Tesis Psicológica | Vol. 16(2) julio-diciembre /21 | pp. 1-20| E- ISSN: 2422-045

PUBLICACIÓN ANTICIPADA EN LÍNEA *(Versión previa a la corrección de estilo y diagramación). La Revista Tesis Psicológica informa que este artículo fue evaluado por pares externos y aprobado para su publicación en las fechas que se indican en la siguiente página. Este documento puede ser descargado, citado y distribuido, no obstante, recuerde que en la versión final pueden producirse algunos cambios en el formato o forma.*

tesis Psicológica
Revista de la Facultad de Psicología

Tesis Psicológica | Vol. 16(2) julio-diciembre /21 | pp. 1-20| E- ISSN: 2422-045

Aproximación desde la Inteligencia Artificial a los comportamientos poco predictivos derivados de modelos cognitivos artificiales[1]

Approach from artificial intelligence to poorly predictive behaviors derived from artificial cognitive models

Jairo Iván Vélez-Bedoya[2]

Luis Fernando Castillo-Ossa[3]

Manuel González-Bedia[4]

Recibido: agosto 13 de 2020 *Revisado: septiembre 4 de 2020* *Aprobado: diciembre 17 de 2020*

Cómo citar este artículo: Vélez-Bedoya, J.I., Castillo-Ossa, L.F., & González-Bedia, M. (2021). Aproximación desde la Inteligencia Artificial a los comportamientos poco predictivos derivados de modelos cognitivos artificiales. *Tesis Psicológica*, *16*(2) 1-20.

Resumen

Antecedentes: Son varias las técnicas que permiten desarrollar modelos de comportamiento artificial poco predictivo, como por ejemplo, las máquinas de estados finitos (FSM) y el uso de arquitecturas cognitivas basadas en la teoría de la mente, para la construcción de agentes cuyo modelo conductual reside sobre un sistema de producciones. Objetivo: Se propuso generar modelos conductuales artificiales para determinar las condiciones en que estos demuestran comportamientos poco predictivos. Metodología: La primera etapa consistió en

[1] Artículo derivado del proyecto: Interpretación y evaluación de modelos cognitivos artificiales para determinar las condiciones en que estos mejor demuestran comportamientos poco predictivos. Maestría en Gestión y Desarrollo de Proyectos de Software de la Universidad Autónoma de Manizales.

[2] MSc. Doctorando en Ingeniería de Sistemas e Informática de la Universidad de Zaragoza (España). Profesor auxiliar Universidad de Caldas. Miembro del grupo de investigación GITIR. Orcid: 0000-0001-8756-1561 Correspondencia: jairo.velez@ucaldas.edu.co

[3] Ph.D. Líder del Grupo Inteligencia Artificial, Facultad de ingenierías, Universidad de Caldas. Profesor del Departamento de Ingeniería Industrial Universidad Nacional de Colombia Sede Manizales. Docente del Doctorado en Ciencias Cognitivas, Grupo de investigación Ingeniería de Software UAM. Orcid: 0000-0002-2878-8229 Correspondencia: luis.castillo@ucaldas.edu.co fcastilloos@unal.edu.co

[4] Ph.D. Líder del Grupo de investigación ISAAC, Profesor titular Universidad de Zaragoza Instituto de Investigación en Ingeniería de Aragón. Orcid: 0000-0002-8263-2444 Correspondencia: mgbedia@unizar.es

la elección de plataformas y herramientas.Se escogieron Pogamut, UT2000, SOAR y Java; en la segunda etapa se desarrolló la interfaz de acoplamiento entre el motor de cognición, el lenguaje y el entorno, por último, en la tercera etapa, se efectuaron pruebas con los modelos de comportamiento. Resultados: En el modelo FSM fue posible contrastar los estados y las decisiones que toman los agentes cuando se presentan restricciones en el conjunto de acciones predefinidas en su lógica. Así mismo fue posible el contraste entre producciones SOAR en cuanto a la predictibilidad de las acciones del agente a razón de lo percibido en el entorno. Conclusiones: Las máquinas de estados finitos son un componente importante cuando se quiere inspeccionar el comportamiento reactivo de un agente que persigue un único objetivo. Los agentes reflejos dependen de su lógica para la percepción inmediata de su entorno sin tener en cuenta las decisiones que han tomado o estados por los que ya hayan pasado. Los programas SOAR ajustan la retroalimentación de sus ambientes en ciertos casos.

Palabras clave: Inteligencia artificial, cognición, algoritmo, análisis de datos, programación informática.

Abstract

Background: Several techniques allow the development of poorly predictive artificial behavior models, such as finite state machines (FSM) and the use of cognitive architectures based on the theory of mind for the construction of agents whose behavioral model lies on a system of productions. Objective: It was proposed to generate artificial behavioral models to determine the conditions under which they demonstrate poorly predictive behaviors. Methodology: The first stage consisted of choosing platforms and tools; Pogamut, UT2000, SOAR, and Java were chosen. In the second stage, the coupling interface between the cognition engine, the language, and the environment was developed. Finally, in the third stage, the behavioral models were tested. Results: In the FSM model, it was possible to contrast the states and decisions made by the agents when there are restrictions in the set of actions predefined in its logic. It was also possible to contrast SOAR productions in terms of the predictability of the agent's actions based on what is perceived in the environment. Conclusions: Finite state machines are an important component when one wants to inspect the reactive behavior of an agent that pursues a single goal. Reflexive agents rely on their logic for the immediate perception of their environment without regard to decisions they have made or states they have already undergone. SOAR programs adjust the feedback of their environments in certain cases.

Keywords: artificial intelligence, cognition, algorithm, data analysis, computer programming

INTRODUCCIÓN

De acuerdo con (Chakraborti *et al.,* 2019), últimamente surge un interés significativo en la comunidad de la robótica y la planificación en el desarrollo de algoritmos que puedan generar el comportamiento de los agentes que sea interpretable para el humano (observador) en el ciclo de vida de un agente inteligente. Esta noción de interpretabilidad puede ser en términos de metas, planes o incluso recompensas que el observador puede atribuir al agente con base en las observaciones de este último. La interpretabilidad sigue siendo un desafío importante en el diseño de agentes de IA con conciencia humana. Aunque (Petersen & Sporns, 2015) sostienen que la mayoría de las descripciones de las arquitecturas cognitivas humanas se han centrado en las explicaciones computacionales de la cognición, sin tener mucho contacto con el estudio de las estructuras anatómicas y los procesos fisiológicos, un área prometedora es la superposición entre sistemas y neurociencia cognitiva por un lado y la disciplina de la ciencia de redes por el otro. Herramientas como Pogamut que define una interfaz (usando la arquitectura GameBots) para programar los bots como módulos externos de Java (Mora, Castillo, García-Sanchez & Merelo, 2015), de manera que facilitan la gestión de prototipos de agentes haciendo posible dotarles de un modelo de comportamiento, que a su vez, es posible validarlo, bien sea, a través de inspecciones de código o visualizaciones en 3D y tiempo real que ofrece la integración con Unreal® que brinda tal entorno virtual.

En cuanto al modelo de comportamiento, son varias las técnicas que lo permiten desarrollar; así pues, en este artículo se presentan las máquinas de estados finitos y el uso de la arquitectura cognitiva SOAR, basada en la teoría de la mente de Newell para la construcción de agentes inteligentes, en la que el modelo conductual reside sobre un sistema de producciones (Laird, Newell & Rosenbloom, 1987).

En la sección 2, se describen varias etapas relacionadas con la metodología, iniciando con la elección de plataformas y herramientas de desarrollo para la producción de los modelos conductuales, estableciendo a Pogamut, Java, SOAR y Unreal® como las opcionadas para tal fin; posteriormente, en la segunda etapa, se construyó un módulo que controla la interacción del agente con el entorno, de manera que esta se dé mediante la transmisión de los estados percibidos en el entorno virtual, a la arquitectura cognitiva, y esta, a modo de producciones determine las acciones que serán transmitidas de vuelta al agente para que las lleve a cabo. Luego, la tercera etapa tiene que ver con la experimentación; en ella todos los modelos, tanto los de los agentes que basan su conducta en máquinas de estados finitos como aquellos en los que su actuar obedece a lo determinado por la arquitectura cognitiva, se someten a prueba en un entorno controlado en términos del terreno, ítems y características homogéneas del entorno. En la sección 3, se analizan los resultados obtenidos en la experimentación con los modelos, y, finalmente, en la sección 4 se contemplan conclusiones tanto para los modelos de comportamiento basados en máquinas de estados finitos como para los modelos asistidos por la arquitectura cognitiva.

METODOLOGÍA

Tipo y diseño

Investigación aplicada de tipo experimental y exploratoria, en ella se consideran distintas variables con el propósito de entender la predictibilidad del comportamiento en un agente artificial por medio de algoritmos y técnicas computacionales. Como se muestra en Vélez, Castillo y González (2010), el diseño contempló varias etapas que tienen que ver con la selección de la infraestructura, el acoplamiento de los distintos componentes que tienen arquitecturas disímiles, desarrollo de los modelos comportamentales y la respectiva experimentación con estos. No obstante, en esta investigación, para el desarrollo de los modelos y la experimentación, se seleccionó la arquitectura cognitiva SOAR, que muestra compatibilidad completa con el lenguaje Java y el entorno de pruebas UT2004; siendo necesario acoplar estos componentes. Posteriormente, se recrearon 4 experimentos que

tienen que ver con la movilidad por el entorno, la imitación de comportamientos, y la recolección de ítems, además de una aproximación determinista por medio de una máquina de estados finitos (FSM). Finalmente, se desarrolló un agente que basa su comportamiento apoyado en la arquitectura cognitiva SOAR y se analizaron los datos obtenidos en aras del desempeño del agente.

Instrumentos

Elección de plataformas y herramientas

La figura 1 muestra las alternativas que se compararon y probaron para seleccionar las que se consideraron más apropiadas en cuanto a la integración y compatibilidad del entorno de desarrollo con la plataforma y el motor de la arquitectura cognitiva. Finalmente, se seleccionó SOAR como arquitectura cognitiva, JAVA como lenguaje de programación, UT2004 como entorno para los agentes y POGAMUT para el desarrollo de prototipos de agentes artificiales.

Figura 1. Arquitecturas, entornos, lenguajes y herramientas para el prototipado

Arquitecturas cognitivas • SOAR • ACT-R • 4CAPS • LIDA	Lenguajes de programación • Java • C++ • Python
Entornos para las pruebas • UT2004 • Quake • Descent III	Herramientas para el desarrollo de prototipos • Pogamut

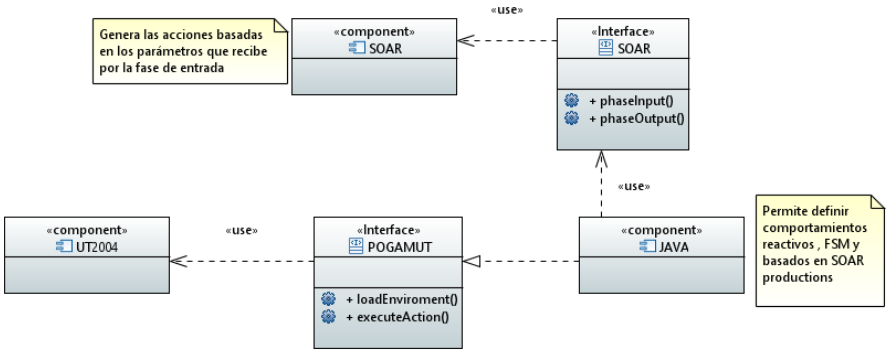
Fuente: autores

La razón de tal elección obedece al interés de probar los modelos en un entorno escalable y adaptable, características que hacen que Unreal®, que es más reciente, sobresalga por encima de Quake y Descent III. En SOAR, lo que la hizo elegible fue el dinamismo para determinar la selección y aplicación de operadores, pues como se enuncia en (Langley, Laird & Rogers, 2008) todas las tareas se formulan como intentos de lograr objetivos. Los operadores realizan los actos deliberativos básicos del sistema, con el conocimiento utilizado para determinar dinámicamente su selección y aplicación; así, el sistema de rendimiento compara las producciones con elementos en la memoria de trabajo y genera submetas automáticamente cuando no puede continuar. En este paso fue relativamente simple elegir a Pogamut ya que es un framework desarrollado en Java que permite el control de los agentes Unreal® (Mora, Castillo, García-Sanchez & Merelo, 2015), y así, aunque el lenguaje C++ eventualmente ofrecería mejor rendimiento, se optó por Java para conservar el mismo lenguaje de programación del framework.

Desarrollo de la interfaz de acoplamiento: motor de cognición, lenguaje de programación y entorno de pruebas

En esta fase se construyó un módulo que relaciona el lenguaje Java, el entorno de pruebas UT2004 y la arquitectura cognitiva SOAR, esto facilitó la programación de los agentes, de manera que pudieran exhibir tanto sus comportamientos reactivos como los que fueron determinados por la arquitectura cognitiva. La figura 2 ilustra el diseño arquitectónico y configuración del sistema.

Figura 2. Componentes arquitectónicos del sistema



Fuente: autores

Desarrollo y experimentación con los modelos de comportamiento

Se codificó un modelo de comportamiento reactivo, que se rige por la arquitectura que se muestra en la figura 3, para un conjunto de agentes que interactúan entre sí de manera individual persiguiendo un objetivo previamente establecido.

Figura 3. Estructura de un agente reactivo

Fuente: (Norvig & Russell, 2010)

El proceso de recolección de datos se hizo a través de la generación de archivos de tipo log en los cuales se registraron las decisiones y los eventos que incidieron en el comportamiento del agente.

Como se muestra en Vélez, Castillo y González (2010), se desarrollaron 4 modelos previos al que le corresponde a la arquitectura cognitiva. Sin embargo, los datos que se obtuvieron con esos modelos se presentan aquí de una manera más granular como lo muestran las figuras 1 y 4.

Procedimiento

Experimento No.1. Navegar por el entorno: En cuanto al desplazamiento del agente, este se realiza buscando puntos de navegación adyacentes a su posición, aunque puede ocurrir que se bloquee, de ser así, por medio de giros aleatorios el agente identificará puntos de navegación que le permitan continuar con el desplazamiento.

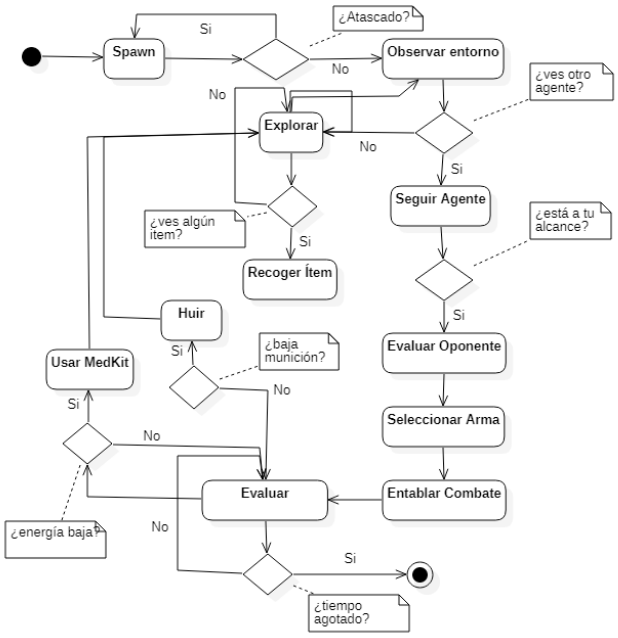
Experimento No.2. Imitar otros agentes: Una vez que el agente pudo desplazarse, se le dotó con el comportamiento de seguir e imitar los movimientos de otro agente que encuentre en su campo de visión y que sea el más próximo, para tal efecto, se inyectó en el entorno otro agente cuyo propósito fue dejarse detectar y navegar por el entorno.

Experimento No. 3. Recolectar: El comportamiento de recolección toma como base el desplazamiento y parte de la lógica implantada para el seguimiento, pues, en lugar de imitar o seguir otro agente, se le da prioridad a recolectar ítems en la medida que los detectara en el entorno, así, en la figura 3, se observa que los datos registrados ante la acción de recolección son muy aproximados a los de la navegación.

Experimento No.4 Agente basado en máquinas de estado finito (FSM): Para determinar el comportamiento reactivo, del agente, basta con inyectar sus estados directamente dentro de su lógica, ya que según Cerny *et al.* (2016), los comportamientos de un agente son diversos porque son inyectados y el uso del comportamiento da como resultado la inserción de un nuevo subárbol en el árbol de decisiones del agente. En particular, un nodo más cercano a la raíz del árbol puede cambiar a un sucesor diferente y terminar el comportamiento inyectado, por ejemplo, cuando se utilizan máquinas de estado finito (FSM). Así, los árboles de decisión proporcionan naturalmente un soporte muy limpio para la descomposición y la estructuración jerárquica de comportamientos.

Según Champandard (2003) las técnicas reactivas son deterministas, y esto presume una ventaja al conocer la salida exacta ante cualquier entrada, haciendo que tanto el código y las estructuras de datos puedan ser optimizados por partes, así como la facilidad en el proceso de depuración, pues, ante un fallo, la razón exacta se identifica fácilmente. Así, en la figura 5 se muestra el diagrama de estados que corresponde al curso normal de los eventos considerado en Vélez, Castillo y González (2010) para la FSM que define el comportamiento del agente reactivo.

F*igura 5. Diagrama de estados de máquina para FSM*



Fuente: autores

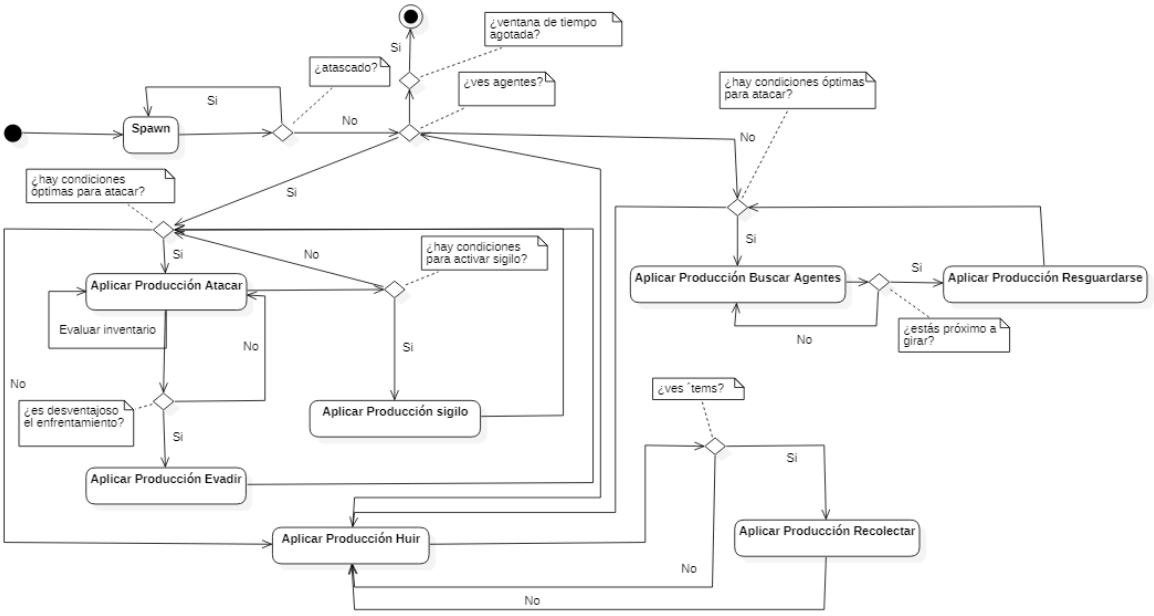
Experimento No.5 Agente que basa sus decisiones en un motor de cognición SOAR: El objetivo que busca el desarrollo del modelo de comportamiento programado en SOAR para un agente, se basa más que nada en la obtención de los datos que soportan las decisiones que toma el agente según lo determine la arquitectura. Para alimentar la base de datos de conocimiento se registran los siguientes campos de información:

Agente: Corresponde al nombre del agente que se le asigna automáticamente cuando ingresa en el entorno.

Evento enviado a SOAR: Los eventos corresponden a los que el agente percibe en el entorno, entre los que se encuentran ver objeto, estar siendo herido, recibir mensaje, entre otros. Este evento se registra en el input-link de SOAR.

Acción recomendada: Las acciones son las aplicaciones de las producciones que SOAR determina como procedentes o viables en el momento en que son percibidos los eventos en el entorno, entre las que se encuentran, evadir, vagar, disparar y emboscar. En la figura 6 se muestran las producciones de SOAR y los eventos que las disparan en la memoria de trabajo del agente.

Figura 6. Diagrama de Estados para Producciones de SOAR



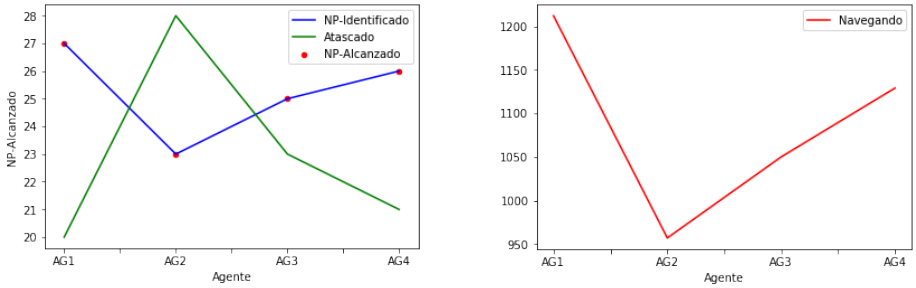
Fuente: autores

En las pruebas individuales de cada producción, el agente fue configurado de manera que tuviera activa la única producción que determinase su comportamiento, pero cuando una producción dependía de la aplicación de otras producciones, se activara el conjunto de aplicaciones necesarias para realizar dicha producción. Los resultados de las pruebas individuales satisficieron las expectativas que se tenían sobre el logro de los comportamientos individuales, para lo cual, se consideró como suficiente la inspección visual para la comprobación del funcionamiento de cada producción. Posteriormente, se configuró la prueba de manera que los cuatro agentes se enfrentaran entre sí para determinar la manera en que cada uno iba asumiendo comportamientos.

RESULTADOS

En cuanto a la implementación de modelos de comportamiento reactivo basados en las tres operaciones reflejo básicas de los agentes, en lo que tiene que ver con el primer experimento, la figura 1 revela que, mayormente, los agentes se comportaron como estaba determinado en su comportamiento (navegar por el entorno); sin embargo, también se evidencia que el agente demuestra una proporción inversa en la cantidad de registros de puntos de navegación alcanzados en contraposición a cuando éste presenta atascamientos, aunque la proporción de puntos alcanzados y atascamientos ocurre entre un mismo rango de veces.

Figura 1. Prueba de desempeño de los agentes navegadores



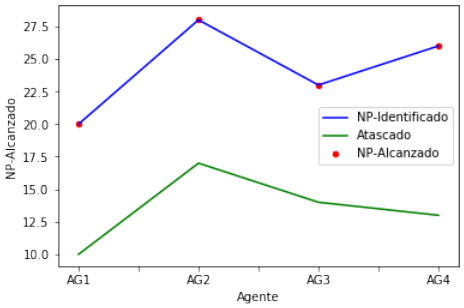
	Agente	Navegando	NP-Identificado	Atascado	NP-Alcanzado
0	AG1	1212	27	20	27
1	AG2	957	23	28	23
2	AG3	1050	25	23	25
3	AG4	1129	26	21	26

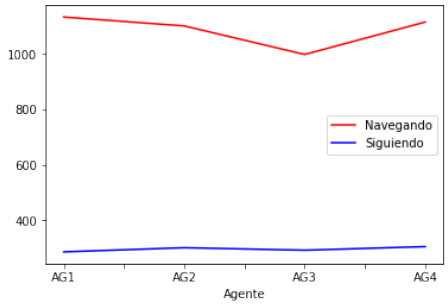
CORREDOR

Fuente: autores

Para el segundo experimento, como se ilustra en la figura 2, el atascamiento fue bajo, en contraste con el registro de las demás acciones. Este hecho aporta a la conducta del agente, la disminución de tomar caminos erráticos valiéndose de la imitación.

Figura 2. Prueba de desempeño de los agentes con capacidad de seguir otros agentes





	Agente	Navegando	Atascado	NP-Identificado	NP-Alcanzado	Siguiendo
0	AG1	1133	10	20	20	286
1	AG2	1101	17	28	28	301
2	AG3	998	14	23	23	292
3	AG4	1115	13	26	26	305

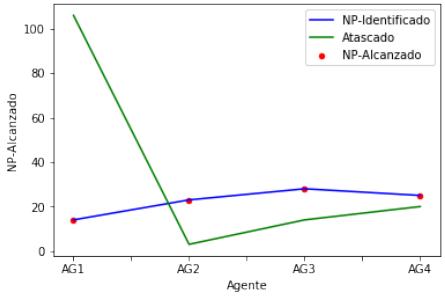
RASTREADOR

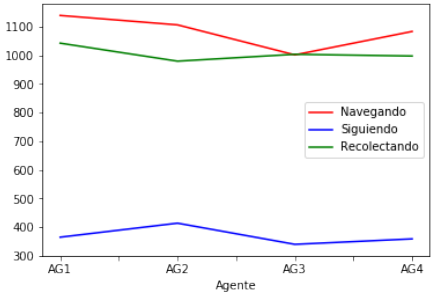
Fuente: autores

La identificación y el alcance de los puntos de navegación estuvieron directamente relacionados en proporción, aunque la mayor parte del tiempo el agente navegaba por el entorno, buscando a quien seguir, la acción de seguimiento fue significativa demostrando cierta cercanía a los valores relacionados con la acción de navegación, lo que demuestra que el agente se comportó como se esperaba según lo establecido en su modelo de conducta; en este experimento fue claro que el comportamiento errático puede ser disminuido notablemente si se imita un patrón de movimiento-desplazamiento. Es importante la evidencia sobre el indicio de que la optimización de las acciones que realiza un agente puede estar sujeta al aprendizaje por medio de la imitación del comportamiento de otros agentes.

En cuanto al tercer experimento, como lo muestra la figura 3, el primer agente tuvo un atascamiento mayor al de los demás agentes, incluso a los de los experimentos anteriores; en lo que tiene que ver con los datos de seguimiento, navegación identificación y alcance de puntos de navegación la proporción se mantiene constante. El proceso de recolección se hace por reflejo y mayormente como una acción secundaria, por ejemplo, cuando se seguía a otro agente, la recolección de ítems se daba más que nada por colisionar con ellos por azar.

Figura 3. Prueba de desempeño de los agentes con capacidad de recolección





	Agente	Navegando	Atascado	NP-Identificado	NP-Alcanzado	Siguiendo	Recolectando
0	AG1	1140	106	14	14	364	1043
1	AG2	1107	3	23	23	413	980
2	AG3	1002	14	28	28	339	1004
3	AG4	1084	20	25	25	358	998

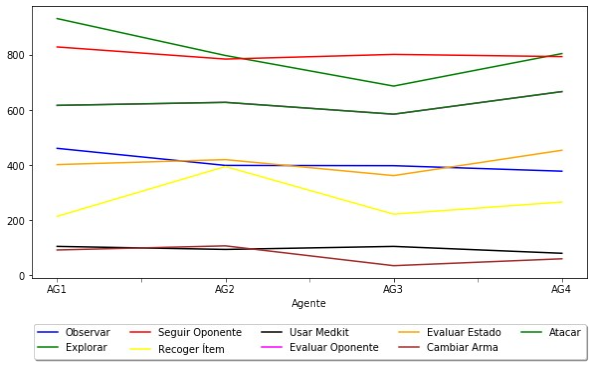
RECOLECTOR

Fuente: autores

En la figura 4, se muestra un comportamiento casi homogéneo de todos los agentes, excepto por algunos casos en los que las condiciones variables del

entorno, como el lugar aleatorio donde aparecen ítems para recolectar y la exploración del entorno debido a que los agentes no siempre transitan por los mismos lugares en el terreno, teniendo en cuenta que cuando un agente es eliminado, pero la ventana de tiempo de la prueba todavía está activo, éste reingresa al entorno, lo hace sin perder su puntuación y no siempre en el mismo lugar, así las cosas, se concluye que cuando los agentes no siguen a otros, presentan un mejor desempeño a la hora de atacar, es probable que el agente tenga más éxito si no persigue a todo lo que se vea. Cuando los agentes no perciben bajos niveles de energía, demuestran mejor desempeño que aquellos que si las perciben pero que no sigue a otros agentes, es factible que al no percibir heridas, el tiempo que se tomarían para decidir curarse, lo utilizan para perseguir y atacar.

Figura 5. Prueba del modelo conductual basado en FSM



	Agente	Observar	Explorar	Seguir Oponente	Recoger Ítem	Usar Medkit	Evaluar Oponente	Evaluar Estado	Cambiar Arma	Atacar
0	AG1	461	932	829	214	105	617	402	92	617
1	AG2	399	798	785	395	94	628	420	107	628
2	AG3	398	687	802	222	105	585	362	35	585
3	AG4	378	805	794	266	80	667	454	60	667

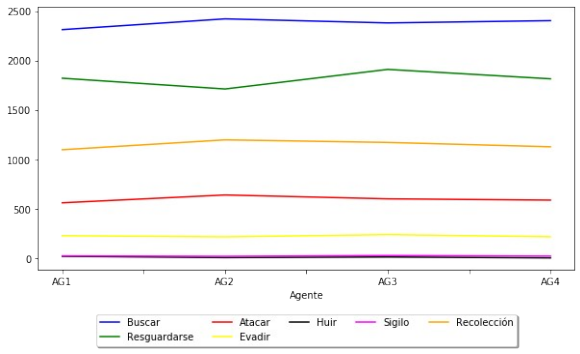
Fuente: autores

En este experimento, el rendimiento, de los agentes que no identifican ítems para recolectar, es bajo, lo que puede deberse a que, ante la ausencia de este estado, el agente se

queda rápidamente sin recursos para atacar, por ejemplo, puede quedarse sin munición o sin armas y de esta manera no sería poco probable que lograra su objetivo.

Como se ilustra en la figura 5, el desempeño de los agentes fue semejante, la navegación fue la acción más ejecutada, aunque se evidencia que el comportamiento adoptó cierta cautela ya que la cantidad de acciones de resguardo va ligada casi en la misma proporción que la navegación; luego de la búsqueda y el resguardo, la activación de las producciones para atacar, evadir y huir son las más predominantes, así, este modelo conductual es distinto al de las FSM, pues éste último, aunque también es determinista, reacciona a lo primero que se presenta.

Figura 5. Prueba del modelo conductual apoyado por SOAR



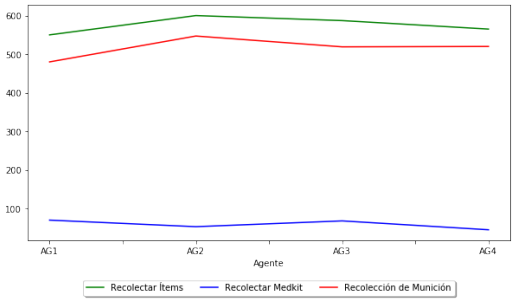
	Agente	Buscar	Resguardarse	Atacar	Evadir	Huir	Sigilo	Recolección
0	AG1	2314	1823	564	231	23	27	1100
1	AG2	2423	1714	643	219	10	23	1200
2	AG3	2382	1912	604	242	16	31	1174
3	AG4	2405	1817	591	220	7	25	1130

Fuente: autores

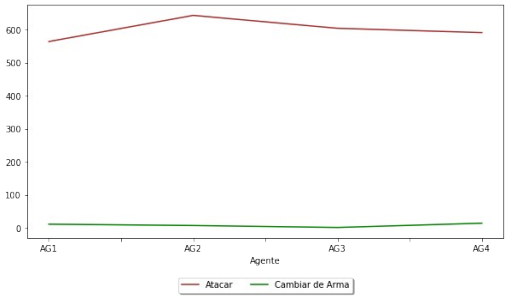
De la recolección se pueden distinguir tres sub-producciones, recoger medkits, munición e ítems; por lo cual, en la figura 6 (izquierda) se muestra que la recolección se dio dependiendo del estado del agente y que los agentes recogían más municiones e ítems que

medkits, esto fue un indicio que las activaciones de las producciones en SOAR no fueron en contrasentido a los parámetros del modelo de conducta que procuran que el agente domine el entorno, algo similar muestra la figura 6 (derecha) en la que el comparativo se hace entre el ataque y las veces en las que los agentes cambiaron de arma, pues al recoger más ítems, el agente asegura no recibir daño mientras la ventana de tiempo de tales ítems esté activa, siendo así necesario el cambio de arma cuando el estado del agente determinaba que el oponente tenía ventaja sobre él.

Figura 6. Activación de producciones de recolección y ataque



	Agente	Recolección de Munición	Recolectar Ítems	Recolectar Medkit
0	AG1	480	550	70
1	AG2	547	600	53
2	AG3	519	587	68
3	AG4	520	565	45



Fuente: autores

DISCUSIÓN Y CONCLUSIONES

Los comportamientos se determinan a través de árboles binarios de manera que con las informaciones que reciben del entorno se fueran desplazando hacia las hojas, en las que se encuentran las “acciones simples” que determinan la acción que va a ejecutar el agente.

El desarrollo de los comportamientos tiene el problema que deben ser determinados antes de empezar y la cantidad de variables a reducir para tener un conjunto de datos manejable.

El comportamiento errático puede ser disminuido notablemente si se copia un patrón adoptado por otro agente, aunque algunas acciones se efectúan por reflejo simple, hay procesos en los que las acciones que desencadenan se realizan por casualidad, es decir, que para lograr un objetivo pueden ocurrir eventos secundarios que dan cierto valor agregado al comportamiento.

Los scripts en SOAR ajustan automáticamente la retroalimentación de sus ambientes en casos donde las preferencias simbólicas son poco apropiadas para tomar decisiones.

Es necesario ajustar los parámetros de los modelos de comportamiento, de manera que se hagan dinámicos según sea el nivel del oponente.

Una posibilidad para realizar trabajo futuro con este tipo de modelos es incorporar redes neuronales convolucionales para (CNN), para que en lugar de procesar y evaluar estados sobre estructuras de datos recursivas (árboles), o matriciales (grafos), se analice cada fotograma, de manera que éste sea transformado por el proceso de convolución hasta que finalmente se llegue una acción a partir del estado actual del agente. Al final sería interesante ver una comparación entre los comportamientos determinados por las máquinas de estado finito FSM, la arquitectura cognitiva SOAR y el Deep Learning, aunque puede anticiparse que el proceso de entrenamiento de la red convolucional supondría una capacidad de cómputo importante y el tiempo que tomaría el entrenamiento de la misma.

Conflictos de interés: Los autores declararnos no tener conflictos de interés relacionados con el artículo.

REFERENCIAS

Cerny, M., Plch, T., Marko, M., Gemrot, J., Ondracek, P. & Brom, C. (2016). *Using Behavior Objects to Manage Complexity in Virtual Worlds.* University of Charles.

Chakraborti, T., Kulkarni, A., Sreedharan, S., Smith,, D. E., & Kambhampati, S. (2019). Explicability? Legibility? Predictability? Transparency? Privacy? Security? The Emerging Landscape of Interpretable Agent Behavior. In J. Benton (Ed) *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling (Vol. 29) (ICAPS 2019)* (pp. 86 - 96). Association for the Advancement of Artificial Intelligence.

Champandard, A. (2003). *AI Game Development: Synthetic Creatures with Learning and Reactive Behaviors.* New Riders Publishing.

Laird, J., Newell, A. & Rosenbloom, P. (1987). SOAR: An Architecture for General Intelligence. *The Journal of Artificial Intelligence [AIJ], 33*(1), 1-64. https://doi.org/10.1016/0004-3702(87)90050-6

Langley, P., Laird, J. & Rogers, S. (2008). Cognitive architectures: Research issues and challenges. (Elsevier, Ed.) *Cognitive Systems Research, 10*(2), 143-158. https://doi.org/10.1016/j.cogsys.2006.07.004

Mora, A., Castillo, P. A., García-Sanchez, P. & Merelo, J. J. (2015). Modelling a Human-Like Bot in a First Person Shooter Game. *International Journal of Creative Interfaces and Computer Graphics, 6*(1), 21-37. https://doi.org/10.4018/IJCICG.2015010102

Norvig, P. & Russell, S. (2010). *A.I. A modern Approach. 3rd Ed.* New Jersey: Pearson.

Petersen, S. E., & Sporns, O. (2015). Brain Networks and Cognitive Architectures. *Neuron Perspective, 88*(1), 207 – 219. https://dx.doi.org/10.1016/j.neuron.2015.09.027

Vélez B., J. I., Castillo O., L. F. & González, M. (2010). Implementación de un modelo de comportamiento reactivo para agentes en un entorno de videojuegos. *Revista Vector, 5*(1), 61 - 68. http://vector.ucaldas.edu.co/downloads/Vector5_7.pdf