71.05	1 *	5	0/0	(497)
39	(26/37)	(1/2)	69.23	1 *	5	0/0	(503)
40	(27/38)	(1/2)	70.0	1 *	5	0/0	(514)
41	(28/39)	(1/2)	70.73	1 *	5	0/0	(536)
42	(29/40)	(1/2)	71.43	1 *	5	0/0	(589)
43	(29/41)	(1/2)	69.77	1 *	5	0/0	(591)
44	(30/42)	(1/2)	70.45	1 *	5	0/0	(599)
45	(30/43)	(1/2)	68.89	1 *	5	0/0	(625)
46	(31/44)	(1/2)	69.57	1 *	5	0/0	(660)
47	(32/45)	(1/2)	70.21	1 *	5	0/0	(691)
48	(33/46)	(1/2)	70.83	1 *	5	0/0	(709)
49	(33/47)	(1/2)	69.39	1 *	5	0/0	(710)
50	(33/48)	(1/2)	68.0	1 *	5	0/0	(712)
51	(34/49)	(1/2)	68.63	1 *	5	0/0	(714)
52	(34/50)	(1/2)	67.31	1 *	5	0/0	(717)
53	(35/51)	(1/2)	67.92	1 *	5	0/0	(732)
54	(36/52)	(1/2)	68.52	1 *	5	0/0	(795)
55	(36/53)	(1/2)	67.27	1 *	5	0/0	(796)
56	(36/54)	(1/2)	66.07	1 *	5	0/0	(805)
57	(36/55)	(1/2)	64.91	1 *	5	0/0	(811)
58	(36/56)	(1/2)	63.79	1 *	5	0/0	(828)
59	(36/57)	(1/2)	62.71	1 *	5	0/0	(830)
60	(37/58)	(1/2)	63.33	1 *	5	0/0	(834)
61	(38/59)	(1/2)	63.93	1 *	5	0/0	(841)
62	(39/60)	(1/2)	64.52	1 *	5	0/0	(848)

Ejemplo RodríguezLuna - Continua . . .

```
63      (40/61) (1/2)   65.08   1 *     5       0/0     (856)
64      (41/62) (1/2)   65.63   1 *     5       0/0     (866)
65      (42/63) (1/2)   66.15   1 *     5       0/0     (877)
66      (42/64) (1/2)   65.15   1 *     5       0/0     (879)
67      (43/65) (1/2)   65.67   1 *     5       0/0     (881)
68      (44/66) (1/2)   66.18   1 *     5       0/0     (947)
69      (45/67) (1/2)   66.67   1 *     5       0/0     (957)
70      (46/68) (1/2)   67.14   1 *     5       0/0     (961)
71      (47/69) (1/2)   67.61   1 *     5       0/0     (963)
72      (48/70) (1/2)   68.06   1 *     5       0/0     (965)
73      (48/71) (1/2)   67.12   1 *     5       0/0     (972)
74      (49/72) (1/2)   67.57   1 *     5       0/0     (984)
75      (50/73) (1/2)   68.0    1 *     5       0/0     (1032)
76      (51/74) (1/2)   68.42   1 *     5       0/0     (1034)
77      (52/75) (1/2)   68.83   1 *     5       0/0     (1037)
78      (52/76) (1/2)   67.95   1 *     5       0/0     (1056)
79      (52/77) (1/2)   67.09   1 *     5       0/0     (1057)
80      (53/78) (1/2)   67.5    1 *     5       0/0     (1062)
81      (53/79) (1/2)   66.67   1 *     5       0/0     (1080)
82      (54/80) (1/2)   67.07   1 *     5       0/0     (1082)
83      (55/81) (1/2)   67.47   1 *     5       0/0     (1085)
84      (56/82) (1/2)   67.86   1 *     5       0/0     (1104)
85      (57/83) (1/2)   68.24   1 *     5       0/0     (1110)
86      (58/84) (1/2)   68.6    1 *     5       0/0     (1117)
87      (58/85) (1/2)   67.82   1 *     5       0/0     (1126)
88      (58/85) (1/3)   67.05   0 *     5/88    0/1     (1145)
89      (59/86) (1/3)   67.42   1 *     5       0/0     (1164)
90      (60/87) (1/3)   67.78   1 *     5       0/0     (1167)
91      (61/88) (1/3)   68.13   1 *     5       0/0     (1177)
92      (62/89) (1/3)   68.48   1 *     5       0/0     (1187)
93      (62/90) (1/3)   67.74   1 *     5       0/0     (1192)
94      (63/91) (1/3)   68.09   1 *     5       0/0     (1212)
95      (64/92) (1/3)   68.42   1 *     5       0/0     (1216)
96      (65/93) (1/3)   68.75   1 *     5       0/0     (1224)
97      (66/94) (1/3)   69.07   1 *     5       0/0     (1274)
98      (66/95) (1/3)   68.37   1 *     5       0/0     (1280)
99      (67/96) (1/3)   68.69   1 *     5       0/0     (1320)
100     (67/97) (1/3)   68.0    1 *     5       0/0     (1322)
101     (68/98) (1/3)   68.32   1 *     5       0/0     (1329)
102     (68/99) (1/3)   67.65   1 *     5       0/0     (1353)
103     (68/100)(1/3)   66.99   1 *     5       0/0     (1365)
104     (68/101)(1/3)   66.35   1 *     5       0/0     (1425)
105     (69/102)(1/3)   66.67   1 *     5       0/0     (1430)
106     (70/103)(1/3)   66.98   1 *     5       0/0     (1446)
107     (70/104)(1/3)   66.36   1 *     5       0/0     (1484)

Recompensa Acumulada 71 Recompensa Promedio 66.36
Factor XLearn: 1.11
Solicita óptima 105 (98.13%) Obtiene óptima 104 (97.2%)
Convergencia 6/89       Sobrevivencia (no-óptimos) 3 (2.8%)

Tiempo promedio y porcentaje por módulo de red:
                t_prom    %
explorar        2.83    15.07
alimentar       4.73    26.20
descansar       11.29   58.73

Matriz de Probabilidades
        explorar alimentar descansar
explorar        0.00    0.55    0.45
alimentar       0.51    0.00    0.49
descansar       0.49    0.51    0.00
```

Listado A.2: Corrida ejemplo - RodríguezLuna

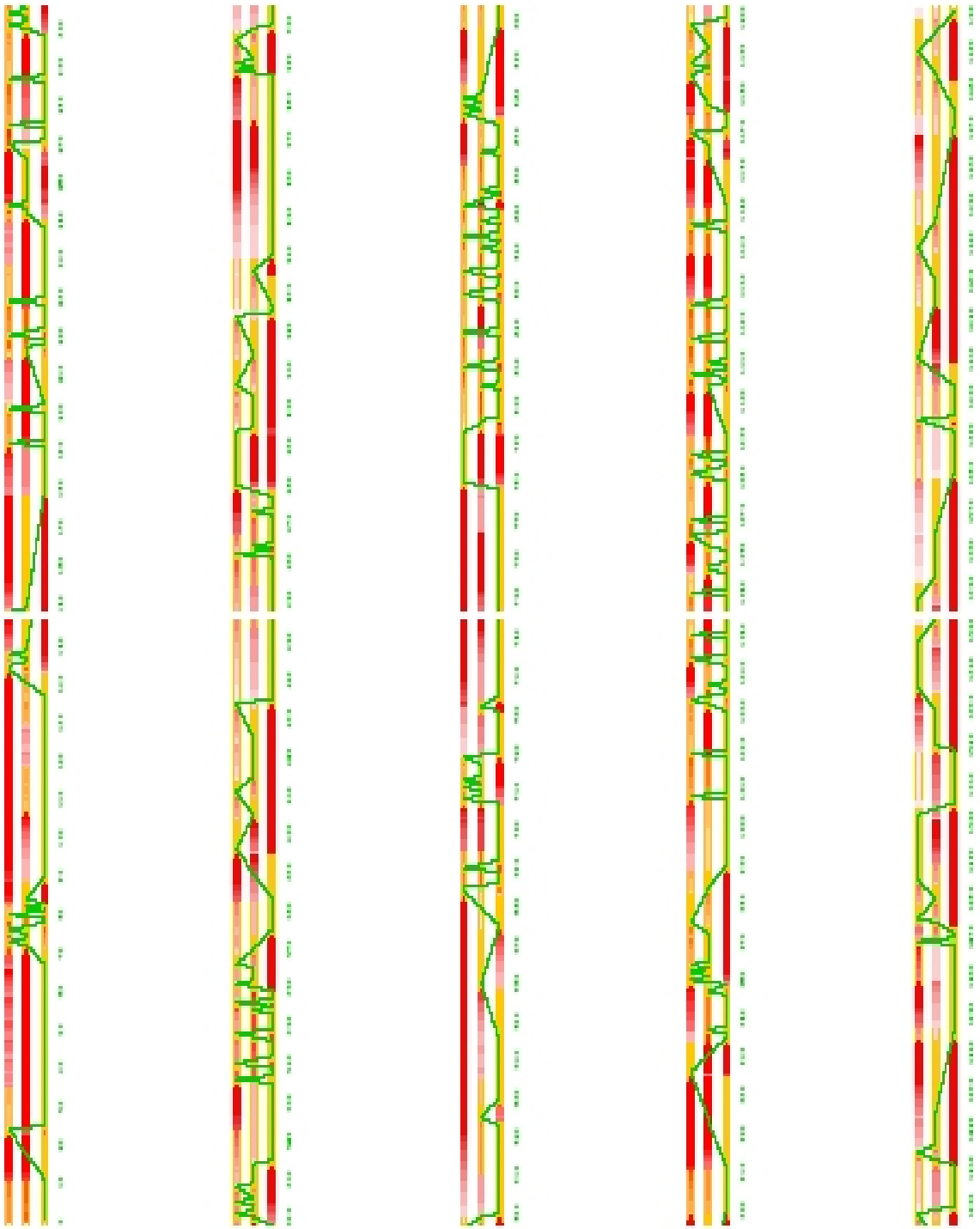

Figura A.2: Gráfica de activación en el tiempo del ejemplo del modelo ABC_RodríguezLuna

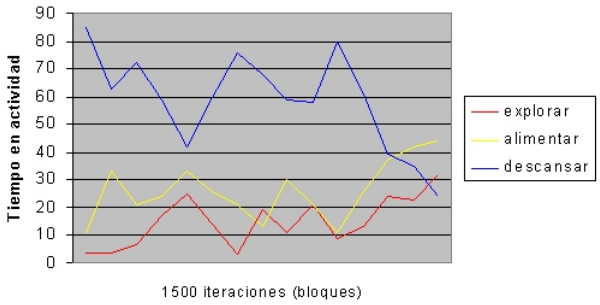

Figura A.3: Patrón de actividades del ejemplo del modelo ABC_RodríguezLuna

Corrida general del ejemplo de ABC_RodríguezLuna

No. Opción: 2 Caso probabilístico: 2 (0.60, 0.40) Mundo abundante aleatorio

Alg.	R_Prom	R_Total	Iter.	XLearn	Conv.	R/A_1	R/A_2	%Sol*	%Obt*	%Sobreviv.
Bandido	60.55	66	109	1.01	19/106	61/96	5/13	91.74	88.07	11.93
Greedy	42.86	42	98	0.71	-/-	3/10	39/88	3.06	10.2	89.8
Egreedy	61.74	71	115	1.03	84/84	70/111	1/4	97.39	96.52	3.48
Softmax	66.36	71	107	1.11	6/89	70/104	1/3	98.13	97.2	2.8
LRP	52.73	58	110	0.88	110/110	41/68	17/42	61.82	61.82	38.18
LR	58.42	59	101	0.97	76/76	50/78	9/23	77.23	77.23	22.77
IE	53.15	59	111	0.89	13/90	59/107	0/4	97.3	96.4	3.6
RC	54.05	60	111	0.90	13/90	60/108	0/3	98.2	97.3	2.7
SinApR	56.0	56	100	0.93	-/100	34/50	22/50	0.0	50.0	50.0
Optimo	63.73	65	102	1.06	1/1	65/102	0/0	100.0	100.0	0.0

Listado A.3: Corrida general ejemplo - RodríguezLuna

El listado A.3 es un resumen del experimento al que pertenece la corrida ejemplo de la simulación ABC_RodríguezLuna. La gráfica A.4 muestra las curvas de aprendizaje y de convergencia de los ocho algoritmos y los dos comparativos implementados, en tanto que la figura A.5 presenta el graficado acumulado de las curvas de aprendizaje generadas bajo la configuración del experimento.

Analizando el resúmen de la corrida, se puede ver que el algoritmo Greedy no convergió al óptimo, pues sólo presenta un 3.06 % de solicitud de la acción óptima (corroborar con su gráfica de convergencia, en la figura A.4); esto repercute en un bajo aprendizaje, al no lograr maximizar la recompensa promedio. En tanto, LRP y LR también exhiben un desempeño bajo, aunque la gráfica de convergencia de LR demuestra que logra converger y alcanza una recompensa promedio cercana a la esperada. Por su parte, IE y RC, a pesar de presentar un buen porcentaje de convergencia, no logran una buena recompensa promedio.

Por otro lado, debe notarse que existen variaciones entre lo solicitado y lo obtenido en el ambiente, la mayoría en forma negativa (Bandido, Egreedy, Softmax, IE y RC) y en contadas ocasiones, positivamente (ver Greedy, donde el ambiente ayuda a seleccionar la acción óptima).

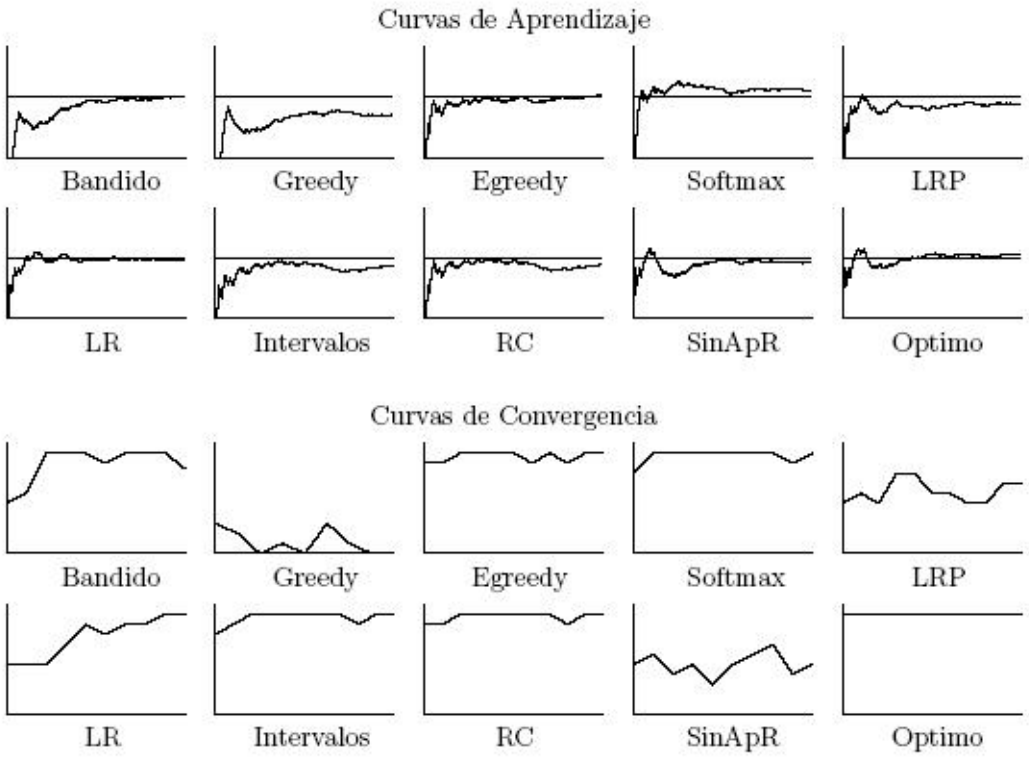

Figura A.4: Curvas de aprendizaje y de convergencia de la corrida general - ABC_RodríguezLuna

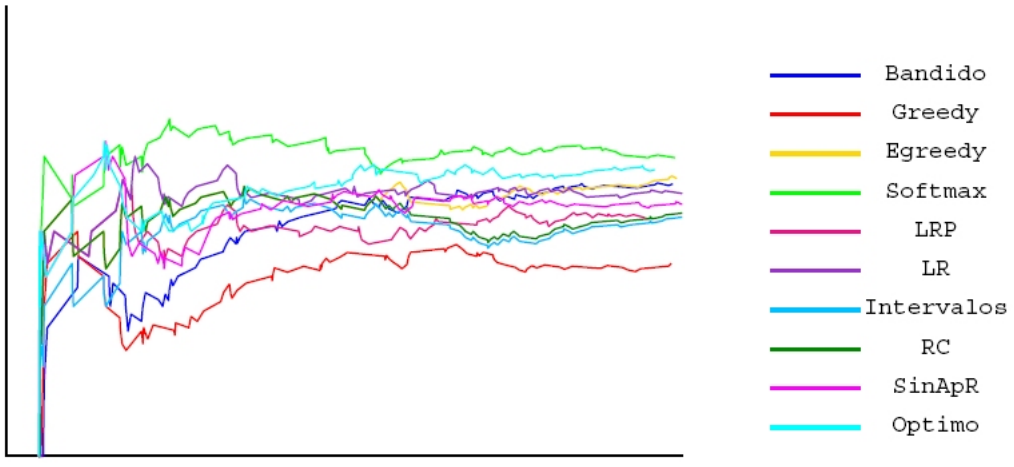

Figura A.5: Curvas de aprendizaje de la corrida general - ABC_RodríguezLuna

Modelo ABC_DomingoBalcells

```
### Archivo..: /modelo_Balcells.tcl

### Parametros
set PI 20.0
set TETA 15
set SIGMA 20.0
set GAMMA 70.0
set DELTA 50.0

### Proposiciones Variables en el Mundo
set PVM {{comida_ausente comida_presente} {suenio_ausente suenio_presente}
                {mono_ausente mono_presente} {sombra_ausente sombra_presente}}
### Intervalo en el que se va a cambiar el Mundo
set interval 20
### Estado inicial del mundo
set PV {sombra_presente suenio_ausente comida_presente mono_ausente}
### Metas del agente
set M {}

### Modulos conductuales
define_mc descansar \
        {sombra_presente} \
         {cansancio} \
        {} \
        {global cansancio hambre inquietud interval
                set cansancio(peso) [expr [valor_campo cansancio peso]  * 1.0 ]
                set hambre(peso) [expr [valor_campo hambre peso]  * 1.3 ]
                set aburrimiento(peso) [expr [valor_campo aburrimiento peso] * 0.1 ]
                set inquietud(peso) [expr [valor_campo inquietud peso] * 1.0 ]
                set interval 21 }

define_mc alimentar \
        {comida_presente} \
        {hambre} \
        {} \
        {global cansancio hambre inquietud interval
                set cansancio(peso) [expr [valor_campo cansancio peso]  * 1.0 ]
                set hambre(peso) [expr [valor_campo hambre peso]  * 0.1 ]
                set aburrimiento(peso) [expr [valor_campo aburrimiento peso] * 0.4 ]
                set inquietud(peso) [expr [valor_campo inquietud peso] * 1.55 ]
                set interval 16 }

define_mc explorar \
        {suenio_ausente} \
        {inquietud} \
        {} \
        {global cansancio hambre inquietud interval
                set cansancio(peso) [expr [valor_campo cansancio peso]  * 1.0 ]
                set hambre(peso) [expr [valor_campo hambre peso]  * 1.0 ]
                set aburrimiento(peso) [expr [valor_campo aburrimiento peso] * 0.1 ]
                set inquietud(peso) [expr [valor_campo inquietud peso] * 0.2 ]
                set interval 10 }

define_mc social \
        {mono_presente} \
        {aburrimiento} \
        {} \
        {global cansancio hambre inquietud interval
                set cansancio(peso) [expr [valor_campo cansancio peso]  * 1.0 ]
                set hambre(peso) [expr [valor_campo hambre peso]  * 1.0 ]
                set aburrimiento(peso) [expr [valor_campo aburrimiento peso] * 0.1 ]
                set inquietud(peso) [expr [valor_campo inquietud peso] * 1.0 ]
                set interval 2 }

### Motivaciones         -> 0 Permanente, 1 Temporal
define_motivacion cansancio {0.7753} {0}
define_motivacion hambre {0.135} {0}
define_motivacion inquietud {0.058} {0}
define_motivacion aburrimiento {0.0317} {0}

### Conducta sobre la que se lanza el modulo de aprendizaje
global OPTAPR; set OPTAPR alimentar
```

Listado A.4: Código del modelo ABC_DomingoBalcells

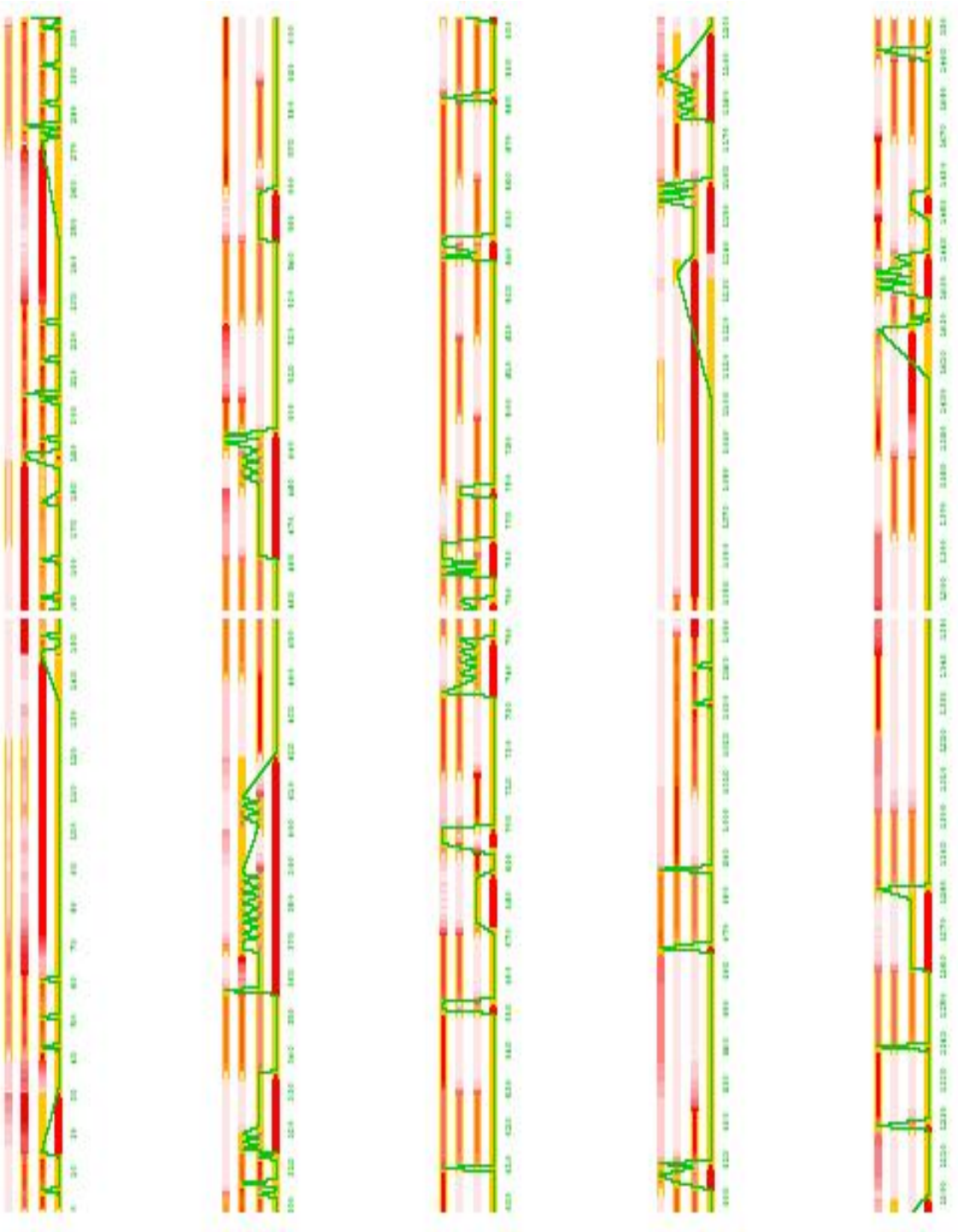

Figura A.6: Gráfica de activación en el tiempo del modelo ABC_DomingoBalcells

Ejemplo de ABC_DomingoBalcells - Algoritmo Greedy, mundo abundante aleatorio

```
Caso 1: 0.90 0.10 (optima: 1)   Sem.Aleat. 0.1
.
It.     R/A_1   R/A_2   REC.    OPT.    Conv.   Sol/Obt timeABC
1       (0/1)   (0/0)   0       1 *     0       0/0     (5)
2       (0/1)   (0/1)   0.0     0 *     2/2     1/1     (15)
3       (1/2)   (0/1)   33.33   1 *     2       0/0     (63)
4       (2/3)   (0/1)   50.0    1 *     2       0/0     (147)
5       (3/4)   (0/1)   60.0    1 *     2       0/0     (148)
6       (4/5)   (0/1)   66.67   1 *     2       0/0     (151)
7       (5/6)   (0/1)   71.43   1 *     2       0/0     (161)
8       (6/7)   (0/1)   75.0    1 *     2       0/0     (171)
9       (6/8)   (0/1)   66.67   1 *     2       0/0     (181)
10      (7/9)   (0/1)   70.0    1 *     2       0/0     (194)
11      (8/10)  (0/1)   72.73   1 *     2       0/0     (204)
12      (9/11)  (0/1)   75.0    1 *     2       0/0     (214)
13      (10/12) (0/1)   76.92   1 *     2       0/0     (225)
14      (11/13) (0/1)   78.57   1 *     2       0/0     (273)
15      (12/14) (0/1)   80.0    1 *     2       0/0     (275)
16      (13/15) (0/1)   81.25   1 *     2       0/0     (283)
17      (14/16) (0/1)   82.35   1 *     2       0/0     (293)
18      (15/17) (0/1)   83.33   1 *     2       0/0     (303)
19      (16/18) (0/1)   84.21   1 *     2       0/0     (313)
20      (17/19) (0/1)   85.0    1 *     2       0/0     (316)
21      (18/20) (0/1)   85.71   1 *     2       0/0     (318)
22      (19/21) (0/1)   86.36   1 *     2       0/0     (321)
23      (20/22) (0/1)   86.96   1 *     2       0/0     (323)
24      (20/23) (0/1)   83.33   1 *     2       0/0     (325)
25      (21/24) (0/1)   84.0    1 *     2       0/0     (326)
26      (22/25) (0/1)   84.62   1 *     2       0/0     (328)
27      (23/26) (0/1)   85.19   1 *     2       0/0     (330)
28      (24/27) (0/1)   85.71   1 *     2       0/0     (332)
29      (25/28) (0/1)   86.21   1 *     2       0/0     (334)
30      (26/29) (0/1)   86.67   1 *     2       0/0     (359)
31      (27/30) (0/1)   87.1    1 *     2       0/0     (363)
32      (28/31) (0/1)   87.5    1 *     2       0/0     (366)
33      (29/32) (0/1)   87.88   1 *     2       0/0     (371)
34      (30/33) (0/1)   88.24   1 *     2       0/0     (374)
35      (31/34) (0/1)   88.57   1 *     2       0/0     (376)
36      (32/35) (0/1)   88.89   1 *     2       0/0     (379)
37      (33/36) (0/1)   89.19   1 *     2       0/0     (382)
38      (34/37) (0/1)   89.47   1 *     2       0/0     (384)
39      (35/38) (0/1)   89.74   1 *     2       0/0     (387)
40      (36/39) (0/1)   90.0    1 *     2       0/0     (401)
41      (37/40) (0/1)   90.24   1 *     2       0/0     (404)
42      (38/41) (0/1)   90.48   1 *     2       0/0     (407)
43      (39/42) (0/1)   90.7    1 *     2       0/0     (462)
44      (40/43) (0/1)   90.91   1 *     2       0/0     (481)
45      (41/44) (0/1)   91.11   1 *     2       0/0     (484)
46      (42/45) (0/1)   91.3    1 *     2       0/0     (487)
47      (43/46) (0/1)   91.49   1 *     2       0/0     (490)
48      (44/47) (0/1)   91.67   1 *     2       0/0     (493)
49      (45/48) (0/1)   91.84   1 *     2       0/0     (495)
50      (46/49) (0/1)   92.0    1 *     2       0/0     (546)
51      (47/50) (0/1)   92.16   1 *     2       0/0     (548)
52      (48/51) (0/1)   92.31   1 *     2       0/0     (550)
53      (49/52) (0/1)   92.45   1 *     2       0/0     (552)
54      (50/53) (0/1)   92.59   1 *     2       0/0     (554)
55      (50/54) (0/1)   90.91   1 *     2       0/0     (556)
56      (51/55) (0/1)   91.07   1 *     2       0/0     (557)
57      (52/56) (0/1)   91.23   1 *     2       0/0     (559)
58      (53/57) (0/1)   91.38   1 *     2       0/0     (674)
59      (54/58) (0/1)   91.53   1 *     2       0/0     (676)
60      (55/59) (0/1)   91.67   1 *     2       0/0     (678)
61      (56/60) (0/1)   91.8    1 *     2       0/0     (680)
62      (57/61) (0/1)   91.94   1 *     2       0/0     (682)
63      (58/62) (0/1)   92.06   1 *     2       0/0     (684)
64      (59/63) (0/1)   92.19   1 *     2       0/0     (686)
```

Ejemplo DomingoBalcells - Continua . . .

```
65      (60/64)         (0/1)   92.31   1 *     2       0/0     (739)
66      (61/65)         (0/1)   92.42   1 *     2       0/0     (742)
67      (62/66)         (0/1)   92.54   1 *     2       0/0     (744)
68      (62/67)         (0/1)   91.18   1 *     2       0/0     (747)
69      (63/68)         (0/1)   91.3    1 *     2       0/0     (748)
70      (64/69)         (0/1)   91.43   1 *     2       0/0     (757)
71      (65/70)         (0/1)   91.55   1 *     2       0/0     (759)
72      (66/71)         (0/1)   91.67   1 *     2       0/0     (841)
73      (67/72)         (0/1)   91.78   1 *     2       0/0     (903)
74      (68/73)         (0/1)   91.89   1 *     2       0/0     (908)
75      (69/74)         (0/1)   92.0    1 *     2       0/0     (1031)
76      (70/75)         (0/1)   92.11   1 *     2       0/0     (1040)
77      (71/76)         (0/1)   92.21   1 *     2       0/0     (1140)
78      (72/77)         (0/1)   92.31   1 *     2       0/0     (1141)
79      (73/78)         (0/1)   92.41   1 *     2       0/0     (1142)
80      (74/79)         (0/1)   92.5    1 *     2       0/0     (1144)
81      (75/80)         (0/1)   92.59   1 *     2       0/0     (1146)
82      (76/81)         (0/1)   92.68   1 *     2       0/0     (1148)
83      (77/82)         (0/1)   92.77   1 *     2       0/0     (1150)
84      (78/83)         (0/1)   92.86   1 *     2       0/0     (1152)
85      (79/84)         (0/1)   92.94   1 *     2       0/0     (1155)
86      (79/85)         (0/1)   91.86   1 *     2       0/0     (1157)
87      (80/86)         (0/1)   91.95   1 *     2       0/0     (1158)
88      (80/87)         (0/1)   90.91   1 *     2       0/0     (1178)
89      (81/88)         (0/1)   91.01   1 *     2       0/0     (1179)
90      (82/89)         (0/1)   91.11   1 *     2       0/0     (1181)
91      (83/90)         (0/1)   91.21   1 *     2       0/0     (1184)
92      (84/91)         (0/1)   91.3    1 *     2       0/0     (1187)
93      (85/92)         (0/1)   91.4    1 *     2       0/0     (1190)
94      (86/93)         (0/1)   91.49   1 *     2       0/0     (1193)
95      (87/94)         (0/1)   91.58   1 *     2       0/0     (1195)
96      (88/95)         (0/1)   91.67   1 *     2       0/0     (1198)
97      (89/96)         (0/1)   91.75   1 *     2       0/0     (1202)
98      (90/97)         (0/1)   91.84   1 *     2       0/0     (1260)
99      (91/98)         (0/1)   91.92   1 *     2       0/0     (1262)
100     (92/99)         (0/1)   92.0    1 *     2       0/0     (1264)
101     (93/100)        (0/1)   92.08   1 *     2       0/0     (1266)
102     (94/101)        (0/1)   92.16   1 *     2       0/0     (1268)
103     (95/102)        (0/1)   92.23   1 *     2       0/0     (1270)
104     (96/103)        (0/1)   92.31   1 *     2       0/0     (1272)
105     (97/104)        (0/1)   92.38   1 *     2       0/0     (1274)
106     (98/105)        (0/1)   92.45   1 *     2       0/0     (1276)
107     (99/106)        (0/1)   92.52   1 *     2       0/0     (1278)
108     (99/107)        (0/1)   91.67   1 *     2       0/0     (1428)
109     (100/108)       (0/1)   91.74   1 *     2       0/0     (1429)
110     (101/109)       (0/1)   91.82   1 *     2       0/0     (1431)
111     (102/110)       (0/1)   91.89   1 *     2       0/0     (1434)
112     (103/111)       (0/1)   91.96   1 *     2       0/0     (1437)
113     (104/112)       (0/1)   92.04   1 *     2       0/0     (1451)
114     (105/113)       (0/1)   92.11   1 *     2       0/0     (1453)
115     (106/114)       (0/1)   92.17   1 *     2       0/0     (1455)

Recompensa Acumulada 106        Recompensa Promedio 92.17
Factor XLearn: 1.02
Solicita óptima 114 (99.13%) Obtiene óptima 114 (99.13%)
Convergencia 3/3        Sobrevivencia (no-óptimos) 1 (0.87%)

Tiempo promedio y porcentaje por módulo de red:
                t_prom    %
social  1.70    3.40
explorar        2.05    5.60
alimentar       2.71    12.67
descansar       23.50   78.33

Matriz de Probabilidades
        social explorar alimentar descansar
social  0.00    0.07    0.40    0.53
explorar        0.05    0.00    0.76    0.20
alimentar       0.20    0.44    0.00    0.36
descansar       0.29    0.16    0.55    0.00
```

Listado A.5: Corrida ejemplo - DomingoBalcells

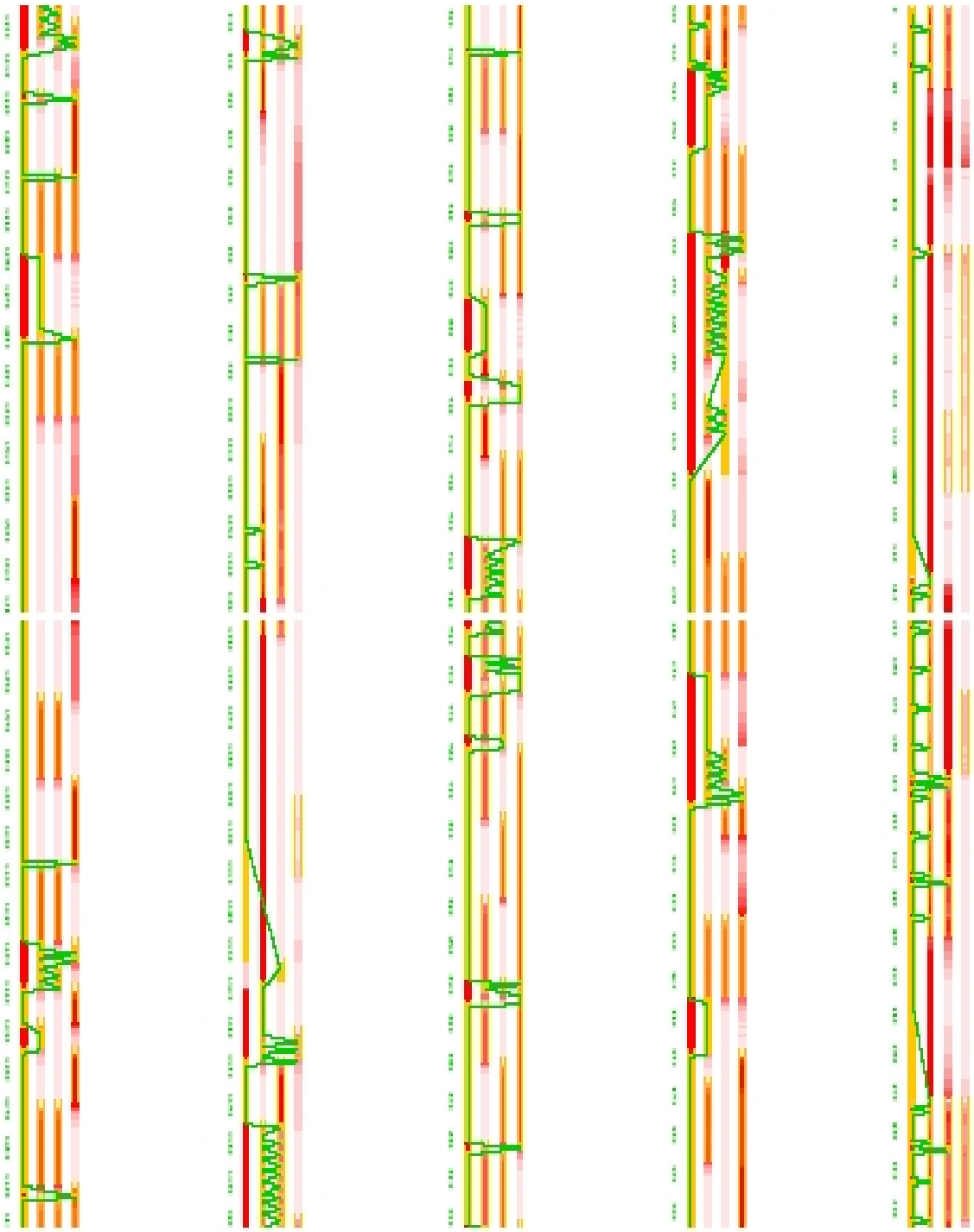

Figura A.7: Gráfica de activación en el tiempo del ejemplo del modelo ABC_DomingoBalcells

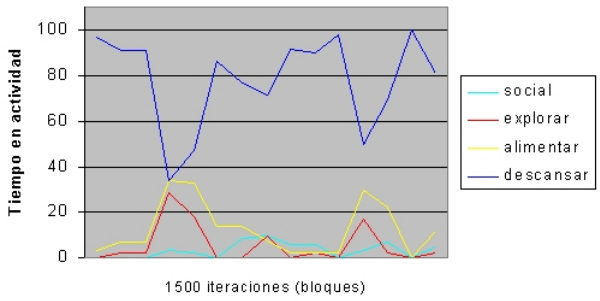

Figura A.8: Patrón de actividades del ejemplo del modelo ABC_DomingoBalcells

Corrida general del ejemplo de ABC_DomingoBalcells

No. Opción: 2 Caso probabilístico: 1 (0.90, 0.10) Mundo abundante aleatorio

Alg.	R_Prom	R_Total	Iter.	XLearn	Conv.	R/A_1	R/A_2	%Sol*	%Obt*	%Sobreviv
Bandido	87.2	109	125	0.97	7/7	109/122	0/3	97.6	97.6	2.4
Greedy	92.17	106	115	1.02	3/3	106/114	0/1	99.13	99.13	0.87
Egreedy	88.8	111	125	0.99	119/119	111/121	0/4	96.8	96.8	3.2
Softmax	90.99	101	111	1.01	6/6	101/109	0/2	98.2	98.2	1.8
LRP	79.66	94	118	0.89	116/116	92/100	2/18	84.75	84.75	15.25
LR	81.48	88	108	0.91	43/43	87/97	1/11	88.89	89.81	10.19
IE	90.68	107	118	1.01	13/13	107/116	0/2	98.31	98.31	1.69
RC	90.68	107	118	1.01	13/13	107/116	0/2	98.31	98.31	1.69
SinApR	60.96	89	146	0.68	-/142	87/96	2/50	0.0	65.75	34.25
Optimo	89.47	102	114	0.99	1/1	102/114	0/0	100.0	100.0	0.0

Listado A.6: Corrida general ejemplo - DomingoBalcells

El listado A.6 muestra la corrida general resumida del experimento al que pertenece la corrida del algoritmo Greedy que ejemplifica al modelo ABC_DomingoBalcells. Las curvas de aprendizaje y de convergencias de todos los demás algoritmos ejecutados y de los comparativos Óptimo y SinApR se muestran en la gráfica A.9, en tanto que la figura A.10 muestra todas las curvas de aprendizaje en una sola gráfica.

Como se puede observar, todos los algoritmos convergieron a la acción óptima, siendo LRP el más lento al hacerlo (ver esto en su gráfica de convergencia en A.9). Por otro lado, los algoritmos superaron tempranamente la recompensa esperada o quedaron próximos a lograrlo al terminar la corrida (a excepción de los métodos basados en vectores, que finalmente se quedan un 10 % debajo de lo esperado).

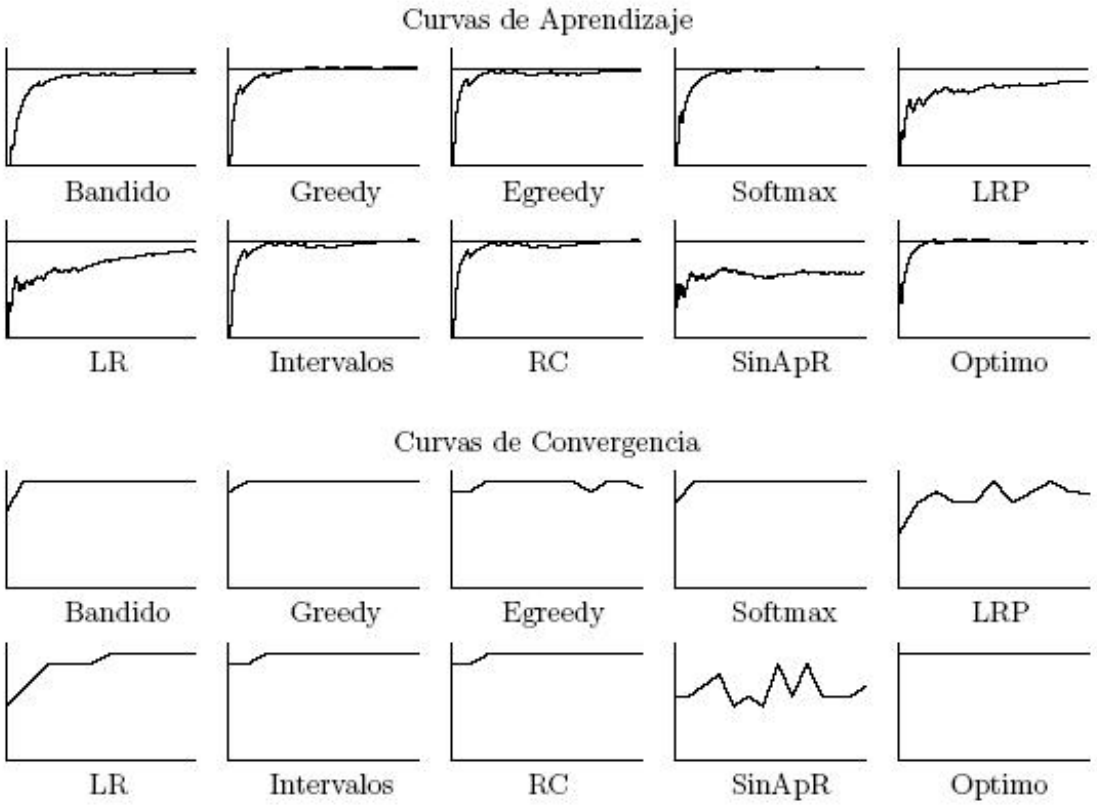

Figura A.9: Curvas de aprendizaje y de convergencia de la corrida general del ejemplo - modelo ABC_DomingoBalcells

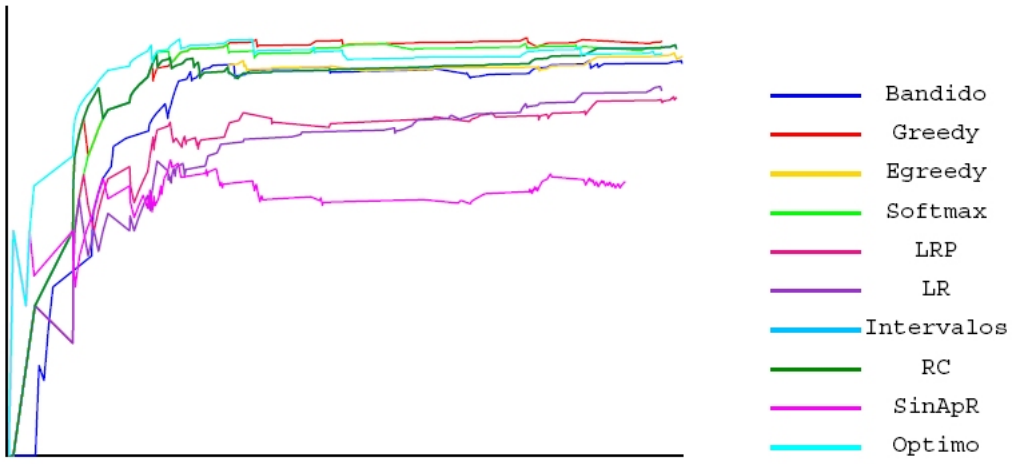

Figura A.10: Curvas de aprendizaje de la corrida general del ejemplo - modelo ABC_DomingoBalcells

Gráficas de tiempo de búsqueda

Los ejemplos presentados pertenecen ambos a mundos abundantes y tuvieron buen desempeño de convergencia y aprendizaje, por lo que sus gráficas de tiempo de búsqueda no son significativas, pues siempre encontraron la acción óptima en el mínimo tiempo.

La escasez de las acciones en el ambiente repercute en la necesidad de seleccionar otras acciones y esto se refleja en el tiempo de búsqueda de las opciones que son ejecutadas y en el porcentaje de obtención de la opción óptima.

Para mostrar que el ambiente modifica la selección de acción, se considera el caso del algoritmo Greedy para ABC_DomingoBalcells con el mismo número de acciones y caso probabilístico que el ejemplo presentado, pero variando el mundo de una cantidad abundante a una escasa, aunque sigue siendo de distribución aleatoria.

Ahora, el porcentaje de solicitud de la acción óptima para Greedy es de 99.34 %, con una obtención de la misma en tan sólo un 57.89 % (en la versión abundante, ambos porcentajes fueron de 99.13 %).

La gráfica de tiempo de búsqueda -por bloques- es presentada en la figura A.11. La línea amarilla representa el tiempo de búsqueda ideal, donde se encuentra la opción solicitada en el tiempo mínimo; este patron de búsqueda corresponde a los ejemplos, pues por lo regular se presenta en mundos abundantes. La línea azul representa el tiempo de búsqueda del ejemplo Greedy en un mundo escaso; como se puede observar, en este tipo de mundos existe variabilidad en los tiempos de percepción y ejecución de las opciones.

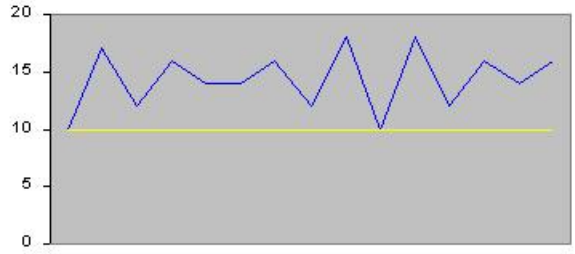

Figura A.11: Tiempo de búsqueda del ejemplo (Greedy) con un mundo escaso - modelo ABC_DomingoBalcells

Bibliografía

[Arr99] Arrioja Cruz, María del Rosario y Grajales Lagunes, María Concepción. Un sistema de información geográfica para la descripción cartográfica de la vegetación arbórea y de los patrones de forrajeo del mono aullador en la isla de Agaltepec, Catemaco, Veracruz. Tesina - Licenciatura en Informática, Universidad Veracruzana, 1999.

[Blu94] Blumberg, Bruce. Action-selection in Hamsterdam: lessons from ethology. En *Proceedings of the Third International Conference on the Simulation of Adaptive Behavior, Brighton*. MIT Press, Cambridge, MA, agosto 1994.

[Bro86] Brooks, Rodney. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, Vol. RA-2 No. 1 pags. 14–23, abril 1986.

[Car89] Carbonell, Jaime G., ed. *Machine Learning: paradigms and methods*. A Bradford Books / MIT Press, E.U.A., 1989.

[Cha85] Charniak, Eugene y McDermott, Drew. *Introduction to Artificial Intelligence*. Addison-Wesley, Reading, Mass., 1985.

[Con90] Connell, Jonathan H. Minimalist mobile robotics: a colony architecture for an artificial creature. Academic Press, Boston, 1990.

[Dom04] Domingo B., Cristina. Plasticidad conductual intergrupal del mono aullador (Alouatta palliata) en contexto de alimentación y/o conductas de cautiverio. Tesis de maestría, Universidad Veracruzana, 2004.

[Fra96] Franklin, Stan y Graesser, Art. Is it an agent or just a program?: a taxonomy for Autonomous Agents. En *Third International Workshop on Agent Theories, Architectures, and Languages*. Springer-Verlag, Berlin, Alemania, 1996.

[Gar97] Garcia Angélica, Martínez Manuel, Muñoz Angélica, Negrete José. Dos modelos de selección de acción para el estudio de la conducta de forrajeo del mono aullador (*Alouatta palliata*). V meeting of the International Biometric Society Network for Central America, The Caribbean, México and Venezuela, Xalapa, México, agosto, 1997.

[Goe97] Goetz, Philip y Walters, Deborah. The dynamics of recurrent Behavior Networks. *Adaptive Behavior*, Vol. 6 No. 2 pags. 247–283, 1997.

[Gue97a] Guerra Alejandro, Martínez Manuel, Montes Fernando. ABC, una herramienta para implantar modelos basados en redes de comportamiento. Primer Encuentro en Computación Enc97, SMCC-Sociedad Mexicana de Ciencias de la Computación, SMIA-Sociedad Mexicana de Inteligencia Artificial, Querétaro, México, 1997.

[Gue97b] Guerra Alejandro, Martínez Manuel, Montes Fernando. Un modelo de conductas animales basado en redes de comportamiento. V meeting of the International Biometric Society Network for Central America, The Caribbean, México and Venezuela, Xalapa, México, 1997.

[Gue97c] Guerra H., Alejandro. Agentes Autónomos y Selección de Acción, una perspectiva basada en Redes de Comportamiento. Tesis de maestría, Universidad Veracruzana, 1997.

[Gue98] Guerra Hernández, Alejandro. Modeling behavior: an action selection approach. Workshop on Distributed Simulation, Artificial Intelligence, and Virtual Environments, UNAM-Universidad Nacional Autónoma de México, UHD-University of Houston Downtown, Lania-Laboratorio Nacional de Informática Avanzada, A.C., México D.F., México, 1998.

[Kae93] Kaelbling, Leslie Pack. *Learning in embeded system.* A Bradford Book / MIT Press, Cambridge, MA, 1993.

[KyR90] Leslie Kaelbling y Rosenschein, Jeffrey. Action and planning in embedded agents. En *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, ed. Maes P. A Bradford Books / MIT Press, 1990.

[Lud76] Ludlow, A. The behavior of a model animal. *Behavior*, Vol. 58, 1976.

[Mae89] Maes, Pattie. The dynamics of Action Selection. En *Proceedings the IJCAI-89 Conference.* MIT Press, Detroit, 1989.

[Mae90a] Maes, Pattie. How do the right thing. *Conection Science, special issue on Hybrid Systems*, Vol. 1 No. 3 pags. 291–323, febrero 1990. Tambien en: MIT AILAB Memo 1180.

[Mae90b] Maes, Pattie. Situated agents can have goals. En *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, ed. Maes P. A Bradford Books / MIT Press, Febrero 1990. Tambien en: Special Issue on Autonomous Agents, Journal of Robotics and Autonumous Systems, Vol. 6 No. 1, junio 1990).

[Mae91a] Maes, Pattie. A bottom-up mechanism for behavior selection in an artificial creature. En *From Animals to Animats: Proceedings of the first International Conference on Simulation of Adaptive Behavior*, ed. Meyer J.A. y Wilson S.W., pags. 238–246. MIT Press, 1991.

[Mae91b] Maes, Pattie. Learning Behavior Networks from experience. En *From Animals to Animats: Proceedings of the first European Conference on Artificial Life*, pags. 238–246. MIT Press, 1991.

[Mae94] Maes, Pattie. Modeling adaptive Autonomous Agents. En *Artificial Life, an Overview*, ed. Langton C., Vol. 1, pags. 135–162. MIT Press, 1994.

[Mae95] Maes, Pattie. Artificial Life meets entertainment: lifelike Autonomous Agents. *Comunications of the ACM*, Vol. 38 No. 11 pags. 108–114, 1995. Special issue on New Horizons of Commercial and Industrial IA.

[Mar96a] Martínez Manuel. Proyecto Monots: una propuesta etorobótica para modelar la conducta de forrajeo del mono aullador (Alouatta palliata). Estudios Primatológicos en México, III. Universidad Veracruzana y Asociación Mexicana de Primatología, Xalapa, México, 1996.

[Mar96b] Martínez Manuel, Garcia Angélica. Seminario sobre vida artificial (antología). Universidad Veracruzana, Serie Textos Universitarios, Xalapa, México, marzo, 1996.

[Mar00a] Martínez M., Barradas P., Caballero H., Cancela N. E., Islas E., Lapizco G., Mendoza B., Sánchez R., Serna E., Varela V., Zavala R. L. Aprendizaje: reporte final. Maestría en Inteligencia Artificial, Universidad Veracruzana, 2000.

[Mar00b] Martínez Morales Manuel, Zavala Gutiérrez Rosa Laura, Islas Pérez Eduardo, Caballero Barbosa Hilda, Lapizco Encinas Grecia, Varela Lara Edna Verónica, Cancela García Nora Esmeralda, Mendoza García Benito, Serna Pérez Eduardo, Barradas Domínguez Pedro Darío, Sánchez Chacón Rodrigo. XLearn: un paquete de apoyo didáctico para un curso de ApR. En *MICAI 2000*. MICAI, Acapulco, Gro., abril 2000.

[Mat94] Mataric, Maja J. *Interaction and intelligent behavior*. Tesis doctoral, MIT LAB, 1994.

[McF95] McFarland, David. Autonomy and self-sufficiency in robots. En *The Artificial Life Route to Artificial Intelligence*, ed. Steels L. y Brooks R., pags. 187–113. Lawrence Erlbaum Associates, E.U.A., 1995.

[Mic86] Michalski, R. S., Carbonell, J. G., Mitchell, T. M., eds. *Machine Learning: an Artificial Intelligence approach*, Vol. II. Morgan Kaufmann, Los Altos, California, E.U.A, 1986.

[Min86] Minsky, Marvin. *The Society of the Mind*. Simon y Schuster, 1986.

[Mit97] Mitchell, Tom. *Machine Learning*. Mc Graw Hill, E.U.A., 1997.

[Mon98] Montes G., Fernando. Un modelo de conductas animales basado en el Mecanismo de Selección de Acción de Maes. Tesis de maestría, Universidad Veracruzana, 1998.

[Nor94] Norman, T. J. Motivated goal and action selection. AISB workshop: Models or Behaviors, which way forward for robotics?, abril, 1994.

[Pir99] Pirjanian, Paolo. Behavior coordination mechanisms – State-of-the-art. USC Robotics Research Laboratory. University of Southern California, octubre, 1999.

[Rod94] Rodriguez Luna, Ernesto et.al. *Forrajeo del mono aullador (Alouatta palliata) en semilibertad*. Parque de la Flora y Fauna Silvestre Tropical, Instituto de Neuroetología, Universidad Veracruzana, 1994.

[RyN95] Stuart Russell y Norving, Peter. *Artificial Intelligence: a modern approach.* Prentice Hall Inc., E.U.A, 1995.

[RyW91] Stuart Russell y Wefald, Erick H. *Do the right thing: studies in limited rationality.* MIT Press, Cambridge, MA, 1991.

[Sim83] Simon, Herbert A. ¿Why should machines learn? En *Machine Learning: an Artificial Intelligence Approach*, Vol. I. Michalski, R. S., Carbonell, J. G., Mitchell, T. M., eds., Los Altos, California, E.U.A, 1983.

[Sin02] Singleton, Darran. An evolvable approach to the Maes Action Selection Mechanism. Tesis de maestría, University of Sussex, 2002.

[SyB98] Richard Sutton y Barto, Andrew. *Reinforcement Learning: an introduction.* MIT Press, Cambridge, MA, 1998.

[Tyr93] Tyrrell, Toby. *Computacional mechanism for Action Selection.* Tesis doctoral, University of Edimburgh, 1993.

[Tyr94] Tyrrell, Toby. An evaluation of Maes's bottom-up mechanism for Behavior Networks. *Adaptive Behavior*, Vol. 2 No. 4 pags. 307–348, 1994.

[Uns97] Unsal, Cem. *Intelligent navigation of autonomous vehicles in an automated highway system: learning methods and interacting vehicles approach.* Tesis doctoral, Virginia Polytechnic Institute and State University, 1997.

[Wil85] Wilson, S.W. Knowledge growth in an artificial animal. En *Proceedings of the First International Conference on Genetic Algorithms and their Applications.* Lawrence Erlbaum Associates, 1985.

[Woo95] Wooldridge, Michael J. y Jennings, Nicholas R. Intelligent Agent: theory and practice. *The Knowledge Engineering Review*, Vol. 10 No. 2 pags. 115–152, 1995.