INTELIGENCIA ARTIFICIAL

Teorías, aplicaciones, futuro

Juan David Gutiérrez
y Rubén Manrique
(edición académica)

INTELIGENCIA ARTIFICIAL

Juan David Gutiérrez
y Rubén Manrique
(edición académica)

INTELIGENCIA ARTIFICIAL

Teorías, aplicaciones, futuro

Universidad de los Andes
Escuela de Gobierno Alberto Lleras Camargo
Ediciones Uniandes

Nombre: Gutiérrez, Juan David. edición académica. | Manrique, Rubén, edición académica.

Título: Inteligencia artificial: teorías, aplicaciones, futuro / Juan David Gutiérrez y Rubén Manrique (edición académica)

Descripción: Bogotá: Universidad de los Andes, Escuela de Gobierno Alberto Lleras Camargo, Ediciones Uniandes, 2025. | 329 páginas: ilustraciones; 17 × 24 cm.

Identificadores: ISBN 978-958-798-844-4 (rústica) | 978-958-798-845-1 (e-book) | 978-958-798-846-8 (e-pub)

Materias: Inteligencia artificial

Clasificación: CDD 006.3–dc23 SBUA

Primera edición: octubre del 2025

Ediciones Uniandes
Carrera 1.ª n.º 18A-12, bloque Tm
Bogotá, D. C., Colombia
Teléfono: 601 339 4949, ext. 2133
http://ediciones.uniandes.edu.co
ediciones@uniandes.edu.co

ISBN: 978-958-798-844-4
ISBN *e-book*: 978-958-798-845-1
ISBN epub: 978-958-798-846-8
DOI: https://doi.org/10.51573/Andes.9789587988444.9789587988451.9789587988468

Corrección de estilo: Camilo Sierra Sepúlveda
Diagramación: María Victoria Mora
Diseño de cubierta: Boga Visual
Imagen de cubierta: adaptada por Boga Visual, con base en Yutong Liu & Kingston School of Art / https://betterimagesofai.org / https://creativecommons.org/licenses/by/4.0/

Impresión:
DGP Editores S. A. S.
Calle 63 n.º 70D-34
Teléfonos: 601 721 7641 - 601 721 7756
Bogotá, D. C., Colombia

Impreso en Colombia – *Printed in Colombia*

Este libro cuenta con el aval de la Escuela de Gobierno Alberto Lleras Camargo y fue sometido a evaluación de pares académicos.

Universidad de los Andes | Vigilada Mineducación. Reconocimiento como universidad: Decreto 1297 del 30 de mayo de 1964. Reconocimiento de personería jurídica: Resolución 28 del 23 de febrero de 1949, Minjusticia. Acreditación institucional de alta calidad, 10 años: Resolución 000194 del 16 de enero del 2025, Mineducación.

CONTENIDO

RECURSOS GRÁFICOS

USO DEL APRENDIZAJE POR REFUERZO PARA EL MANEJO DE COMPORTAMIENTO DESCONOCIDO EN SISTEMAS DE *SOFTWARE* DINÁMICOS

POTENCIAL DE LA INTELIGENCIA ARTIFICIAL EN TELEDETECCIÓN PARA EL DESARROLLO SOSTENIBLE Y LA GESTIÓN AMBIENTAL

INNOVACIONES DE LA WEB SEMÁNTICA PARA LA EDUCACIÓN SUPERIOR

INTELIGENCIA ARTIFICIAL PARA LA PREVENCIÓN DEL ABUSO SEXUAL EN LÍNEA DE LA INFANCIA Y LA ADOLESCENCIA EN COLOMBIA

SISTEMAS DE INTELIGENCIA ARTIFICIAL EN EL SECTOR PÚBLICO DE AMÉRICA LATINA Y EL CARIBE

IMAGINARIOS SOCIOTÉCNICOS Y PRÁCTICAS ANTICIPATORIAS EN EL CUBRIMIENTO MEDIÁTICO DE LA INTELIGENCIA ARTIFICIAL Y SU RELACIÓN CON EL ESTADO EN COLOMBIA

INTRODUCCIÓN

Juan David Gutiérrez y Rubén Manrique

ESTE LIBRO REÚNE ESFUERZOS Y DOMINIOS TEMÁTICOS DE investigadores y profesores de la Universidad de los Andes que trabajan en el estudio y desarrollo de la inteligencia artificial (IA). De esta manera, busca aportar desde tres frentes: el avance y la crítica de la teoría sobre la IA; la creación y el análisis de herramientas y aplicaciones basadas en IA, y la reflexión sobre las implicaciones actuales y potenciales de estas tecnologías.

Esta obra colectiva está dividida en tres secciones y comprende diez capítulos, los cuales abordan el estudio de la IA a partir de disciplinas y áreas tan diversas como la ingeniería, la historia, la ciencia de datos, la filosofía, la arquitectura, el diseño, el desarrollo y gestión ambiental, la psicología, la administración, la política y la salud públicas.

La realización de este libro es una iniciativa de la Escuela de Gobierno Alberto Lleras Camargo, que financió su publicación y apoyó su gestión editorial, y de Ediciones Uniandes, que propuso la realización del libro y gestionó su edición. El proyecto apunta a fomentar la misión de la Escuela de Gobierno de articular esfuerzos de investigación interdisciplinarios para que la Universidad de los Andes contribuya al aprovechamiento de oportunidades colectivas y a la solución de problemas complejos de Colombia.

Para presentar este volumen comenzamos por exponer brevemente el objeto de estudio, la IA; luego, explicamos el papel de la Universidad de los Andes en los procesos de transformación digital del país, con énfasis en sus aportes a la teoría y la práctica de la IA; para terminar, en la última sección presentamos cada uno de los diez capítulos que componen este libro.

Qué es la inteligencia artificial

En 1950, el escritor Isaac Asimov publicó un libro de ciencia ficción que describía el desarrollo de "máquinas pensantes" por parte de una empresa denominada U. S. Robots & Mechanical Men Inc. Su obra, *Yo, robot*, es conocida por el planteamiento de las tres leyes de la robótica y por anticipar algunos retos técnicos, sociales, políticos y económicos asociados a la creación y adopción de lo que hoy conocemos como sistemas de IA.

Ese mismo año, Alan M. Turing publicó el ensayo titulado "Maquinaria computacional e inteligencia", en el cual propuso reemplazar el abordaje de la pregunta "¿pueden pensar las máquinas?" con un experimento mental que denominó *juego de la imitación* (luego conocido como *test de Turing*). En síntesis, el juego plantea una situación en la que un interrogador humano tiene la tarea de interactuar con dos interrogados, un humano y una máquina, y debe distinguir cuál es cuál —exclusivamente a partir de las respuestas de los interrogados (Turing, 1950)—.

De esta forma, Turing reemplazó la pregunta sobre si las máquinas pueden pensar por otro interrogante: "¿hay computadores digitales imaginables que tendrían un buen desempeño en el juego de la imitación?". Al final del ensayo, Turing reflexiona sobre cómo podría lograrse que una máquina imitara la mente de un humano: su respuesta inicial es partir por la simulación de procesos educativos, es decir, indagó sobre cómo desarrollar procesos de aprendizaje de máquinas, lo cual, como veremos más adelante, es una de las técnicas contemporáneas más exitosas en el desarrollo de la IA.

El término *inteligencia artificial* fue acuñado en 1955 por John McCarthy y Marvin L. Minsky, quienes plantearon que "el problema de la inteligencia artificial consiste en hacer que una máquina se comporte de un modo que se consideraría inteligente si lo hiciera un ser humano"; su propuesta de investigación partía de la conjetura de que "cada aspecto del aprendizaje o cualquier otra característica de la inteligencia puede, en principio, describirse con tanta precisión que se puede hacer que una máquina lo simule" (McCarthy *et al*., 1955, p. 2).

Siete décadas después de que la IA fuera fundada como disciplina de investigación, no hay consenso acerca de la definición de su objeto de estudio. Sin embargo, para efectos de este texto introductorio, podemos entender las herramientas de IA como sistemas computacionales que operan a partir de datos y que, con diferentes grados de autonomía, pueden resolver problemas o alcanzar objetivos establecidos por seres humanos (Gutiérrez, 2024).

Los sistemas de IA se construyen a partir de diferentes técnicas y métodos, como la IA simbólica, el aprendizaje automático (p. ej., supervisado, no supervisado, por refuerzo, profundo), el procesamiento de lenguaje natural, entre

otros. Estas metodologías varían en sus principios y enfoques, pero persiguen un objetivo común: emular capacidades cognitivas humanas para resolver problemas complejos y realizar tareas de manera autónoma.

En los primeros días de la informática, los programadores creaban algoritmos para resolver problemas específicos, que luego codificaban en programas ejecutados por computadoras. Estos programas, aunque no poseían inteligencia propia (eran un reflejo de la inteligencia del programador), permitían resolver tareas de manera eficiente. A pesar de años de investigación entre las décadas de los sesenta y los noventa, los algoritmos desarrollados no fueron lo suficientemente buenos para muchas aplicaciones, como el entendimiento del mundo visual o del lenguaje humano.

La solución de los desarrolladores fue recopilar datos para estas tareas y emplear programas de aprendizaje automático —el área más importante de la IA en la actualidad— para crear algoritmos estocásticos a partir de estos. En un programa que "aprende", se especifica cómo utilizar los datos para actualizar los parámetros de un modelo matemático y mejorar su rendimiento en una labor particular (Alpaydin, 2020). El aumento en la disponibilidad de datos de alta calidad, junto con el rápido crecimiento del poder computacional, ha permitido en la actualidad desarrollar modelos capaces de manejar diversas tareas complejas, logrando un desempeño que con frecuencia supera al del ser humano.

Gracias a la IA, potenciada en las últimas dos décadas por el aprendizaje automático, se pueden construir soluciones a problemas tan variados como identificar y categorizar datos (p. ej., reconocimiento facial); detectar patrones, anomalías, valores atípicos (p. ej., detección de riesgo de fraude financiero); predecir futuros comportamientos a partir de hechos pasados y presentes (p. ej., predicción de la conducta de la población en medios de transporte masivos); desarrollar un perfil de un individuo y adaptarse a través del tiempo (p. ej., recomendación asistida en motores de búsqueda de internet); generar contenido a partir de interacción con seres humanos (p. ej., *chatbots*); encontrar soluciones óptimas a un problema (p. ej., optimización de operaciones logísticas); e inferir resultados a partir de modelamiento y simulación (p. ej. reclutamiento de talento humano) (Organización para la Cooperación y el Desarrollo Económico [OECD], 2022).

En este contexto, también se destaca la emergente y disruptiva área de la IA generativa. Se trata de sistemas capaces de crear contenido original a partir de datos existentes. Estos sistemas no solo identifican patrones en grandes volúmenes de información, sino que utilizan estos patrones para generar nuevos datos que imitan y combinan los existentes, innovando de forma constante sobre los datos originales.

Un ejemplo emblemático de la IA generativa son los modelos de lenguaje de gran tamaño (LLM, por sus siglas en inglés) que pueden generar ensayos, programar o mantener conversaciones con un humano. Esta capacidad de producir contenido sintético útil abre nuevas posibilidades en campos como el entretenimiento, la educación, el diseño, entre otros, llevándonos un paso más allá en el desarrollo del potencial de la IA en nuestra vida cotidiana.

En la última década y a nivel global, ha aumentado el porcentaje de organizaciones privadas y públicas que han adoptado diferentes sistemas de IA para apoyar o realizar sus procesos productivos o sus flujos de trabajo. De manera similar, cada vez más individuos utilizan tecnologías basadas en IA en su día a día con fines laborales y personales. Estas tendencias se han acentuado desde finales del 2022, debido al crecimiento de la oferta de herramientas de IA generativa como ChatGPT, Copilot, Gemini, Claude, Meta AI, Grok, DeepSeek, entre otros, que tienen acceso a través de aplicaciones móviles o plataformas web sin que medie contraprestación monetaria.

Recientemente, ha cobrado relevancia el concepto de *inteligencia artificial agente* (*agentic AI*), que hace referencia a sistemas capaces no solo de generar contenido o responder instrucciones, sino también de planear, razonar, tomar decisiones y ejecutar acciones de manera autónoma, para alcanzar objetivos definidos. Estos agentes inteligentes pueden descomponer una tarea compleja en subtareas, buscar información relevante, adaptar sus estrategias en tiempo real y coordinarse con otros agentes o usuarios. Su funcionamiento se apoya en LLM, que proporcionan las capacidades cognitivas necesarias para el procesamiento del lenguaje natural, el razonamiento contextual y la toma de decisiones en entornos dinámicos. Esta evolución expande las fronteras de la IA, más allá de la generación pasiva de contenidos, hacia sistemas proactivos que interactúan con el entorno de forma autónoma y efectiva.

En paralelo, la IA continúa expandiéndose hacia el mundo físico, lo que dio origen a lo que en la actualidad se conoce como *physical AI*. Esta área integra modelos inteligentes con sensores, actuadores y plataformas físicas, permitiendo que los sistemas operen directamente en entornos reales. La IA física habilita a sistemas autónomos —como robots humanoides, vehículos sin conductor, drones o espacios inteligentes—, para percibir su entorno, interpretar situaciones complejas y ejecutar acciones físicas con un alto grado de precisión y adaptabilidad. A diferencia de los modelos digitales que operan de forma exclusiva en entornos virtuales, esta vertiente exige una integración coordinada entre percepción sensorial, razonamiento computacional y control motor. Aunque plantea retos técnicos considerables, como la sincronización en tiempo real o la robustez frente a entornos cambiantes, también abre un campo de

aplicaciones transformadoras en sectores como la robótica de servicio, la automatización industrial y la movilidad inteligente.

El auge de la IA supone nuevas oportunidades y retos. Oportunidades para impulsar el desarrollo humano y retos asociados a usos que exacerban problemas públicos, o que producen nuevos problemas. Como ocurre con toda tecnología, las aplicaciones de diversas herramientas basadas en IA generan tanto efectos negativos como positivos, y dichos impactos tienden a estar heterogéneamente distribuidos en la sociedad (Postman, 2011)[1].

La maximización de los beneficios, la prevención de vulneración a los derechos fundamentales y la debida gestión de los riesgos vinculados con los usos de los sistemas de IA es el resultado de las decisiones que toman múltiples actores a largo del ciclo de vida de estas tecnologías: desarrolladores, comercializadores, responsables del despliegue, usuarios y quienes participan en la gobernanza de las herramientas basadas en IA, entre otros (Organización de las Naciones Unidas para la Educación, la Ciencia y la Cultura [Unesco], 2024).

Por último, vale la pena resaltar que el campo de estudio de estas tecnologías es muy dinámico, no solo por los constantes avances técnicos y las nuevas aplicaciones de los sistemas de IA, sino también por las variadas implicaciones sociales y las consecuencias políticas y económicas asociadas a su implementación en los sectores públicos y privados. En el futuro cercano, lo que entendemos por IA evolucionará a medida que estos sistemas ejecuten nuevas funciones con un mayor grado de autonomía y que estos cambios tecnológicos generen nuevos retos públicos.

Contribuciones de la Universidad de los Andes a la teoría y práctica de la inteligencia artificial

Los aportes de la Universidad de los Andes a la transformación digital en Colombia datan de comienzos de la década de los sesenta. En 1963, la Facultad de Ingeniería instaló el primer computador en una universidad colombiana, el IBM 650, y lo inauguró con un curso sobre "aplicación de computadores al diseño de carreteras" (Aristizábal, 2004, p. 105).

1 Esta idea la expresó el filósofo Neil Postman en la introducción de *Technopoly* (2011), a propósito de su reflexión sobre el intercambio descrito en los *Diálogos* de Platón entre el rey Thamus y el dios Theuth, en los siguientes términos: "Es un error suponer que toda innovación tecnológica tiene un efecto unidireccional. Toda tecnología es a la vez una carga y una bendición; no es una cosa o la otra, sino ambas a la vez" ["it is a mistake to suppose that any technological innovation has one-sided effect. Every technology is both a burden and a blessing; not either-or, but this-and-that"] (pp. 11-12).

Posteriormente, en 1980, Manuel Dávila, un ingeniero de sistemas de la Universidad de los Andes, cofundó la primera empresa de importación de microcomputadores del país y fue pionero en el desarrollo de *software* para empresas (Montes, 2004).

En la segunda mitad de los años ochenta la Universidad realizó los primeros proyectos para conectarse a redes internacionales de computadores, las precursoras de lo que hoy conocemos como la World Wide Web. Luego, en la década de los noventa, la Universidad contribuyó a la difusión del acceso a internet en Colombia (Borja Gómez, 2022).

La Universidad de los Andes también ha sido precursora en el campo de la IA. En noviembre del 2020 fue inaugurado el Centro de Investigación y Formación en Inteligencia Artificial (CinfonIA), el primer centro de investigación académico en esta materia fundado en América Latina. Dentro de los muchos proyectos liderados por CinfonIA se destaca Guacamaya, una colaboración internacional con Microsoft, el Instituto Alexander von Humboldt y el Instituto Amazónico de Investigaciones Científicas SINCHI, que aplica IA para el monitoreo y la conservación de la Amazonía. En este proyecto se han desarrollado algoritmos avanzados capaces de procesar grandes volúmenes de datos bioacústicos, imágenes de cámaras trampa y fotografías satelitales, facilitando un análisis acelerado y preciso de la biodiversidad y la deforestación en la región amazónica (Universidad de los Andes, 2023).

Además, CinfonIA ha establecido una alianza significativa con Google DeepMind, una de las empresas líderes en IA, para fortalecer y diversificar la comunidad que está alrededor de estas investigaciones. A través del programa de becas de Google DeepMind, se han financiado estudios de maestría de estudiantes en la Universidad. Esta iniciativa no solo promueve la excelencia académica, sino que también proporciona recursos y mentoría especializada, apoyando a los becarios en su desarrollo personal y profesional en este campo.

Desde CinfonIA y otras iniciativas lideradas por profesores e investigadores de diferentes facultades, la Universidad de los Andes ha contribuido al estudio y desarrollo de la IA a través de procesos de investigación e innovación, enseñanza, desarrollo institucional y contribución a la agenda pública.

En el 2023, la Universidad obtuvo el registro calificado para la Maestría en Inteligencia Artificial, actualmente ofrecida en modalidad virtual en alianza con Coursera. Esta maestría, la primera en español impartida de forma virtual en la plataforma, ha sido un éxito desde su lanzamiento. Para el primer semestre del 2025, cuenta con cerca de 340 estudiantes inscritos, consolidándose como uno de los programas de posgrado más exitosos de la Universidad.

Además, en el primer semestre del 2024, la Escuela de Gobierno de la Universidad de los Andes entrenó a través de un curso en línea de Educación Continua a 1400 magistrados, jueces y servidores judiciales en fundamentos de IA para la administración de justicia. Este curso en línea de 50 horas (35 sincrónicas, 15 asincrónicas) fue pionero a nivel global tanto por las temáticas desarrolladas como por el número de estudiantes que lo tomaron.

En relación con los aportes de la Universidad a la agenda pública, por ejemplo, desde finales del 2023 la Escuela de Gobierno ha organizado junto con la Universidad Externado de Colombia ocho sesiones de la mesa de trabajo multiactor sobre regulación de IA en Colombia. En dichas sesiones se han abordado temáticas variadas como las implicaciones de la IA para los derechos de propiedad intelectual, la administración de justicia, la protección de los datos personales, la democracia y los derechos humanos.

Además, las investigaciones y aportes de los profesores de la Escuela de Gobierno han sido citados en documentos Conpes de política pública sobre IA, hojas de ruta para el desarrollo y la aplicación de la IA publicadas por ministerios, y sentencias de la Corte Constitucional, como la T-323 del 2024, que resolvió sobre el uso de IA en la administración de justicia, y la T-067 del 2025, sobre transparencia algorítmica en el sector público.

Más recientemente, en marzo del 2025, la Escuela de Gobierno lanzó el proyecto Sistemas de Algoritmos Públicos, que busca contribuir al conocimiento sobre los sistemas algorítmicos utilizados en el sector público, así como a la gobernanza de estas herramientas. En la plataforma en línea del proyecto están disponibles repositorios que documentan casi ochocientas herramientas de IA piloteadas o adoptadas por entidades públicas de América Latina y el Caribe, y casi seiscientos regulaciones y proyectos de regulación relacionados con estas tecnologías en la región (Sistemas de Algoritmos Públicos, 2025)[2].

Por otra parte, en octubre del 2024, la Universidad publicó los *Lineamientos para el uso de inteligencia artificial generativa (IAG) en la Universidad de los Andes*, un documento pionero en América Latina que orienta a estudiantes, profesores, investigadores y empleados administrativos sobre el uso responsable de este tipo de herramientas en actividades pedagógicas, investigativas y administrativas (Universidad de los Andes, 2024).

Por último, en noviembre del 2024, la Universidad importó el primer computador cuántico a Colombia, con la finalidad de que sus estudiantes tengan la oportunidad de profundizar en el aprendizaje e investigación sobre física y computación cuántica (Laguna Cardozo, 2024).

2 Los repositorios pueden consultarse en la plataforma: https://algoritmos.uniandes.edu.co/

Contribuciones a las conversaciones globales sobre la inteligencia artificial

Las contribuciones que componen este libro buscan responder preguntas teóricas y prácticas sobre el conjunto de tecnologías que denominamos IA y sus implicaciones sobre las personas, organizaciones y sociedades. El contenido de este volumen participa en conversaciones globales sobre el desarrollo y las aplicaciones de estas tecnologías, las interacciones entre tecnología y humanidad, y la gobernanza de la IA.

Este libro es útil e interesante para múltiples audiencias. Se estructura en tres secciones. La primera, "Teoría y métodos avanzados de la inteligencia artificial", está dividida en cuatro capítulos. En el primero, Giraldo, Quijano y Manrique investigan la IA distribuida (DAI) y sus aplicaciones para resolver problemas complejos, mediante la interacción de múltiples agentes en entornos compartidos. Así, ofrecen un panorama general de la evolución de la DAI y destacan cómo ha integrado conceptos de diversas disciplinas para abordar la cooperación y la toma de decisiones en sistemas multiagente. Luego, se enfocan en mejorar la resiliencia de comunidades vulnerables a través de estrategias de cooperación financiera, por medio de simulaciones y modelos dinámicos. Finalmente, exponen el uso de modelos de lenguaje de gran tamaño (*large language model*, LLM) y agentes autónomos en la gestión de dilemas sociales, sugiriendo oportunidades para futuras investigaciones.

En el segundo capítulo, Cardozo y Dusparic abordan el desarrollo de sistemas de adaptación dinámica (SAS) y cómo estos pueden ajustarse proactivamente a entornos cambiantes. Analizan la estructura fundamental para construir SAS, que incluye la definición del comportamiento base, la identificación de condiciones para la adaptación y la especificación del comportamiento especializado. Sin embargo, reconocen que los SAS tradicionales tienen una capacidad limitada para adaptarse a situaciones no previstas durante el diseño. Para superar estas limitaciones, los autores proponen mecanismos de aprendizaje dinámico, como Auto-COP y ComInA, que permiten que los SAS generen y compongan adaptaciones en respuesta a situaciones completamente desconocidas, aumentando de manera significativa su flexibilidad y capacidad de respuesta.

Por su parte, en el tercer capítulo, Manrique Gómez y Borja Gómez exploran nuevas formas de hacer investigación histórica a partir de la integración de herramientas de aprendizaje automático. Con ese fin, presentan el estado del arte del procesamiento de archivos multimodales (textos, manuscritos, imágenes y otros), analizan la contribución del proyecto Arte Colonial Americano (ARCA) de la Universidad de los Andes y discuten las implicaciones futuras de la

multidimensionalidad de las fuentes históricas en relación con el quehacer de los historiadores (por ejemplo, nuevas preguntas de investigación y desafíos éticos).

En el último capítulo de la primera sección, Páez examina qué significa que un algoritmo sea transparente y por qué la opacidad (jurídica y epistémica) de los algoritmos puede implicar retos éticos para el desarrollo y uso ético de sistemas de IA. El texto reflexiona sobre los diferentes tipos de opacidad algorítmica, los retos de los métodos de explicabilidad para proveer un mayor grado de comprensión sobre el funcionamiento de algoritmos y sobre la relación entre transparencia y confianza en el contexto de las decisiones automatizadas basadas en sistemas de IA.

La segunda sección del libro, "Aplicaciones y efectos de la inteligencia artificial", está compuesta por cuatro capítulos. Esta inicia en el capítulo cinco, en el cual Núñez Castro, Calderón Romero, Díaz Meza, Sierra Ramírez y Vásquez Pachón investigan el uso de técnicas de IA para el análisis de imágenes satelitales, con el objetivo de mejorar la comprensión de los cambios ambientales y apoyar el desarrollo sostenible y la gestión de recursos naturales. Utilizando imágenes satelitales y técnicas avanzadas de aprendizaje automático, presentan dos aplicaciones de *software* diseñadas para apoyar la toma de decisiones en políticas medioambientales. La primera aplicación es un atlas interactivo para estimar el potencial energético de la biomasa residual poscosecha, mientras que la segunda se centra en la monitorización del crecimiento periférico de ciudades para promover un desarrollo urbano sostenible y planificado.

En el sexto capítulo, Mariño Drews, Paquette y Manrique Piramanrique investigan los beneficios de la web semántica en la educación superior. Discuten las principales ventajas de esta tecnología de IA simbólica en actividades educativas, como el diseño de cursos, la personalización del aprendizaje y el apoyo en la búsqueda de recursos. Además, presentan resultados de más de una década de investigación y desarrollo en este ámbito en la Télé-Université de Quebec y la Universidad de los Andes.

En el séptimo capítulo, Bustamante Duarte, Pajarito Grajales, Portela y Parra Agudelo reflexionan de manera crítica sobre los recientes desarrollos en IA enfocada en aspectos geoespaciales (GeoAI) aplicada al entendimiento de los territorios y comunidades urbanas y rurales del sur global. El capítulo plantea que estas sociotecnologías, pese a diversas ventajas en nuestros contextos de eficiencia y recursos, necesitan ser analizadas cuidadosamente, entendiendo que no siempre reflejan la realidad y necesidades a nivel socioespacial de ciertas comunidades, debido a temas de sus sistemas de clasificación y categorías. De esta manera, la discusión se centra en casos de GeoAI en los que el uso de datos geoespaciales implícitos y participativos ha permitido responder a dichos retos.

En el último capítulo de la segunda sección, Arbeláez, Saldarriaga, Quintero, Castillo, Puentes, Calderón, Osejo, Castañeda, Paz, Hernández y Agudelo investigan cómo la IA puede contribuir a combatir la explotación sexual de niñas, niños y adolescentes en línea (OCSEA). Para ello, desarrollaron dos modelos de IA. El primero se centró en optimizar el análisis de los reportes sobre sextorsión, *sexting*, *grooming* y ciberacoso sexual recibidos por la línea de reporte Te Protejo en Colombia, lo que contribuye a los analistas a clasificar los casos según su gravedad y tipo de daño y reduce el riesgo de exposición a material perjudicial. El segundo modelo se diseñó como un prototipo de sistema de alerta que analiza información en foros de la web profunda, identificando patrones de conversación que podrían indicar actividades de OCSEA y sus posibles efectos en las víctimas.

La tercera sección cierra el libro con dos capítulos sobre "La inteligencia artificial y el Estado". En el noveno capítulo, Gutiérrez y Muñoz-Cadena examinan cómo los sistemas de IA adoptados por entidades públicas en América Latina pueden contribuir con el agendamiento, formulación, implementación y evaluación de políticas públicas. El texto se fundamenta en una nueva base de datos que documenta más de 700 herramientas adoptadas por entidades públicas en veintitrés países de la región y en casos de estudio de Argentina, Brasil, Chile, Colombia, Guatemala, Honduras, México y Perú.

Por último, en el capítulo que cierra esta compilación, Díaz-Valderrama, Niño-Machado, Guerrero-C. y González-Uribe rastrean los imaginarios a futuro sobre la IA y su relación con el Estado, así como las acciones estatales que buscan responder de manera anticipada a la creciente incursión de estas tecnologías en la sociedad colombiana. El capítulo identifica en el discurso mediático cuatro campos de imaginación sociotécnica: (1) innovación vs. regulación, (2) cuarta revolución industrial, (3) primicia, liderazgo y modernización regional, y (4) desarrollo y presencia nacional amplificada.

En conjunto, los capítulos que componen este libro ofrecen nuevo conocimiento sobre la IA a quien esté interesado en aprender sobre sus conceptos teóricos, aplicaciones prácticas e implicaciones presentes y futuras. Esperamos además que cada lector encuentre aquí la chispa que estimule su curiosidad y lo motive a involucrarse de manera crítica y constructiva en el área. Que este libro inspire a pensar más allá, innovar y aplicar de manera ética y responsable los avances de la IA en beneficio de todos.

Referencias

Alpaydin, E. (2020). *Introduction to machine learning*. Massachusetts Institute of Technology Press.

Aristizábal, J. C. (2004). Del 650 al 360: Los primeros computadores de la Universidad de los Andes. *Revista de Ingeniería*, *1*(20), 105-107. https://doi.org/10.16924/revinge.20.14

Borja Gómez, J. H. (2022, 21 de junio). Internet y la web en Colombia: Una historia de sus primeros años. *Credencial Historia*. https://www.banrepcultural.org/biblioteca-virtual/credencial-historia/numero-388/internet-y-la-web-en-colombia-una-historia-de-sus

Gutiérrez, J. D. (2024, 15 de noviembre). De qué hablamos cuando hablamos de IA. *Foro Administración, Gestión y Política Pública* (blog). https://forogpp.com/2024/11/15/de-que-hablamos-cuando-hablamos-de-ia/

Laguna Cardozo, M. (2024, 27 de noviembre). El primer computador cuántico llega a Colombia. *Noticias Universidad de los Andes*. https://www.uniandes.edu.co/es/noticias/ciencias-aplicadas/el-primer-computador-cuantico-llega-a-colombia.

McCarthy, J., Minsky, M. L., Rochester, N. y Shannon, C. E. (1955). A proposal for the Dartmouth Summer Research Project on artificial intelligence. http://jmc.stanford.edu/articles/dartmouth/dartmouth.pdf

Montes, A (2004, 30 de mayo). Marzo 3 de 1957: La máquina que cambió al país. *Semana*. https://www.semana.com/especiales/articulo/marzo-1957-brla-maquina-cambio-pais/65917-3/.

Organización de las Naciones Unidas para la Educación, la Ciencia y la Cultura (Unesco). (2024). *Consultation paper on AI regulation: Emerging approaches across the world*. Unesco. https://unesdoc.unesco.org/ark:/48223/pf0000390979

Organización para la Cooperación y el Desarrollo Económico (OECD). (2022). OECD framework for the classification of AI systems. OECD Digital Economy Papers. https://www.oecd.org/content/dam/oecd/en/publications/reports/2022/02/oecd-framework-for-the-classification-of-ai-systems_336a8b57/cb6d9eca-en.pdf

Postman, N. (2011). *Technopoly: The surrender of culture to technology*. Vintage.

Sistemas de Algoritmos Públicos. (2025). *Informe de los repositorios 2.0*. Escuela de Gobierno, Universidad de los Andes. https://sistemaspublicos.tech/wp-content/uploads/Informe-de-Repositorios-Proyecto-SAP-v2.0.pdf

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, *LIX* (236), 433-460. https://doi.org/10.1093/mind/LIX.236.433

Universidad de los Andes. (2023). "Guacamaya", para salvar la Amazonía con inteligencia artificial. https://uniandes.edu.co/es/noticias/ambiente-y-sostenibilidad/guacamaya-para-salvar-la-amazonia-con-inteligencia-artificial

Universidad de los Andes. (2024). Lineamientos de uso de inteligencia artificial generativa (IAG) en la Universidad de los Andes. https://secretariageneral.uniandes.edu.co/images/documents/lineamientos-uso-inteligencia-artificial-generativa-IAG-uniandes.pdf

PARTE I

TEORÍA Y MÉTODOS AVANZADOS DE LA INTELIGENCIA ARTIFICIAL

INTELIGENCIA ARTIFICIAL DISTRIBUIDA: EVOLUCIÓN Y APLICACIONES

Luis Felipe Giraldo, Rubén Manrique, Nicanor Quijano

Para citar este capítulo:
https://doi.org/10.51573/Andes.9789587988444.9789587988451.9789587988468.01

Introducción

La inteligencia artificial distribuida (*distributed artificial intelligence*, DAI) involucra la interacción de múltiples agentes dentro de un entorno compartido, para alcanzar objetivos individuales y colectivos. Este paradigma permite modelar y simular dinámicas complejas, ofreciendo una perspectiva única para abordar problemas en una amplia gama de dominios, como la optimización de recursos, el transporte inteligente, la gestión de redes y el estudio de la cooperación y resiliencia en comunidades, lo que contribuye al desarrollo de políticas públicas. La interacción entre agentes facilita la resolución de problemas difíciles o imposibles de tratar mediante enfoques tradicionales, al mismo tiempo que promueve la emergencia de comportamientos colectivos y soluciones adaptables.

En este capítulo se presentan algunos de los desafíos y soluciones propuestas por investigadores de la Universidad de los Andes en este campo, abarcando aplicaciones de la inteligencia artificial (IA) que van desde la teoría de juegos evolutiva y el aprendizaje por refuerzo multiagente, hasta agentes cooperativos basados en los hoy populares modelos de lenguaje de gran tamaño (*large language model*, LLM).

Primero, se ofrece un panorama general de la evolución de la DAI, destacando cómo esta disciplina ha integrado conceptos de la sociología, el control, la economía y la teoría de juegos para abordar problemas de cooperación y toma de decisiones en sistemas multiagente. Esta sección presenta aquellos avances y aplicaciones principalmente lideradas por el profesor Nicanor Quijano, que sustentan el desarrollo de sistemas donde múltiples agentes interactúan en un entorno compartido, toman decisiones autónomas y optimizan su desempeño colectivo.

A continuación, el texto se enfoca en la aplicación de estas nociones para mejorar la resiliencia de comunidades vulnerables, mediante estrategias de cooperación financiera. Se examinan los desafíos que enfrentan las comunidades, como la incertidumbre en los ingresos y la falta de acceso a servicios financieros, y cómo las herramientas basadas en DAI pueden ofrecer soluciones innovadoras. Basados en la investigación liderada por el profesor Luis Felipe Giraldo, se analizan casos específicos de asociaciones de ahorro y crédito, y se evalúa su efectividad por medio de simulaciones de Monte Carlo y modelos dinámicos que permiten optimizar las estrategias de cooperación, según las características de cada comunidad.

Finalmente, se aborda el uso de LLM y agentes autónomos basados en estos modelos, destacando su capacidad para tomar decisiones y cooperar en entornos complejos. A través de experimentos en entornos simulados, liderados por el profesor Ruben Manrique, se evalúa la eficacia de estos agentes en la gestión de dilemas sociales y se resaltan tanto sus capacidades actuales como las áreas que requieren mejoras. Esta sección presenta un análisis de los desafíos y las oportunidades que presentan estas tecnologías, así como caminos para futuras investigaciones que podrían mejorar la cooperación y la resiliencia en una variedad de contextos, desde aplicaciones comunitarias hasta entornos más amplios, como la ciberseguridad y la gestión de infraestructuras críticas.

Evolución de la inteligencia artificial distribuida

La IA se ha inspirado y ha utilizado la psicología y el comportamiento no solo en el desarrollo teórico, sino también a nivel metafórico. Además de las aplicaciones tradicionales de reconocimiento de patrones, imágenes y características, la IA moderna tiene aplicaciones que buscan tomar decisiones basadas en conocimientos adquiridos. Estas aplicaciones modernas de IA se están transformando desde el reconocimiento de características puras (por ejemplo, detectar un gato en una imagen) a la toma de decisiones (conducir vehículos autónomos a través de un cruce de tráfico de manera segura), donde emerge la interacción entre diferentes actores (Yang y Wang, 2020). La aproximación moderna a la IA está centrada alrededor del concepto de *agente racional*. Un agente es un ente computacional (programa) o físico (robot), el cual puede percibir (sensar) y actuar en un ambiente; es autónomo y por lo general su comportamiento dependerá en parte de su experiencia y de la retroalimentación que le brinde el lugar. Esta flexibilidad y racionalidad se logran sobre la base de procesos clave, como la toma de decisiones, la planificación y el aprendizaje. Como este agente interactúa con el ambiente, su comportamiento se verá afectado por otros

agentes o entes humanos, y esta interacción puede ser de índole cooperativa o competitiva (Russell y Norvig, 2016). Un agente que siempre trata de optimizar una medida de desempeño determinada se denomina *agente racional*. Al ser tan general esta definición, se podría ligar a agentes humanos (poseen ojos como sensores y manos como actuadores), agentes robóticos (cámaras y llantas) o agentes de *software* (interfaz gráfica que cumple ambas labores de sensado y actuación). En otras palabras, desde esta perspectiva, la IA se concibe como el estudio de las bases y el diseño de agentes artificiales racionales. Como en la mayoría de los casos estos agentes no están solos e interactúan con otros agentes, estaríamos hablando de sistemas de múltiples agentes (*multi-agent system*, MAS) y de ahí que se derive el subcampo de investigación de la DAI (Vlassis, 2022).

La DAI se podría definir como el estudio, construcción y aplicación de MAS, por ejemplo, sistemas en los que diversos agentes inteligentes interactúan, en busca de un propósito común o lograr una serie de tareas definidas (Weiss, 1999). Estos múltiples agentes por lo general son heterogéneos (p. ej., han sido diseñados y concebidos en *hardware* de diferentes formas); interactúan en ambientes dinámicos (en su mayoría, un solo agente suele ser entrenado en un ambiente estático, pero al interactuar con otros su comportamiento varía con el tiempo y, por ende, se requiere un mejor desarrollo de la parte matemática [Gómez *et al.*, 2022]); la información que se tiene de los sensores es distribuida (p. ej., a nivel espacial o temporal), lo que hace que el mundo percibido por cada agente sea parcialmente observable; el control de los agentes es descentralizado (p. ej., cada agente toma su propia decisión y no está atado a un ente central, debido a los problemas de robustez y tolerancia a fallos; en este caso, los problemas de toma de decisiones pueden resolverse mediante la teoría de juegos); y la interacción entre agentes depende en gran medida de la comunicación que puede facilitar la coordinación y la cooperación entre individuos, como se haría en la naturaleza (p. ej., la forma en la que las abejas se comunican entre sí a la hora de buscar alimento [Quijano y Passino, 2010; Giraldo *et al.*, 2015]) (Vlassis, 2022).

Las primeras investigaciones en DAI datan de los años setenta y ochenta, cuando se emplearon modelos computacionales para simular MAS, combinando elementos de la teoría de juegos, la simulación de Monte Carlo, la programación evolutiva y las teorías de la emergencia y los sistemas complejos. La DAI es un área que toma ideas, conceptos y resultados de disciplinas como la IA, las ciencias de la computación, la sociología, la economía, la filosofía, el manejo de las organizaciones, los sistemas de control, entre otros. Así, se podría decir que la IA tradicional utiliza la psicología y el comportamiento para sus ideas, su inspiración y como metáfora; la DAI utiliza la sociología, la economía y el control, por lo que se percibe como una generalización de la IA. Hay dos razones para

tratar con DAI: (1) los MAS desempeñan un papel fundamental en los sistemas tecnológicos de hoy en día. Los sistemas de cómputo y control que encontramos desplegados en diferentes ámbitos tienen como características ser distribuidos y heterogéneos. Por lo tanto, se requiere ver a estos elementos como agentes en lugar de partes, con el fin de interactuar de una manera más adecuada, sobre todo en esta era de la cuarta revolución industrial; y (2) los MAS tienen la capacidad de interactuar dentro de lo que llamaríamos *cyber-physical human systems* (CPHS), con lo cual hay que pensar en el desarrollo de sistemas autónomos que interactúen no solo en su entorno, sino que tengan en cuenta al humano o los seres vivos con los que lo hace (Annaswamy *et al*., 2023).

Como se puede observar, hay varios conceptos teóricos que forman parte de la idea general de DAI. En primera instancia, podríamos hablar de *cooperación*. Los problemas de cooperación, en los que los agentes tienen oportunidades de mejorar su bienestar común, pero son notablemente capaces de hacerlo, son omnipresentes e importantes. Es posible encontrarlos en todas las escalas, desde nuestras rutinas diarias, como conducir, programar reuniones y trabajar en colaboración, hasta nuestros retos globales, como el experimentado recientemente a la hora de prepararnos para pandemias (Dafoe *et al*., 2020). Otros trabajos, como el del politólogo Robert Axelrod (1984), muestran cómo esta cooperación emerge en áreas como la política, lo cual fue uno de los elementos que nos salvó de tener conflictos de mayor envergadura durante la época de la Guerra Fría. La investigación de la IA relacionada con la cooperación se ha llevado a cabo en muchas áreas diferentes, dentro de las que se incluyen los MAS, la teoría de juegos y la elección social, la interacción y alineación entre el ser humano y la máquina, el procesamiento del lenguaje natural y la construcción de herramientas y plataformas sociales.

La teoría de juegos es el nombre que se le da a la metodología que utiliza herramientas matemáticas para modelar y analizar situaciones en las que se toman decisiones de forma interactiva. Estas nociones se habían comenzado a discutir a finales del siglo XIX, principios del siglo XX, pero es hasta el desarrollo del teorema del minimax, por parte de John von Neumann y el libro que publica con Oskar Morgenstern (Von Neumann y Morgenstern, 2007), que se concretan las ideas iniciales. La diseminación del concepto toma varios años, y es gracias a los aportes de la corporación RAND que estas nociones logran permear diferentes áreas del conocimiento (Bhattacharya, 2021). La investigación en *machine learning* (ML) y aprendizaje por refuerzo (*reinforcement learning*, RL) se ha centrado en casos de conflicto de intereses y, en particular, en entornos con dos jugadores de suma cero (Dafoe *et al*., 2020). En el ámbito del aprendizaje por refuerzo basado en juegos, en los últimos años se han visto enormes avances

en los juegos de suma cero de dos jugadores, como el ajedrez, Go, StarCraft II y póquer de dos jugadores. Este tipo de juegos fueron un dominio productivo para la investigación inicial de múltiples agentes, ya que son especialmente tratables: la solución minimax coincide con el equilibrio de Nash y se puede calcular en tiempo polinomial a través de un programa lineal, sus soluciones son intercambiables y tienen garantías de peor caso (Von Neumann y Morgenstern, 2007). Esta tratabilidad puede explicar por qué estos juegos han recibido una considerable atención de investigación, a pesar de ser relativamente raros en el mundo real y en el espacio de posibles juegos.

Otro de los aspectos relevantes de los DAI es el problema de toma de decisiones. Este problema está ligado al área de control óptimo[1]; en este punto es donde la programación dinámica desempeña un papel fundamental (Vlassis, 2022). Varios ejemplos se dan hoy en cuanto a problemas asociados a toma de decisiones. Por ejemplo, en vehículos autónomos (VA) se tienen dos grandes problemas: (1) en cada instante de tiempo, durante la toma de decisión, el agente no solo debe considerar sus acciones presentes, sino también las consecuencias de las acciones futuras; y (2) para determinar las decisiones correctas y seguras, se deben tener en cuenta las acciones y los comportamientos de los demás agentes y elementos en el ambiente en el que se desenvuelven (Yang y Wang, 2020; Detjen-Leal *et al*., 2023). La necesidad de tener un marco adaptativo de toma de decisiones, además de la complejidad de tener interacción con múltiples aprendices, lleva al desarrollo del aprendizaje por refuerzo de múltiples agentes (*multi-agent reinforcement learning*, MARL). Esta noción busca resolver problemas de toma de decisiones secuenciales con agentes inteligentes, que operan en un ambiente compartido con otros agentes y con cierta incertidumbre y estocasticidad, donde cada ente buscará maximizar sus recompensas mediante la interacción con el ambiente y otros agentes. Históricamente, el mecanismo de RL se desarrolló basado en el estudio del comportamiento de los gatos en una caja (Thorndike, 1898). En 1954, Minsky propuso por primera vez el modelo computacional de RL en su tesis de doctorado y nombró su máquina analógica resultante la calculadora estocástica de refuerzo neural-analógico. En 1961, Minsky sugirió por primera vez la conexión entre la programación dinámica (Bellman, 1952) y el RL (Minsky, 1961). Más adelante, Bertsekas y Tsitsiklis (1996) propusieron métodos aproximados de programación dinámica

1 R. Bellman es el creador de la programación dinámica, mientras que Pontryagin es uno de los pioneros en el cálculo de variaciones. Estas nociones son la base del control óptimo, en cuya área autores como D. Bertsekas y J. Tsitsiklis han realizado aportes fundamentales.

fundamentados en redes neuronales, lo cual muestra de forma clara la relación entre las nociones de aprendizaje y control. En el dominio de control clásico se han estudiado extensamente los enfoques basados en modelos en los que el agente de aprendizaje primero construirá un "modelo" explícito de espacio de estado, para comprender cómo funciona el entorno en términos de dinámica de transición de estado y función de recompensa, y luego aprenderá del "modelo". La ventaja de los algoritmos basados en modelos reside en el hecho de que a menudo requieren muchas menos muestras de datos del entorno. En principio, la comunidad MARL ha trabajado con enfoques basados en modelos, por ejemplo, el famoso algoritmo R-MAX, de hace casi dos décadas. Sorprendentemente, los desarrollos en la línea de sistemas basados en modelos se detuvieron desde entonces; teniendo en cuenta los impresionantes resultados que estos enfoques han demostrado en las tareas de RL de un solo agente, los métodos de MARL merecen más atención de la comunidad (Yang y Wang, 2020).

Por otro lado, la toma óptima de decisiones por parte de múltiples agentes que interactúan entre sí por medio de una red (sea física o de comunicaciones) ha sido uno de los temas que más auge ha tenido en la comunidad de sistemas dinámicos y de control en las últimas décadas (Obando *et al*., 2024). ¿La razón? Estos problemas emergen en ingeniería, ciencias sociales y económicas, sistemas urbanos o inteligencia artificial, donde se encuentran aplicaciones como el análisis de redes sociales (Jackson, 2008), el manejo o control de redes inteligentes (Mojica-Nava *et al*., 2013; Ananduta *et al*., 2018), las redes inalámbricas (Han *et al*., 2019), la ciberseguridad (Pawlick y Zhu, 2021), la infraestructura crítica (Rass *et al*., 2020) o los sistemas ciberfísicos (Groot *et al*., 2014).

Una de las maneras de modelar estos sistemas complejos de gran escala con múltiples decisiones y acciones, en la que los diferentes entes/controladores interactúan entre sí, es mediante el uso de la teoría de juegos (Muros, 2021; Carrasco Martínez *et al*., 2023). Dentro de los trabajos desarrollados, es pertinente mencionar el de Bacci *et al.* (2016), el cual muestra la relación entre los juegos, la optimización y el aprendizaje para el procesamiento de señales en red. Otros ejemplos similares incluyen la carga de vehículos eléctricos (Grammatico *et al*., 2016), la coordinación de redes de robots (Jaleel y Shamma, 2020; Park y Barreiro-Gómez, 2023), los problemas de congestión vehicular (Kara *et al*., 2022), el control de pandemias y epidemias (Martins *et al*., 2023), las técnicas de aprendizaje por refuerzo (Gao y Pavel, 2021), la respuesta a la demanda (Genis-Mendoza *et al*., 2022) y el manejo de recursos y regulación de sistemas de agua (Lu *et al*., 2022). En estos trabajos, los autores abordan los problemas desde diferentes ángulos de la teoría de juegos. Algunos parten de los *juegos matriciales*, lo cuales son conocidos por su forma normal, en la que la interacción simultánea entre jugadores

se produce de manera estática y cada jugador se entiende como un ente individual. Por otra parte, en *juegos continuos* los jugadores pueden elegir entre una amplia gama de estrategias, las cuales cambian con el tiempo.

Hay otros llamados *juegos dinámicos*, que usan un mecanismo de aprendizaje que permite ajustar las acciones basadas en eventos previos. Estos juegos se distinguen por tres problemas principales: (1) modelar el ambiente en el que interactúan los jugadores; (2) modelar los objetivos que persiguen los jugadores; y (3) describir el orden en el que los jugadores toman decisiones y la cantidad de información que tienen. En este caso, se asume que la interacción ocurre entre un gran número (desconocido) de jugadores, y lo que nos interesa estudiar es la proporción de individuos que utiliza una estrategia u otra. Los *juegos evolutivos*, que fueron creados con base en el comportamiento ecológico, se clasifican como juegos dinámicos. Las nociones de las estrategias evolutivamente estables fueron desarrolladas por Maynard-Smith y Price (1973), quienes fueron los pioneros de este concepto. Después, Taylor y Jonker (1978) crearon el modelo de replicadores (*replicator dynamics*), que se utiliza ampliamente en aplicaciones de ingeniería, para examinar el comportamiento dinámico y su relación con la parte genética. De igual manera, mediante la introducción de protocolos de revisión y *mean dynamics*, se han desarrollado aproximaciones de juegos evolutivos desde el punto de vista de sistemas económicos (Sandholm, 2010).

El trabajo de Obando *et al.* (2024) presenta algunos ejemplos de asignación dinámica de recursos a través de juegos poblacionales y modelos dinámicos de pago (Park *et al.*, 2019). Esta investigación destaca la utilidad y la idoneidad de estas técnicas para modelar dinámicas de sistemas complejos de ingeniería, así como para diseñar estrategias de gestión y control, de acuerdo con políticas particulares y restricciones físicas y operativas, tanto locales como globales. Las estrategias desarrolladas por medio de este paradigma son de fácil implementación física, como se observa en distintos trabajos (Martínez-Piazuelo *et al.*, 2022a; J. Barreiro-Gómez *et al.*, 2021), en los que se muestran criterios para seleccionar parámetros y su implementación. Por otra parte, este nuevo paradigma también es capaz de abarcar problemáticas como los retrasos (Obando *et al.*, 2016; Park y Leonard, 2021), lo que ofrece una nueva alternativa respecto a otras técnicas desarrolladas, cuya implementación tiene sus dificultades. Algunas ideas que se presentan en ese artículo se derivan de la evolución de trabajos anteriores; se puede encontrar un resumen de los trabajos que se presentaron hasta el 2017 en Quijano *et al.* (2017). Desde entonces, se han hecho contribuciones en términos de dinámicas distribuidas en tiempo continuo (Barreiro-Gómez *et al.*, 2017a) y en tiempo discreto (Martínez-Piazuelo *et al.*, 2022a), así como en la combinación de técnicas en sistemas híbridos (Ochoa *et al.*, 2021). De igual modo, se ha trabajado

en temas como aplicaciones para vehículos autónomos no tripulados (Barreiro-Gómez *et al*., 2021) o la combinación de técnicas de control para aplicaciones en redes de agua (Barreiro-Gómez *et al*., 2017b; Obando *et al*., 2022). Además, se han reportado recientemente resultados significativos sobre la relación entre estas poblaciones dinámicas y los equilibrios generalizados de Nash (Martínez-Piazuelo *et al*., 2022b, 2022c, 2022d, 2023; Sánchez-Amores *et al*., 2023).

Inteligencia artificial distribuida para resiliencia comunitaria

Las personas con ingresos muy bajos que pertenecen a comunidades desfavorecidas enfrentan un desafío significativo al gestionar su vida financiera. Muchas de ellas son trabajadores por cuenta propia y forman parte del 10,7 % de la población mundial que sobrevive con menos de USD 1,90 al día. Estas personas enfrentan continuamente eventos negativos e impredecibles, llamados *shocks*, como problemas de salud o condiciones climáticas adversas en los cultivos. Además, se ven afectadas por la falta de servicios de salud y financieros, y por una escasa formación educativa. Los desafíos de la gestión financiera para individuos de bajos ingresos y las herramientas utilizadas en la actualidad para enfrentar la escasez giran en torno a tres aspectos principales: manejar ingresos inciertos, resistir *shocks* financieros y encontrar estrategias efectivas para ahorrar dinero (Collins, 2009).

El alcance de las instituciones de microfinanzas, que podrían ayudar a mitigar algunos de estos desafíos, sigue siendo limitado, al igual que el uso de la tecnología para brindar asesoramiento financiero. En respuesta a esta situación han surgido estrategias de cooperación financiera informal en diversas partes del mundo, lo que ha fortalecido la resiliencia de las comunidades de bajos ingresos, al fomentar el apoyo mutuo entre sus miembros. Estas estrategias, basadas en planes de ahorro y crédito, son relativamente fáciles de implementar y ofrecen a los participantes una medida de estabilidad financiera. Ejemplos de estos esquemas de cooperación incluyen la asociación rotativa de ahorro y crédito (*rotating savings and credit associations*, ROSCA) y la asociación de ahorro y crédito acumulativo (*accumulating savings and credit associations*, ASCA) (Bouman, 1995; Zambrano *et al*., 2023). En una ROSCA, un grupo de personas acuerda reunirse de manera periódica para contribuir con una cantidad fija de dinero a un fondo común. En cada reunión, uno de los miembros recibe la totalidad del fondo, lo que le proporciona una suma significativa de dinero en ese momento. Este proceso continúa hasta que todos los miembros hayan recibido el fondo en alguna de las rondas. En algunas regiones de Colombia, esta estrategia de cooperación es llamada *cadena* o *natillera* (Salas Bahamón, 2022). Por

otro lado, una ASCA es una forma de ahorro y crédito más estructurada y compleja que una ROSCA; a diferencia de la rotación simple de fondos que ocurre en una ROSCA, en una ASCA los fondos se acumulan y se utilizan para otorgar préstamos con intereses a los miembros del grupo.

En comunidades de bajos ingresos, donde los servicios bancarios son escasos o inaccesibles, las ROSCA y las ASCA ofrecen una manera de participar en actividades financieras sin necesidad de cumplir con requisitos complicados, como historial crediticio, garantías o altos depósitos iniciales. "No me gusta tener que lidiar con otras personas por dinero, pero si eres pobre, no hay alternativa. Tenemos que hacerlo para sobrevivir" (Collins, 2009, p. 13), señala uno de los participantes de estas estrategias de cooperación, resaltando la importancia de estos esquemas en comunidades vulnerables.

Aunque estos esquemas de cooperación son útiles para promover el desarrollo socioeconómico de la comunidad, aún tienen limitaciones para prevenir fallos de gestión cuando varios miembros se ven afectados por *shocks* o cuando alguien decide dejar de participar una vez la asociación está implementada. El diseño de nuevas estrategias de cooperación que minimicen estas limitaciones implicaría varios meses de observación *in situ*, retroalimentación y un proceso de rediseño. Ante esta necesidad, se han propuesto herramientas computacionales basadas en DAI para proporcionar, en un periodo relativamente corto de tiempo, una mejor comprensión del comportamiento de una comunidad que implementa estas estrategias de cooperación y facilitar el diseño de nuevas estrategias o variaciones de estas, minimizando tales desventajas en una amplia variedad de escenarios y tipos de comunidades.

González Villasanti *et al*. (2018) presentan el resultado de una investigación conjunta entre investigadores de la Universidad de los Andes, en Colombia, y la Universidad Estatal de Ohio, en Estados Unidos. Este es un trabajo seminal que marca una línea de estudio en el área, donde se propone un modelo basado en la teoría de control óptimo, que caracteriza la vida financiera de individuos y su participación en asociaciones informales de ahorro y crédito, con el fin de estudiar estrategias optimizadas de gestión de recursos a través de simulaciones computacionales. Primero, se introduce un modelo de toma de decisiones que describe el comportamiento de un individuo en términos de riqueza, salud y educación. Este modelo se basa en la gestión financiera personal y asume que los individuos tienen la capacidad de tomar decisiones que dependen de sus intereses y prioridades, su capacidad para generar riqueza, su condición de salud, su educación actual y eventos inesperados, como *shocks* financieros. Esta toma de decisiones se realiza mediante proyecciones, las cuales se asume que el individuo puede realizar de acuerdo con un horizonte de tiempo. Luego, los

individuos se interconectan para crear una comunidad heterogénea, donde interactúan según una estrategia cooperativa.

En este trabajo se estudiaron dos estrategias de cooperación financiera: la ASCA y una nueva estrategia basada en donaciones. Así, se introdujeron indicadores como la tasa de fracaso en la gestión y el índice de desarrollo comunitario, para cuantificar el rendimiento de la comunidad en términos de las tres dimensiones que describen el desarrollo humano (riqueza, salud y educación). A través de simulaciones de Monte Carlo y estos indicadores de desempeño y resiliencia comunitaria, se evaluaron una gran cantidad de escenarios y poblaciones. Las simulaciones muestran que la ASCA presenta un mejor desempeño que la estrategia de donaciones en comunidades con baja desigualdad en activos y habilidades, mientras que la estrategia basada en donaciones muestra mejores resultados en comunidades desiguales. Estos resultados sugieren que estas estrategias deberían adaptarse a cada grupo para maximizar los impactos positivos.

Esta investigación fue el punto de partida para estudios posteriores liderados por investigadores de la Universidad de los Andes y financiados por Google Research, debido a su pertinencia e impacto. El trabajo propone dos variaciones de las ROSCA convencionales, que potencialmente aumentan la resiliencia de la comunidad (Zambrano *et al*., 2022). Estas estrategias fueron evaluadas por medio de herramientas computacionales. Basado en el trabajo de González Villasanti *et al*. (2018), se emplearon modelos dinámicos para simular escenarios donde se implementan estas estrategias de cooperación financiera. Por un lado, se formula una estrategia que involucra una reserva de efectivo, similar a la usada en instituciones financieras formales, para reducir el impacto de los individuos que abandonan la asociación. Como resultado, se muestra que la ausencia de confianza entre los miembros de la asociación debe ser compensada en igual medida por la cantidad de reserva de efectivo, con el fin de reducir la probabilidad de fracaso en el esquema de cooperación. Por otro lado, se evalúa el desempeño de comunidades en las que sus miembros participan en varias ROSCA al mismo tiempo, visto como una estrategia de participación descentralizada. Se demostró que los individuos deben cooperar en varias asociaciones con un bajo número de miembros, con el objetivo de reducir el tiempo necesario para recibir un retorno de su cooperación y disminuir la probabilidad de perder su dinero, debido a individuos no confiables. Estos resultados son de particular interés en países como Colombia, donde los esquemas financieros de este tipo se consideran ilegales cuando la cantidad de participantes supera un umbral.

Este cuerpo de investigación ha sido fundamental para entender y abordar los desafíos financieros a los que se enfrentan las comunidades de bajos ingresos. Al utilizar herramientas computacionales avanzadas y modelos dinámicos de

simulación, no solo se profundiza en la comprensión de las estrategias de cooperación financiera existentes, sino que también se proponen nuevas soluciones que pueden fortalecer la resiliencia comunitaria ante *shocks* económicos. La relevancia de este trabajo radica en su potencial para influir en políticas públicas y en el diseño de intervenciones más efectivas, las cuales mejoren la estabilidad financiera y el desarrollo humano en contextos vulnerables. La investigación en curso, respaldada por entidades como Google, promete avances significativos en la teoría y práctica de la cooperación financiera, y tiene el potencial de transformar la vida de millones de personas que dependen de estos esquemas para su subsistencia y bienestar.

En la actualidad, investigaciones en curso en la Universidad de los Andes, financiadas por Google a través del Google Research Award y el Google DeepMind Scholar Programme, buscan avanzar en este trabajo mediante el uso de MAS basados en DAI. Estas investigaciones implementan estrategias de aprendizaje avanzadas, como el MARL y el LLM, dentro de un marco de IA cooperativa. El objetivo es comprender cómo se pueden construir comunidades resilientes en escenarios complejos e inciertos, donde no solo interactúan agentes humanos entre sí, sino también humanos con máquinas y máquinas con otras máquinas.

El impacto de la DAI en políticas públicas está en desarrollo y se espera que sus efectos se perciban a medida que se comprendan mejor estas propuestas basadas en modelos computacionales. Los resultados preliminares de esta investigación han contribuido al diseño de intervenciones en comunidades de bajos ingresos, donde estrategias de ahorro y cooperación, como las ROSCA, han mostrado potencial para mejorar la estabilidad financiera. Así, se anticipa que estas estrategias de inclusión financiera generen recomendaciones de política que fortalezcan la resiliencia comunitaria a nivel local e internacional en el futuro.

Inteligencia artificial distribuida basada en LLM

Los LLM son algoritmos de IA entrenados en grandes corpus de texto para predecir, generar y manipular el lenguaje humano. Estos modelos se destacan por su capacidad para entender contextos y producir texto coherente. Un ejemplo destacado es GPT-4, que ha demostrado que puede generar respuestas detalladas y relativamente precisas a partir de un amplio rango de preguntas (OpenAI *et al.*, 2024). El desarrollo continuo de los LLM ha permitido que estos modelos se apliquen en tareas de procesamiento de lenguaje natural (PLN) y en dominios más complejos, que requieren comprensión semántica y generación de textos a partir de instrucciones abstractas (Broekhuizen *et al.*, 2023). Así, estas capacidades los posicionan como herramientas clave para aplicaciones que van desde *chatbots*

hasta la generación automática de contenido. Sin embargo, un reto inherente en el uso de LLM es que pueden generar respuestas basadas en información incorrecta o incompleta, lo que afecta su desempeño en tareas específicas. Esto ha llevado a investigaciones para mejorar la precisión y fiabilidad de estos modelos mediante la integración de módulos extra, como las memorias a corto y largo plazo (Schick *et al*., 2023; Yao *et al*., 2023).

Los agentes autónomos basados en LLM (*augmented autonomous agents*, LAA) son una extensión de los LLM que se emplean para ejecutar tareas de forma autónoma dentro de entornos específicos. Estos agentes no solo generan texto de manera coherente, sino que también toman decisiones basadas en entendimientos previos y observaciones del entorno (Park *et al*., 2023). Para funcionar eficazmente, los LAA requieren de arquitecturas que gestionen de manera efectiva la memoria y las capacidades cognitivas complejas. Estas arquitecturas suelen incluir módulos para la percepción, la planificación, la reflexión y la acción. Por ejemplo, el *framework* Generative Agents proporciona un robusto armazón que permite a los LAA almacenar recuerdos a largo plazo, reflexionar sobre experiencias pasadas y planificar acciones futuras con base en esa información (Park *et al*., 2023). Este enfoque ha sido útil para crear comportamientos humanos convincentes en entornos simulados y ha facilitado la navegación autónoma en videojuegos como Minecraft (Wang *et al*., 2023). A medida que se exploran más áreas de aplicación, se revela que los LAA pueden imitar comportamientos humanos realistas y manejar interacciones complejas; sin embargo, su capacidad para colaborar eficazmente con otros agentes aún está en fase de investigación (Du *et al*., 2023; Zhang *et al*., 2023).

La cooperatividad entre agentes es una habilidad crítica para resolver problemas en entornos complejos. Esta capacidad se deriva de la comprensión mutua y la toma de acciones colaborativas. En el ámbito de los LAA y la DAI, la cooperación se da cuando múltiples agentes colaboran para alcanzar un objetivo común, utilizando el conocimiento de su entorno y las habilidades adquiridas durante su entrenamiento (Gross *et al*., 2023). Para que la cooperación entre LAA sea efectiva, es fundamental que estos agentes puedan comunicarse y coordinarse entre sí. Esta habilidad está influenciada por sus arquitecturas, que deben incluir módulos para la comunicación y el entendimiento de los otros agentes. Por ejemplo, un agente necesita ser capaz de interpretar las intenciones y acciones de otros agentes, así como comunicarse con ellos para lograr objetivos comunes (Dafoe *et al*., 2020a). A pesar de los avances, los estudios muestran que aunque los LAA tienden a cooperar, sus acciones no siempre reflejan una comprensión clara de la colaboración efectiva dentro del entorno; esto subraya la necesidad de arquitecturas más robustas para mejorar su capacidad de colaboración (Agapiou *et al*., 2023).

En el estudio "Can LLM-augmented autonomous agents cooperate? An evaluation of their cooperative capabilities through Melting Pot" (Mosquera *et al*., 2024), desarrollado por investigadores de la Universidad de los Andes, se utilizaron entornos del proyecto Melting Pot para evaluar la capacidad de cooperación de los LAA como avance fundamental de DAI. Estos entornos están diseñados para simular escenarios de dilemas sociales, donde agentes independientes deben tomar decisiones que afectan el bienestar individual y colectivo. El experimento clave utilizó el escenario Commons Harvest, que se desarrolla en una cuadrícula, donde los agentes deben recolectar manzanas. Los agentes que recolectan de manera insostenible pueden agotar los recursos, lo que refleja la "tragedia de los comunes" (Agapiou *et al*., 2023). Los LAA en este entorno recibieron acciones de alto nivel, como "moverse a una posición específica" o "explorar un área", y se evaluó su capacidad para manejarse en un entorno poblado tanto por otros agentes como por *bots* que recolectan de manera insostenible.

Los resultados del experimento destacaron que, aunque los LAA manifestaron una tendencia a la cooperación, tuvieron problemas para entender claramente cómo colaborar de manera efectiva en el entorno dado (Mosquera *et al*., 2024; Park *et al*., 2023). Los agentes presentaron comportamientos que incluyeron la coordinación en algunos aspectos, como evitar la recolección de la última manzana de un árbol para prevenir su agotamiento; no obstante, también se observó que los agentes fallaron en intercambiar información crítica entre sí y establecer estrategias comunes a largo plazo. Estos problemas resaltan las limitaciones actuales de las arquitecturas utilizadas, lo que sugiere que aspectos como la comunicación y la planificación conjunta necesitan mejorarse para alcanzar una cooperación más eficiente.

Los desafíos identificados incluyen la necesidad de mejorar las capacidades de comunicación y entendimiento entre los agentes. Las arquitecturas actuales, aunque efectivas en algunos contextos, demuestran limitaciones significativas cuando se enfrentan a dilemas sociales complejos. Un área prometedora de investigación es el desarrollo de módulos especializados que mejoren la interpretación y el uso de la intención detrás de las acciones de otros agentes (Dafoe *et al*., 2020a). El trabajo futuro podría centrarse en extender las capacidades de memoria y reflexión, para incluir evaluaciones críticas de las acciones pasadas y su impacto en el entorno. Esto implicaría la integración de sistemas de puntuación de reputación y módulos de compromiso que incentiven comportamientos cooperativos a largo plazo (Ni y Buehler, 2024). A pesar de que los LAA muestran potencial para la cooperación, es claro que se necesitan arquitecturas más robustas y especializadas para que estos puedan colaborar de manera eficaz y eficiente en una variedad de entornos complejos.

Conclusiones

Los avances presentados en este documento subrayan el compromiso de la Universidad de los Andes con la investigación de vanguardia en DAI. Los resultados obtenidos hasta ahora no solo han demostrado la relevancia de estas tecnologías en problemas de ingeniería, resiliencia comunitaria y optimización de recursos en entornos complejos, sino que también han abierto nuevas vías para explorar aplicaciones más amplias en diferentes contextos. A medida que la DAI sigue evolucionando, su combinación con modelos avanzados de IA, como los LLM, tiene el potencial de abrir nuevas oportunidades de investigación en problemas relevantes.

La toma de decisiones en entornos multiagente enfrenta retos críticos, entre ellos la necesidad de que los agentes interpreten correctamente las acciones de los demás. En escenarios dinámicos y descentralizados, los agentes deben coordinarse y comunicarse, así como entender las intenciones y respuestas de otros agentes para adaptarse en tiempo real. Esta capacidad es esencial para manejar la incertidumbre y los posibles fallos de comunicación, factores que afectan la estabilidad y el desempeño del sistema. El trabajo futuro debería centrarse en construir sistemas más robustos, donde los agentes autónomos puedan comprender y predecir mejor las acciones de humanos y máquinas. En algunos contextos esto se denomina teoría de la mente (*theory of mind*) (Strachan *et al*., 2024). Esto llevaría a un trabajo colectivo más efectivo en situaciones que requieren de un alto nivel de coordinación, control, adaptabilidad y resiliencia. Además, es necesario mejorar los sistemas actuales, en especial en lo que respecta a la comunicación, el control, el entendimiento, las estructuras sociales y el compromiso entre los agentes para cooperar. Al incorporar capacidades de memoria y toma de decisiones más avanzadas, los sistemas futuros podrían trabajar juntos de manera más efectiva y resistir desafíos de forma resiliente.

A medida que avanzamos, la integración de DAI con otras tecnologías emergentes, como el internet de las cosas y los sistemas ciberfísicos, probablemente dará lugar a nuevas aplicaciones en áreas como las ciudades inteligentes y un nexo para la gestión optimizada de infraestructuras críticas. También hay un esfuerzo importante por reducir limitaciones en cuanto a la colaboración efectiva entre agentes y humanos en entornos reales. Aunque los modelos actuales permiten cierto nivel de interacción, persisten desafíos en la comprensión mutua, la adaptabilidad y la comunicación en escenarios no controlados. La DAI aún tiene un largo camino por recorrer, con muchos desafíos y oportunidades por delante; sin embargo, con investigación y desarrollo continuos, tiene el potencial de transformar no solo la tecnología, sino también las comunidades que dependen de ella.

Agradecimientos

Los autores quieren agradecer el aporte de Google Research para el desarrollo de algunas de las ideas expuestas en este artículo.

Referencias

Agapiou, J. P., Vezhnevets, A., Duéñez-Guzmán, E., Matyas, J., Mao, Y., Sunehag, P., Köster, R., *et al.* (2023). Melting Pot 2.0. *arXiv*. https://arxiv.org/abs/2211.13746

Ananduta, W., Barreiro-Gómez, J., Ocampo-Martínez, C. y Quijano, N. (2018). Mitigation of communication failures in distributed model predictive control strategies. *IET Control Theory & Applications*, *12*(18), 2507-2515. https://doi.org/10.1049/iet-cta.2018.5044.

Annaswamy, A., Khargonekar, P., Spurgeon, S. y Lamnabhi-Lagarrigue, F. (2023). *Cyberphysical-human systems: Fundamentals and applications*. John Wiley & Sons.

Axelrod, R. (1984). *The evolution of cooperation*. Basic Books.

Bacci, G., Lasaulce, S., Saad, W. t Sanguinetti, L. (2016). Game theory for networks: A tutorial on game-theoretic tools for emerging signal processing applications. *IEEE Signal Processing Magazine*, *33*(1), 94-119. https://doi.org/10.1109/MSP.2015.2451994

Barreiro-Gómez, J., Obando, G. y Quijano, N. (2017a). Distributed population dynamics: Optimization and control applications. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *47*(2), 304-314. https://doi.org/10.1109/TSMC.2016.2523934

Barreiro-Gómez, J., Ocampo-Martínez, C. y Quijano, N. (2017b). Dynamical tuning for multiobjective model predictive control based on population games. *ISA Transactions*, *69*(1), 175-186. https://doi.org/10.1016/j.isatra.2017.03.027

Barreiro-Gómez, J., Mas, I., Giribet, J., Moreno, P., Ocampo-Martínez, C., Sánchez-Peña, R. y Quijano, N. (2021). Distributed data-driven UAV-formation control via evolutionary games: Experimental results. *Journal of The Franklin Institute*, *358*(1), 5334-5352. https://doi.org/10.1016/j.jfranklin.2021.05.002.

Bellman, R. (1952). On the theory of dynamic programming. *Proceedings of the National Academy of Sciences*, *38*(8), 716-719.

Bertsekas, D. y Tsitsiklis, J. (1996). *Neuro-dynamic programming*. Athena Scientific.

Bhattacharya, A. (2021). *The man from the future: The visionary life of John von Neumann*. Penguin.

Bouman, F. (1995). Rotating and accumulating savings and credit associations: A development perspective. *World Development*, *23*(3), 371-384.

Broekhuizen, T., Dekker, H., De Faria, P., Firk, S., Nguyen, D. y Sofka, W. (2023). AI for managing open innovation: Opportunities, challenges, and a research agenda. *Journal of Business Research*, *167*, 114196. https://doi.org/10.1016/j.jbusres.2023.114196

Carrasco Martínez, S., Gamboa Montero, J. J., Maroto Gómez, M., Alonso Martín, F. y Salichs, M. Á. (2023). Aplicación de estrategias psicológicas y sociales para incrementar el vínculo en interacción humano-robot. *Revista Iberoamericana de Automática e Informática Industrial*, *20*(2), 199-212. https://doi.org/10.4995/riai.2023.18739

Collins, D. (2009). *Portfolios of the poor: How the world's poor live on $2 a day*. Princeton University Press.

Dafoe, A., Hughes, E., Bachrach, Y., Collins, T., McKee, K., Leibo, J., Larson, K. y Graepel, T. (2020). Open problems in cooperative AI. *arXiv*. https://arxiv.org/abs/2012.08630

Detjen-Leal, D., Quijano, N. y Rodríguez, C. (2023). *Dynamic policy evaluation for ethical decision-making in autonomous vehicles* [ponencia]. IEEE 6th Colombian Conference on Automatic Control (CCAC).

Du, Y., Li, S., Torralba, A., Tenenbaum, J. y Mordatch, I. (2023). Improving factuality and reasoning in language models through multiagent debate. *arXiv*. https://arxiv.org/abs/2305.14325

Gao, B. y Pavel, L. (2021). On passivity, reinforcement learning, and higher order learning in multiagent finite games. *IEEE Transactions on Automatic Control*, *66*(1), 121-136. https://doi.org/10.1109/TAC.2020.2978037

Genis-Mendoza, F., Konstantopoulos, G. y Bauso, D. (2022). Online pricing for demand-side management in a low-voltage resistive micro-grid via a Stackelberg game with incentive strategies. *IET Smart Grid*, *5*(2), 76-89. https://doi.org/10.1049/stg2.12053

Giraldo, J., Quijano, N. y Passino, K. (2015). Honeybee social foraging algorithm for resource allocation. En *Springer handbook of computational intelligence* (pp. 1361-1376). Springer.

Gómez, D., Quijano, N. y Giraldo, L. F. (2022). Information optimization and transferable state abstractions in deep reinforcement learning.

ieee Transactions on Pattern Analysis and Machine Intelligence, *45*(4), 4782-4793.

González Villasanti, H., Giraldo, L. F. y Passino, K. (2018). Feedback control engineering for cooperative community development: Tools for financial management advice for low-income individuals. *ieee Control Systems Magazine*, *38*(3), 87-101.

Grammatico, S., Parise, F., Colombino, M. y Lygeros, J. (2016). Decentralized convergence to nash equilibria in constrained deterministic mean field control. *ieee Transactions on Automatic Control*, *61*(11), 3315-3329. https://doi.org/10.1109/TAC.2015.2513368.

Groot, N., De Schutter, B. y Hellendoorn, H. (2014). Toward system-optimal routing in traffic networks: A reverse Stackelberg game approach. *ieee Transactions on Intelligent Transportation Systems*, *16*(1), 29-40. https://doi.org/10.1109/TITS.2014.2322312.

Gross, J., Méder, Z., De Dreu, C., Angelo Romano, A., Molenmaker, W. y Hoenig, L. (2023). The evolution of universal cooperation. *Science Advances*, *9*(7), eadd8289. https://www.science.org/doi/508abs/10.1126/sciadv.add8289

Han, Z., Niyato, D., Saad, W. y Bacsar, T. (2019). *Game theory for next generation wireless and communication networks: Modeling, analysis, and design*. Cambridge University Press.

Jackson, M. (2008). *Social and economic networks* (vol. 3). Princeton University Press.

Jaleel, H. y Shamma, J. (2020). Distributed optimization for robot networks: From real-time convex optimization to game-theoretic self-organization. *Proceedings of the ieee*, *108*(11), 1953-1967. https://doi.org/10.1109/JPROC.2020.3028295

Kara, S., Martins, N. y Arcak, M. (2022). *Population games with Erlang clocks: Convergence to Nash equilibria for pairwise comparison dynamics* [ponencia]. 2022 ieee Conference on Decision and Control, Cancún, México. https://doi.org/10.1109/CDC51059.2022.9993228

Lu, Z., Cai, F., Liu, J., Yang, J., Zhang, S. y Wu, S. (2022). Evolution of water resource allocation in the river basin between administrators and managers. *Hydrology Research*, *53*(5), 716-732. https://doi.org/10.2166/nh.2022.128

Martínez-Piazuelo, J., Díaz-García, G., Quijano, N. y Giraldo, L. F. (2022a). Discrete-time distributed population dynamics for optimization and control. *ieee Transactions on Systems, Man, and Cybernetics: Systems*, *52*(11), 7112-7122. https://doi.org/10.1109/TSMC.2022.3151042

Martínez-Piazuelo, J., Ocampo-Martínez, C. y Quijano, N. (2022b). Generalized Nash equilibrium seeking in population games under the Brown-von Neumann-Nash dynamics. En *2022 European Control Conference, London, UK* (pp. 2161-2166). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.23919/ECC55457.2022.9838437

Martínez-Piazuelo, J., Quijano, N. y Ocampo-Martínez, C. (2022c). A payoff dynamics model for generalized Nash equilibrium seeking in population games. *Automatica*, *140*(1), 110227. https://doi.org/10.1016/j.automatica.2022.110227

Martínez-Piazuelo, J., Quijano, N. y Ocampo-Martínez, C. (2022d). Nash equilibrium seeking in full-potential population games under capacity and migration constraints. *Automatica*, *141*(1), 110285. https://doi.org/10.1016/j.automatica.2022.110285.

Martínez-Piazuelo, J., Ananduta, W., Ocampo-Martínez, C., Grammatico, S. y Quijano, N. (2023). Population games with replicator dynamics under eventtriggered payoff provider and a demand response application. IEEE *Control Systems Letters*, *7*(1), 3417-3422. https://doi.org/10.1109/LCSYS.2023.3285532

Martins, N. C., Certorio, J. y La, R. J. (2023). Epidemic population games and evolutionary dynamics. *Automatica*, *153*(1), 111016. https://doi.org/10.1016/j.automatica.2023.111016.

Maynard-Smith, J. y Price, G. R. (1973). The logic of animal conflict. *Nature*, *246*(5427), 15-18. https://doi.org/10.1038/246015a0

Minsky, M. (1961). Steps toward artificial intelligence. *Proceedings of the* IRE, *49*(1), 8-30.

Mojica-Nava, E., Macana, C. A. y Quijano, N. (2013). Dynamic population games for optimal dispatch on hierarchical microgrid control. IEEE *Transactions on Systems, Man, and Cybernetics: Systems*, *44*(3), 306-317. https://doi.org/10.1109/TSMCC.2013.2266117

Mosquera, M., Pinzón, J. S., Ríos, M., Fonseca, Y., Giraldo, L. F., Quijano, N. y Manrique, R. (2024). Can LLM-augmented autonomous agents cooperate? An evaluation of their cooperative capabilities through Melting Pot. *arXiv*. https://arxiv.org/abs/2403.11381.

Muros, F. J. (2021). El control coalicional en el marco de la teoría de juegos cooperativos. *Revista Iberoamericana de Automática e Informática Industrial*, *18*(2), 97-112. https://doi.org/10.4995/riai.2020.13456

Ni, B. y Buehler, M. J. (2024). MechAgents: Large language model multi-agent collaborations can solve mechanics problems, generate new data, and

integrate knowledge. *Extreme Mechanics Letters*, *67*, 102131. https://doi.org/10.1016/j.eml.2024.102131

Obando, G. Martínez-Piazuelo, J., Quijano, N. y Ocampo-Martínez, C. (2024). Juegos poblacionales y modelos dinámicos de pago: Un nuevo paradigma para control y optimización. *Revista Iberoamericana de Automática e Informática Industrial*, *21*(4), 287-305.

Obando, G., Poveda, J. I. y Quijano, N. (2016). Replicator dynamics under perturbations and time delays. *Mathematics of Control, Signals, and Systems*, *28*(3), 1-32. https://doi.org/10.1007/s00498-016-0170-9.

Obando, G., Quijano, N. y Ocampo-Martínez, C. (2022). Decentralized control for urban drainage systems using replicator dynamics. *IEEE Access*, *10*(1), 56740-56762. https://doi.org/10.1109/ACCESS.2022.3177631

Ochoa, D. E., Poveda, J. I., Uribe, C. y Quijano, N. (2021). Robust optimization over networks using distributed restarting of accelerated dynamics. *IEEE Control Systems Letters*, *5*(1), 301-306. https://doi.org/10.1109/LCSYS.2020.3001632.

OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., *et al.* (2024). GPT-4 technical report. *arXiv.* https://arxiv.org/abs/2303.08774

Park, S. y Barreiro-Gómez, J. (2023). Payoff mechanism design for coordination in multi-agent task allocation games. *arXiv*. https://arxiv.org/abs/*2306.02278*

Park, S. y Leonard, N. E. (2021). KL divergence regularized learning model for multi-agent decision making. En *2021 American Control Conference, New Orleans, US*. (pp. 4509-4514). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.23919/ACC50511.2021.9483414.

Park, S., Martins, N. C. y Shamma, J. S. (2019). From population games to payoff dynamics models: A passivity-based approach. En *2019 IEEE Conference on Decision and Control* (pp. 6584-6601). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/CDC40024.2019.9029756

Park, S., O'Brien, J., Cai, C., Morris, M., Liang, P. y Bernstein, M. (2023). Generative agents: Interactive simulacra of human behavior. *arXiv*. https://arxiv.org/abs/2304.03442

Pawlick, J. y Zhu, Q. (2021). *Game theory for cyber deception: From theory to applications*. Springer-Verlag.

Quijano, N., Ocampo-Martínez, C., Barreiro-Gómez, J., Obando, G., Pantoja, A. y Mojica-Nava, E. (2017). The role of population games and evolutionary dynamics in distributed control systems. *IEEE*

Control Systems Magazine, *37*(1), 70-97. https://doi.org/10.1109/MCS.2016.2621479.

Quijano, N. y Passino, K. M. (2010). Honeybee social foraging algorithms for resource allocation: Theory and application. *Engineering Applications of Artificial Intelligence*, *23*(6), 845-861.

Rass, S., Schauer, S., Konig, S. y Zhu, Q. (2020). *Cyber-security in critical infrastructures: A game-theoretic approach*. Springer.

Russell, S. J. y Norvig, P. (2016). *Artificial intelligence: A modern approach*. Pearson.

Salas Bahamón, L. M. (2022). Inclusión financiera en Colombia. Evaluación de impacto del programa Grupos de Ahorro y Crédito Comunitario. *Cuadernos de Economía*, *41*(87), 747-782.

Sánchez-Amores, A., Martínez-Piazuelo, J., Maestre, J. M., Ocampo-Martínez, C., Camacho, E. F. y Quijano, N. (2023). Coalitional model predictive control of parabolic-trough solar collector fields with population-dynamics assistance. *Applied Energy*, *334*(1), 120740. https://doi.org/10.1016/j.apenergy.2023.120740

Sandholm, W. H. (2010). *Population games and evolutionary dynamics*. MIT Press.

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N. y Scialom, T. (2023). Toolformer: Language models can teach themselves to use tools. *arXiv*. https://arxiv.org/abs/2302.04761

Strachan, J., Albergo, D., Borghini, G., Pansardi, O., Scaliti, E., Gupta, S., Saxena, K., *et al.*, (2024). Testing theory of mind in large language models and humans. *Nature Human Behavior*, *8*, 1285-1295.

Taylor, P. y Jonker, L. (1978). Evolutionary stable strategies and game dynamics. *Mathematical Biosciences*, *40*(1-2), 145-156. https://doi.org/10.1016/0025-5564(78)90077-9

Thorndike, E. (1898). Animal intelligence: An experimental study of the associative processes in animals. *The Psychological Review: Monograph Supplements*, *2*(4), i-109.

Vlassis, N. (2022). *A concise introduction to multiagent systems and distributed artificial intelligence*. Springer Nature.

von Neumann, J. y Morgenstern, O. (2007). *Theory of games and economic behavior: 60th anniversary commemorative edition*. Princeton University Press.

Wang, G. Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L. y Anandkumar, A. (2023). Voyager: An open-ended embodied agent with large language models. *arXiv*. https://arxiv.org/abs/2305.16291

Weiss, G. (1999). Multiagent systems: A modern approach to distributed artificial intelligence. MIT Press.

Yang, Y. y Wang, J. (2020). An overview of multi-agent reinforcement learning from game theoretical perspective. *arXiv*. https://arxiv.org/abs/2011.00583

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. y Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. *arXiv*. https://arxiv.org/abs/2210.03629

Zambrano, A. F., Giraldo, L. F., Perdomo, M. T., Hernández, I. D. y Godoy, J. M. (2022). Variations of rotating savings and credit associations forcommunity development. *IEEE Transactions on Computational Social Systems*, *10*(2), 614-622.

Zambrano, A. F., Giraldo, L. F., Perdomo, M. T., Hernández, I. D. y Godoy, J. M. (2023). Rotating savings and credit associations: A scoping review. *World Development Sustainability*, *3*, 100081.

Zhang, J., Xu, X. y Deng, S. (2023). Exploring collaboration mechanisms for LLM agents: A social psychology view. *arXiv*. https://arxiv.org/abs/2310.02124

USO DEL APRENDIZAJE POR REFUERZO PARA EL MANEJO DE COMPORTAMIENTO DESCONOCIDO EN SISTEMAS DE *SOFTWARE* DINÁMICOS

Nicolás Cardozo, Ivana Dusparic

Para citar este capítulo:
http://dx.doi.org/10.51573/Andes.9789587988444.9789587988451.9789587988468.02

Introducción

Los sistemas de *software* actuales están en continua interacción con sistemas externos o distintas fuentes de información; esta interacción permite que los sistemas sean más conscientes de su entorno de ejecución y, en respuesta, puedan adaptar su comportamiento para que este sea el más adecuado con respecto a dicho entorno. Estos sistemas, llamados *sistemas de adaptación dinámica* (*self-adaptive systems*, SAS) (Salehie y Tahvildari, 2009), continuamente adaptan su comportamiento de acuerdo con múltiples situaciones detectadas en tiempo de ejecución.

Los SAS han probado ser de utilidad en diversos ambientes, como el manejo de sistemas de transporte (Castagna y Dusparic, 2022; Khan *et al*., 2016), en sistemas robóticos de enjambre (Zhao *et al*., 2018; Zhu *et al*., 2024), en sistemas de comportamiento emergente (Cardozo, 2016) y, en general, en sistemas autonómicos (Cabrera *et al*., 2024; Kephart y Chess, 2003). La construcción de los SAS está compuesta por tres pasos principales: (1) definir el comportamiento base del sistema, (2) determinar las situaciones o condiciones para adaptar el comportamiento y (3) establecer el comportamiento especializado del sistema. Sin embargo, bajo estos parámetros, los SAS tienen una capacidad de adaptación y dinamicidad limitada a los problemas o situaciones que son identificados durante el diseño del sistema, las llamadas *conocidas situaciones desconocidas* (*known-unknowns*) (D'Angelo *et al*., 2019). Por lo tanto, la realización de SAS en situaciones desconocidas requiere afrontar los siguientes retos (Cardozo y Dusparic, 2021, 2022):

1. Las adaptaciones y las situaciones en las que aplican no deben ser prescritas a situaciones conocidas, pues se afecta la dinamicidad y adaptabilidad de los sistemas.
2. El manejo de interacciones y la composición entre adaptaciones no previstas puede causar errores en el comportamiento. Las soluciones actuales a dicho problema necesitan la definición inicial de las reglas de composición (Schmerl *et al*., 2017), lo que de nuevo afecta la dinamicidad y adaptabilidad de los sistemas.

Para abordar estos problemas, proponemos un mecanismo de aprendizaje que permite la generación dinámica de adaptaciones (Cardozo y Dusparic, 2023; Sanabria *et al*., 2024), denominado Auto-COP, y la mejor composición de dichas adaptaciones (Cardozo y Dusparic, 2020), denominado ComInA. De esta forma, el sistema responde a situaciones por completo desconocidas, en la que la estrategia de adaptación no está prescrita, sino más bien es aprendida, lo que ofrece una mayor flexibilidad para que el sistema responda correctamente a todo tipo de situaciones.

Conceptos preliminares

Para la construcción de SAS capaces de adaptar su comportamiento a situaciones desconocidas, utilizamos dos conceptos principales. Primero, la programación orientada al contexto (*context-oriented programming*, COP) se emplea como una herramienta para construir SAS de forma modular y con un alto grado de granularidad de las adaptaciones. Segundo, usamos aprendizaje por refuerzo (*reinforcement learning*, RL) como una estrategia de aprendizaje para la generación y composición de adaptaciones en tiempo de ejecución. El uso de RL permitirá aprender situaciones, condiciones de ejecución y comportamientos inicialmente desconocidos, en tiempo de ejecución. En concreto, utilizamos opciones de RL para aprender comportamientos y la técnica de aprendizaje multiobjetivo *aprendizaje W* para resolver la composición más adecuada con respecto a los objetivos del sistema.

Programación orientada al contexto

La COP (Hirschfeld *et al*., 2008; Salvaneschi *et al*., 2012) es un paradigma de programación para realizar adaptaciones de forma dinámica al comportamiento de los programas, a un alto nivel de granularidad (*e.g.,* métodos). La COP permite una clara independencia entre los módulos de adaptaciones y el

comportamiento base de los programas, así como de otras adaptaciones. Las adaptaciones se incorporan a un sistema durante la ejecución, mediante la recomposición dinámica del sistema; de esta manera, el modelo de composición dinámica utilizado en COP reifica la arquitectura de adaptación de *monitoring, analysis, plan, execute* (MAPE) (Rutten *et al.*, 2017).

Para implementar SAS capaces de adaptar su comportamiento a situaciones desconocidas, en los ejemplos a continuación utilizamos el lenguaje de ContextTraits (González *et al.*, 2013), una extensión de ECMAScript que permite adaptaciones dinámicas. Sin embargo, nótese que los conceptos que desarrollaremos no son específicos de este lenguaje y son aplicables a SAS en general. Hay tres conceptos principales detrás de las adaptaciones dinámicas en COP: *contextos*, *variaciones de comportamiento* y *activaciones de contexto*. Los contextos corresponden a objetos (p. ej., entidades de primera clase del sistema) que representan situaciones del entorno de ejecución capturadas por variables del sistema (p. ej., el estado) o eventos externos monitoreados por sensores. Siempre que se cumplan las condiciones específicas del entorno para los contextos, se dice que estos se encuentran *activos*; de lo contrario, estarán *inactivos*.

Por ejemplo, en el caso del asistente de navegación para un vehículo, el contexto `closeProximity` en el algoritmo 1 define la situación en la que un vehículo se acerca a otro vehículo delante de este.

```
closeProximity = new cop.Context ({
       description: "vehicle in close proximity"
})
```

Algoritmo 1. Definición estática de un contexto en ContextTraits

Cada contexto está asociado a un conjunto de variaciones de comportamiento (p. ej., métodos) que especifican adaptaciones al comportamiento base del sistema. Por ejemplo, en el algoritmo 2 se define el comportamiento especializado para gestionar la proximidad a un vehículo, al adaptar de forma efectiva el comportamiento base drive (p. ej., continuar en línea recta), definido para el vehículo, con nuevas acciones por ejecutar en su lugar —`steerLeft()` y `steerRight()`—, que permitan girar para evitar el vehículo que está delante. El contexto detectado `closeProximity` se asocia con su comportamiento por medio de la abstracción `adapt`, como se muestra en la línea 7 del algoritmo 2.

```
closeProximityBehavior = Trait ({
   drive: function () {
      steerLeft()
      steerRight()
   }
})
closeProximity.adapt(vehicle, closeProximityBehavior)
```

Algoritmo 2. Variación de comportamiento definida para evitar vehículos lentos delante

A medida que los contextos se activan, sus variaciones de comportamiento se asocian con el sistema y se presentan disponibles para la ejecución, es decir, las variaciones serán el comportamiento observable del sistema. Internamente, al activarse un contexto, sus variaciones de comportamiento asociadas se componen con el sistema en ejecución. La desactivación del contexto retira todas las variaciones de comportamiento asociadas con el contexto del sistema en tiempo de ejecución. En ambos casos, el comportamiento del sistema se adapta de forma dinámica. El contexto `closeProximity` se activa y desactiva, como se muestra en el algoritmo 3, basado en la información del sensor de proximidad.

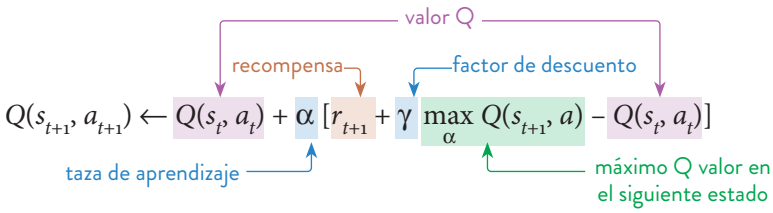
```
if(proximitySensor.receive() < 300)
       closeProximity.activate()
else
       closeProximity.deactivate()
```

Algoritmo 3. Condiciones de activaciones de contextos

Aprendizaje por refuerzo

En el RL, los agentes inteligentes aprenden a mapear situaciones del entorno (estados del entorno) sobre acciones, para maximizar una señal de recompensa numérica que reciben del entorno a largo plazo (Sutton y Barto, 2018). Los agentes de RL están definidos por: *S*, el espacio de estados, formado por todos los estados relevantes del entorno; *A*, el espacio de acciones, es decir, el conjunto de todas las acciones que un agente puede ejecutar y que afectan al entorno; y la recompensa *r*, la señal numérica que codifica el impacto positivo o negativo de la acción en cada paso de ejecución.

El aprendizaje Q (Watkins y Dayan, 1992) es una implementación libre de modelo muy utilizada en RL. La calidad a largo plazo de una acción realizada en un determinado estado se calcula de forma iterativa en una serie de pasos y está representada por un valor $Q(s, a)$. Formalmente, cada paso de ejecución t captura información del entorno y la asigna a un estado $s_t \in S$ de su espacio de estados. A continuación, selecciona una acción $a_t \in A$ de su espacio de acciones y la ejecuta. El agente recibe una recompensa r_t del entorno, cuando pasa al siguiente estado $s_{t+1} \in S$. La recompensa se utiliza para actualizar la optimalidad de realizar la acción a_t en el estado s_t. El objetivo del agente es aprender una política (p. ej., la acción más adecuada para cada estado) que maximice la recompensa del comportamiento a largo plazo. La tasa de aprendizaje α determina qué tanto las nuevas experiencias sobrescriben las experiencias aprendidas con anterioridad, y el factor de descuento γ determina cuánto se descuentan las recompensas futuras para que los agentes prioricen las acciones inmediatas y puedan planificar las mejores acciones a largo plazo. En cada paso de tiempo t, el valor Q de una acción a_t tomada en el estado s_t se actualiza mediante la ecuación de aprendizaje de Bellman, como se observa en la siguiente ecuación.



Opciones en aprendizaje por refuerzo

Las opciones (Sutton *et al.*, 1998) o macroacciones en RL son acciones extendidas temporalmente, que se utilizan para acelerar el aprendizaje, o minimizar los periodos de rendimiento subóptimo durante exploración del entorno, e incorporar acciones a diferentes niveles de granularidad. Las opciones son adecuadas para aprender e integrar secuencias de acciones en sistemas adaptativos basados en COP. Las adaptaciones en COP responden a cambios en el contexto, de forma similar a como se aprenden las acciones en condiciones observadas en el entorno.

Dentro del modelo de ejecución de RL, una opción codifica secuencias de acciones atómicas ejecutadas por el agente en acciones temporalmente extendidas. Las opciones se definen por tres componentes: una política π, que es la

correspondencia entre el espacio de estados *S* y el espacio de acciones *A*; una condición de inicio (p. ej., espacio de estados para comenzar la opción) $I \subseteq S$, y una condición de terminación, que determina la longitud de la opción. Existen numerosas formas de construir opciones a partir de acciones atómicas (Elfwing *et al.*, 2004; Girgin y Polat, 2005; McGovern y Sutton, 1998; Randløv, 1998; Stolle y Precup, 2002). En este trabajo, en particular utilizaremos técnicas estrechamente relacionadas con el cumplimiento de submetas (Stolle y Precup, 2002) y los comportamientos (Girgin y Polat, 2005).

Aprendizaje de múltiples objetivos

El aprendizaje Q utiliza una única fuente de recompensa, esto es, permite optimizar un único objetivo del sistema. Para aprender múltiples objetivos, es posible implementar varios procesos de aprendizaje Q; sin embargo, como un agente solo puede ejecutar una acción a la vez, es necesario añadir un método de arbitraje que resuelva cuál de los procesos de aprendizaje Q, es decir, cuál de los objetivos del agente, toma el control de la ejecución de la acción. El aprendizaje W (Humphrys, 1995, 1996) proporciona ese mecanismo. En cada paso de tiempo, cada proceso de aprendizaje Q propone una acción ejecutable, la más adecuada para el objetivo que representa. Al implementar el aprendizaje W, los agentes también aprenden, en términos de recompensas recibidas para cada uno de los estados, la importancia de la selección de su acción propuesta en contraposición a las acciones propuestas por otros agentes. Dicha importancia se expresa como un valor *W(s)*, aprendido por cada agente de aprendizaje Q para cada estado. El agente con el máximo valor *W* tiene prioridad en el siguiente paso temporal al ejecutar la acción propuesta.

Aprendizaje y generación de adaptaciones

Esta sección presenta el diseño de Auto-COP (Cardozo y Dusparic, 2018, 2022, 2023), nuestra solución para la generación de adaptaciones para SAS. Primero, presentamos la funcionalidad de alto nivel de Auto-COP y, luego, describimos sus dos componentes principales: (1) el aprendizaje de secuencias de acciones, mediante opciones de RL, y (2) la generación de adaptaciones basadas en las opciones aprendidas.

Generación de adaptaciones

En el desarrollo de los SAS, la información de las adaptaciones y los estados en las que deben suceder puede no estar disponible o no ser conocida de antemano por los desarrolladores. Esto limita la adaptabilidad del sistema al comportamiento conocido y especificado. Además, es difícil saber si las adaptaciones predefinidas corresponden realmente al comportamiento más apropiado para el estado actual, ya que es posible que el entorno evolucione durante la ejecución. Para abordar estos problemas, Auto-COP permite la generación dinámica de adaptaciones basadas en ejecuciones previas del sistema y la interacción con el entorno.

El proceso para la generación de adaptaciones en Auto-COP se muestra en la figura 2.1. Las acciones ejecutadas por el sistema generan una traza de ejecución de acciones. El origen de estas acciones puede ser un comportamiento predefinido, la intervención de usuarios (p. ej., acciones atómicas) o acciones generadas por otros procesos (p. ej., adaptaciones). La traza de las acciones ejecutadas se utiliza como entrada para el modelo de aprendizaje de opciones RL, que aprende las secuencias de acciones más adecuadas para las condiciones específicas del entorno. Estas secuencias se utilizan como entrada para el generador de adaptaciones, que las empaqueta en adaptaciones reutilizables, lo que define un contexto a partir del estado del sistema y las variaciones de comportamiento de dicho contexto con base en las opciones aprendidas.

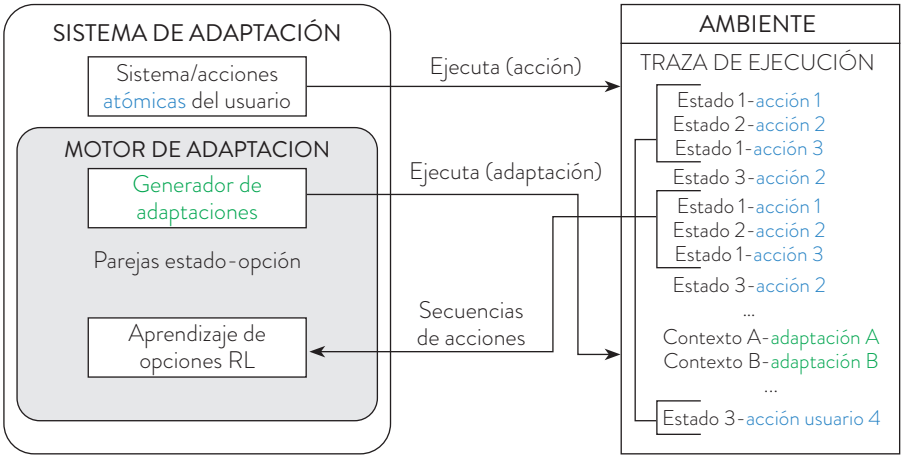


Figura 2.1. Modelo del proceso de aprendizaje de adaptaciones
Fuente: elaboración propia.

Aprendiendo opciones

El aprendizaje de opciones es un proceso que se lleva a cabo a lo largo de la ejecución del sistema. Para aprender opciones, observamos el conjunto de estados *S* del entorno (p. ej., variables monitoreadas), los cuales serán candidatos para convertirse en contextos, y el conjunto de acciones *A* (atómicas u opciones aprendidas), que pueden ejecutarse para cada estado. El módulo `Aprendizaje de opciones RL` aprende un conjunto de opciones *O*, que contiene un mapa de todos los estados $s_i \in S$, asociados a un conjunto de secuencias de acciones, definidas como opciones O_{si}, disponibles para su ejecución en ese estado. Todos los conjuntos de opciones O_{si} están inicialmente vacíos, ya que las opciones se aprenden durante la ejecución. Los conjuntos de opciones O_{si} se procesan en pequeños lotes de tamaño `batchSize`. Usamos el procesamiento por lotes, ya que el procesamiento después de cada paso de ejecución puede ser costoso y tener un efecto insignificante en las opciones generadas. Las líneas 4-15 del algoritmo 4 muestran el pseudocódigo con los detalles del proceso.

```
1 S := {s1, ..., sn} ; A := {a1, ..., am} ; O :
  = {{s1, Ø}, ..., {sn, Ø}} ; lastBatchEnd := 0
2 //EXTRACCION DE OPCIONES
3 while(true){
4  AtomicActionLog:=write(si, ai, r(si, ai))
5  if(timestep t mod batchSize == 0){
6  //Construir opciones de lotes de funciones en la
   traza de ejecucion (Log)
7     for(i=lastBatchEnd; i<logSize; i++){
8        loggedState := readLogLine(i, statePosition)
         //Obtener estado en el log
9        for(j=0; j<maxOptionLength&&!goalReached; j++)
10          actionSequencei += readLogLine(i+j, action
            Position) //Obtener acciones
11    }
12    Oi := {loggedState, actionSequencei} ; O.add(Oi)
13 }
14 lastBatchEnd := logSize
15 }
16 //GENERACION DE ADAPTACIONES
```

```
17 ReinforcementLearning.initialize(S, O)
18 currentState := senseEnvironment()
19 if(contextAdaptationAvailable(currentState)) {
   //seleccion de opciones
20    selectedOption := currentState.pickAnOption(E)
      //epsilon-greedy
21    currentState. activate ()
22    execute(selectedOption) //ejecutar la adaptacion
23    currentState. deactivate ()
24    ReinforcementLearning.updateOption(S, O,
      r(currentState, selectedOption))
25 } else { //no hay opciones disponibles, ejecute la
   accion atomica
26    atomicAction := currentState.pickAction()
27    execute(atomicAction)
28 }
29 newCOPAdaptation := generate(currentState,
   highestQOption)
30 O.add(newCOPAdaptation)
31 }
```

Algoritmo 4. Proceso continuo para la generación de adaptaciones

Durante las primeras fases de ejecución, solo se ejecutan acciones atómicas (predefinidas o ejecutadas por usuarios), ya que el sistema no dispone de adaptaciones generadas a partir de opciones. La ejecución de cada acción se registra en la traza de ejecución del programa (p. ej., el `log`) junto con el estado en las que la acción se ejecutó (a la derecha en la figura 2.1). Para cada acción también registramos la recompensa del efecto de la acción en el estado actual. Esta recompensa, $r(s_i, a_i)$, puede tener múltiples fuentes. Por ejemplo, si las acciones subyacentes se aprenden mediante un sistema basado en RL, la recompensa corresponde a la que se obtiene del entorno. Si las acciones las ejecuta un ser humano, es posible asociar a cada una de ellas una recompensa positiva fija, bajo el supuesto de que tales intervenciones son realizadas por expertos. Por último, puede darse una recompensa constante (p. ej., 1) cada vez que se encuentre un par estado-acción, suponiendo que las acciones que se ejecutan con más frecuencia son las más adecuadas.

El resultado de este proceso es *O*, el mapa de todos los estados identificados en la traza de ejecución, los cuales se consiguen por medio de la función `readLogLine`, que obtiene la información guardada dentro de la traza de ejecución. Cada estado está asociado a un conjunto de secuencias de opciones con tamaño entre 1 (p. ej., una acción atómica) y la longitud máxima de opción *n*; esta puede especificarse externamente, ser calculada de forma experimental, teniendo en cuenta la frecuencia y la utilidad de las secuencias registradas, o fijarse en el número máximo de acciones necesarias para alcanzar un estado específico en el sistema. El algoritmo 5 ilustra un ejemplo genérico de posibles secuencias de acciones generadas para los estados `stateVariablesSet1` y `stateVariablesSet2`.

```
1  state: [stateVariablesSet1]
2  reward:1 -> actions: [“action3”]
3  reward:4 -> actions: [“action3”, “action1”]
4  reward:10 -> actions: [“action3”,
   “action1”,”adaptation1”]
5  reward:10 -> actions: [“action3”, “action1”,
   “adaptation1”, “action4”]

7  state: [stateVariablesSet2]
8  reward:11 -> actions: [“action1”]
9  reward:17 -> actions: [“action1”, “adaptation1”]
```

Algoritmo 5. Secuencias de acciones extraídas como un conjunto ordenado (*O*)

Note que cada estado puede contener múltiples opciones extraídas, y es posible que las secuencias de acciones incluyan opciones generadas previamente (acciones `adaptation` en el fragmento de código). Sin embargo, la mayoría de las opciones extraídas serán inadecuadas para la adaptación, ya que podrían no llevar a un estado correcto. Auto-COP reduce las opciones, para seleccionar una única opción como el comportamiento más adecuado para cada estado (p. ej., contexto), el cual utilizamos para generar una adaptación.

Generación automatizada de adaptaciones

El módulo `Generador de adaptaciones` en Auto-COP toma como entrada las diferentes opciones generadas para cada estado explorado y utiliza un proceso para aprender la opción más adecuada, con el fin de ejecutarla como una

adaptación al comportamiento del sistema. Este proceso de generación se especifica en las líneas 17-30 del algoritmo 4. Para cada opción ejecutada en un estado (p. ej., `currentState`), registramos la recompensa de ejecutar la opción mediante actualizaciones estándar de *Q-learning* (estado, acción, recompensa), con el objetivo de maximizar el rendimiento del sistema a largo plazo. Luego del proceso de exploración, las opciones propuestas como adaptaciones son aquellas con un valor *Q* mayor (p. ej., las mayores recompensas esperadas a largo plazo) para cada estado. El modelo de recompensa de las opciones tiene en cuenta la recompensa del sistema por alcanzar el estado final de la opción. Utilizamos este modelo porque nos interesa que el sistema alcance su objetivo final; solo se consideran más apropiadas las opciones que cierran la brecha entre el estado actual y el estado objetivo. Las opciones con mayor recompensa son las que se utilizan en el módulo `Generador de adaptaciones`, para producir los objetos de contexto y las variaciones del comportamiento

```
1  ContextCurrentState = new cop.Context({name:
   “currentState” })
2  BehavioralVariation = Trait({
3     option:function () {
4        //Secuencia de acciones aprendida
5        action1();
6        ...
7        actionn();
8     }
9  });
10 ContextCurrentState.adapt(BaseSystem, Behavioral
   Variation);
```

Algoritmo 6. Template de la generación de adaptaciones

La definición de contextos generados se muestra en la línea 1 del algoritmo 6. Cada contexto recibe como nombre una cadena correspondiente al estado en el que se debe ejecutar. La generación de variaciones de comportamiento implica la redefinición del comportamiento base del sistema, mediante el uso de la secuencia de acciones de la opción seleccionada, tal y como se muestra en las líneas 2-9 del algoritmo 6. Por último, también generamos la asociación de las variaciones de comportamiento con su respectivo contexto; esto se hace en la línea 10 del algoritmo 6.

Una vez se generan las adaptaciones, mientras el sistema sigue ejecutándose, siempre que se detecte el estado asociado a una de estas (su contexto), el sistema activa el contexto correspondiente `ContextCurrentState.activate()`. Esto llama a la composición de la variación de comportamiento asociada al contexto con el sistema en ejecución. Una vez ejecutada la variación de comportamiento, el contexto se desactiva y el sistema vuelve a su comportamiento base, ejecutando acciones atómicas.

Este proceso se desarrolla de forma continua durante la ejecución del sistema: las trazas se registran, se procesan por lotes en opciones y se explora su idoneidad en interacción con el entorno, para utilizar las opciones más adecuadas como adaptaciones. A medida que el sistema se ejecuta, las opciones generadas también pueden considerarse para la generación de nuevas opciones, lo que compone opciones dentro de opciones. Además, hay que tener en cuenta que, si las condiciones del entorno cambian, la idoneidad de las adaptaciones generadas puede cambiar, y es posible que las nuevas opciones reciban una mejor recompensa que las opciones utilizadas para generar las adaptaciones actuales (p. ej., las adaptaciones obtienen una recompensa negativa, o se ejecutarán acciones atómicas diferentes por los usuarios del sistema). Este proceso permite la generación continua de adaptaciones. De este modo, Auto-COP elimina la necesidad de predefinir las adaptaciones en el momento del diseño y garantiza que el sistema se adapte continuamente a medida que cambian las condiciones, sin cambios manuales en el código fuente.

Aprendiendo estrategias de composición

Una vez generadas las adaptaciones de comportamiento del sistema, nos encontramos con el problema de cómo manejar su combinación. Cada una de las adaptaciones es generada como el comportamiento más apropiado para un estado específico del sistema; sin embargo, en los grandes sistemas, múltiples estados pueden estar presentes en un mismo momento de la ejecución, y el sistema debe responder con la mejor combinación de adaptaciones posibles.

Dado que predefinir todas las posibles combinaciones de adaptaciones para un sistema es imposible, junto con Auto-COP presentamos un novedoso enfoque de composición de adaptaciones, ComInA, que aprende de forma autónoma las interacciones entre adaptaciones, así como las composiciones de adaptaciones más apropiadas para cada combinación de contextos activos. Este proceso se basa en *w-learning* y está formado por: (1) agentes de contexto, los cuales tienen la tarea de realizar una adaptación específica para cada contexto y aprender la idoneidad de otras adaptaciones disponibles en el sistema;

(2) agentes de interacción, cuya tarea es aprender cómo los contextos y las adaptaciones afectan a un contexto determinado (p. ej., la relación entre contextos); y (3) compositor de contextos, que, con base en las aportaciones de los contextos, determina la adaptación o combinación que se debe ejecutar para cada combinación de contextos activos. Ahora, describiremos los algoritmos implementados por cada uno de los agentes.

Diseño de los agentes de contexto

ComInA define un conjunto de agentes de contexto A_{c1}, ..., A_{cn} para cada sistema, implementados mediante un proceso de aprendizaje Q para cada contexto c_i. Inicialmente, cada agente tiene un conjunto de estados Sq_{ci} que indican si su contexto está activo $c_i - 1$ o si esta inactivo $c_i - 0$, y un conjunto de acciones que contiene una única adaptación $A = \{a_{ci}\}$. a_{ci} es la adaptación necesaria para el contexto c_i. La definición de la adaptación, preespecificada o aprendida, es irrelevante, solo requerimos que la adaptación exista para la creación del agente. El proceso de aprendizaje de un agente de contexto se describe en el algoritmo 7. Cada vez que el contexto c_i está activo, el agente A_{ci} propone ejecutar la adaptación a_{ci}. Sin embargo, en función de otros contextos activos y de la decisión del compositor de contextos, podrían ejecutarse otras adaptaciones. En tal caso, el agente amplía su espacio de acción con la adaptación ejecutada (desconocida antes) y aprende (mediante Q-*learning*) su impacto en su sistema. El conjunto de acciones del agente de contexto se construye en tiempo de ejecución. Si los agentes conocen otras adaptaciones en el sistema, estas pueden afectar las preferencias a la hora de ejecutar las propias adaptaciones (p. ej., si se descubre una adaptación mejor).

```
1  Sq_i := {c_i - 0, c_i - 1}
2  A_i := {a_i}
3  QLearning.INITIALIZE(Sq_i, A_i)
4  while(true){ //Ejecucion continua del sistema
5     CurrentContexts[] := senseEnvConditions() //
       evaluar el estado de los contextos en el ambiente
6     if Context C is in CurrentContexts[] {
7        nominateAdaptationToExecute(A_i)
8        currentState:=c_i - 1
9     } else {
```

```
10        currentState:=c_i - 0 // obtiene la recompensa
          para el estado actual
11        reward := QLearning.getReward{currentState)
12     }
13 // si la adaptacion no se has visto anteriormente,
   expanda el conjunto de acciones
14     execAdapt := getExecutedAdaptation()
15     if execAdapt is not in A_i
16        A_i := A_i ∪ execAdaptation
17     QLearning.update(prevState, execAdapt, reward)
       //actualizacion de aprendizaje Q
18 }
```

Algoritmo 7. Definición del agente de contexto y proceso de aprendizaje

Diseño de los agentes de interacción

Los agentes de interacción Aw_{c1}, ... Aw_{cn}, definidos por contexto, aprenden cómo su contexto interactúa con otros agentes. Se implementan utilizando *w-learning*. Elegimos *w-learning* como base de nuestro enfoque por su capacidad de codificar relaciones entre adaptaciones, sin que sus prioridades relativas sean predefinidas o codificadas en el momento del diseño (Cardozo *et al*., 2017). La prioridad relativa inicial de las adaptaciones se expresa mediante recompensas a los agentes de contexto; no obstante, durante la ejecución del sistema, si la adaptación se "descuida" durante un tiempo, su valor *W* asociado acabará siendo superior a otros valores *W*, e incluso las adaptaciones menos prioritarias se harán con el control de la ejecución, permitiendo un cambio dinámico de prioridades. Además, el aprendizaje *W* permite implícitamente la detección de adaptaciones complementarias. Por ejemplo, varias adaptaciones pueden ser adecuadas para múltiples objetivos, es decir, es posible que una adaptación sea adecuada para otro contexto como "efecto secundario" de su ejecución, aunque haya sido designada para adaptarse a otro contexto. En tal caso, el valor *W* de ambos contextos será bajo, ya que ninguno de ellos tiene que competir para ejecutar su adaptación. Este tipo de interacción permite detectar relaciones entre múltiples adaptaciones al aprender la mejor adaptación o combinación de ellas.

El proceso de aprendizaje de un agente de interacción se describe en el algoritmo 8. Al principio, el espacio de estados de A_{wi} es idéntico al espacio de estados del agente de contexto, $S_{wc1} = S_{qc1}$. El espacio de estado de los agentes de interacción se amplía en tiempo de ejecución, a medida que se observan nuevos

contextos. Esta expansión permite a los agentes aprender (cuantitativamente) el impacto de aplicar la adaptación preferida en el rendimiento del sistema, para cada combinación de contextos concreta. Por ejemplo, un agente de interacción para el contexto `c1` podría ampliar su espacio de estado, en tiempo de ejecución, para representar todas las combinaciones activas/inactivas con el contexto `c2` (`statesW=["c1-0,c2-0","c1-0,c2-1","c1-1,c2-0","c1-1,c2-1"]`) y aprender los valores *W* para cada una de las combinaciones.

Se observa que, en sistemas muy grandes, los agentes de interacción no monitorean todos los contextos activos; solo se monitorean los contextos que afectan a los componentes parte de su propio contexto.

```
1  Sw_i := Sq_i
2  WLearning.initialize(Sw_i)
3  while(true){ // ejecucion continua del sistema
4     //todos los contextos activos se convierten a un
       estado
5     CurrentContexts[] := senseEnvConditions()
6     for All (Context c in cCurrentContexts)
7        currentWState += c
8  // si conoce el estado actual, obtiene su
   importancia (w-valor)
9     if (currentWState is in Sw_i) {
10       w := WLearning.getW(currentWState)
11       // nominar la accion de acuerdo al aprendizaje Q
12       WLearning.nominateAdaptation(w, a_i)
13    } else {
14       // Si no conoce el estado actual, expandir el
         conjunto de estados
15 }
16    Sw_i := Sw_i ∪ currentWState // actualizacion de
      aprendizaje W
17    WLearning.update(prevWState, execAdapt, reward)
18 }
```

Algoritmo 8. Definición del agente de interacción y el proceso de aprendizaje

Agentes compositores de contexto

ComInA contiene al menos un agente compositor de contexto[1]. Este módulo se encarga de componer adaptaciones seleccionadas procedentes de los otros agentes. En cada paso de tiempo, un compositor de contexto recibe todos los candidatos a adaptación de los agentes de contexto y su impacto asociado de los agentes de interacción. Así, se ejecuta la adaptación con el valor *W* más alto; sin embargo, como parte de la exploración, un compositor de contexto también ejecuta composiciones de adaptaciones, variando sus combinaciones y orden, para evaluar el impacto de las adaptaciones compuestas en el sistema. Estas adaptaciones compuestas se consideran igual que las adaptaciones individuales y, por tanto, se añaden al espacio de acciones de los compositores de contexto y se aprende su idoneidad para un contexto concreto (p. ej., `adapts=["A1", "A2","A1,A2","A2,A1"]`). Así, con el tiempo, los agentes de contexto individuales pueden sugerir las adaptaciones compuestas más adecuadas para su contexto.

```
1  currentWState
2  while(true){// obtener adaptaciones preferidas de
   los agentes de contexto
3     var adaptNominations[][], finalAdaptation
4     for All (Ac1 to Acn)
5        adaptNominations.push(Ac_i.adaptation, w_i)
6     maxAdapt := findMaxWvalue(adaptNominations)
7     if (exploring) // Intenta combinaciones de
      adaptaciones
8        finalAdaptation = Ac_i.adaptation ∪ Acj.
         adaptation
9     else
10       finalAdaptation := maxAdapt
11    finalAdaptation.execute()
12 }
```

Algoritmo 9. Selección y ejecución de la composición de contextos

1 En sistemas pequeños, un único compositor puede tener una visión de todos los agentes de contexto e interacción, mientras que, en grandes sistemas, múltiples agentes están a cargo de un número limitado de componentes que cooperan como un sistema multiagente, con el fin de garantizar un rendimiento global.

Las adaptaciones se seleccionan con base en las interacciones aprendidas mediante aprendizaje *W*, tanto en el contexto como en los agentes de interacción. Las combinaciones propuestas se ejecutan a continuación, utilizando la estrategia de composición basada en el orden de activación de los contextos de COP; no obstante, ComInA no fija la estrategia de composición, sino que permite definir continuamente estrategias de composición basadas en los valores *W* de los espacios de estado expandidos del agente.

Evaluación del aprendizaje de adaptaciones

Ahora nos disponemos a mostrar la aplicación del proceso automatizado para aprender adaptaciones y su composición dentro de SAS, sin la necesidad de predefinir dichos comportamientos o interacciones. Nótese que hasta el momento esta es la primera solución que permite dicho comportamiento para los SAS.

En el desarrollo de la evaluación, primero nos concentramos en la generación de las adaptaciones (p. ej., los contextos de ejecución y las variaciones de comportamiento). Luego mostramos cómo el sistema puede aprender la composición de las adaptaciones. Para la evaluación, utilizamos un sistema de gestión del transporte urbano.

Aprendiendo adaptaciones

Para evaluar la generación de adaptaciones usamos como aplicación un asistente de navegación para carros autónomos. En particular, evaluamos la utilidad y correctitud de las variaciones de comportamiento generadas en los diferentes estados (p. ej., contextos) en las que estas suceden.

Asistente de navegación

El asistente de navegación en una autopista de dos carriles utiliza cinco acciones de intervención atómicas para controlar los vehículos: `straight`, `steerRight`, `steerLeft`, `speedUp` y `slowDown` (las últimas acciones modifican la velocidad actual en ±10 km/h). El proceso de conducción consiste en una serie de acciones que se repiten con frecuencia en respuesta a las condiciones de la autopista. El comportamiento base del sistema es manejar por el carril derecho de la autopista. Para mantener un comportamiento adecuado en diferentes situaciones, el sistema debe adaptar su comportamiento ejecutando nuevas secuencias de acciones en situaciones particulares. Por ejemplo, la figura 2.2 muestra la situación cuando el vehículo *v1* se encuentra con *v2*, el cual maneja a una velocidad

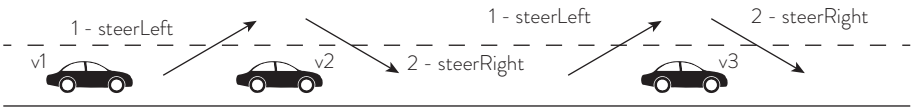


Figura 2.2. Escenario para adelantar en el asistente de navegación
Fuente: elaboración propia.

inferior a la suya; *v1* adelanta a *v2* mediante las acciones `steerLeft` y `steerRight`. Cuando se encuentra con el siguiente vehículo, *v3*, realiza las mismas acciones `steerLeft` y `steerRight`.

El comportamiento esperado del sistema es circular por el carril derecho y usar el carril izquierdo para adelantar, si el vehículo de delante circula demasiado despacio. El vehículo tiene tres objetivos: circular a la velocidad límite, evitar chocar con otros vehículos y no circular por el carril izquierdo. Utilizamos el comportamiento del vehículo con respecto a estos objetivos para medir el rendimiento del sistema. Las métricas de evaluación corresponden al número de veces que el vehículo choca, el número de veces que el vehículo se equivoca de carril y el número de veces que el vehículo supera el límite de velocidad. Las tres métricas deben reducirse al mínimo.

Para la aplicación, desarrollamos un entorno de simulación en JavaScript[2]. Este consiste en una autopista de dos carriles de 500 km. El límite de velocidad de la autopista es de 60 km/h. Otros vehículos aparecen con una probabilidad del 10 %, al menos tres pasos de tiempo después del último vehículo. Los vehículos de tráfico circulan a una velocidad de 30 km/h (para que puedan ser adelantados), siempre en el carril de circulación (el carril derecho del entorno).

Ejecución y resultados

Desarrollamos un algoritmo que aprende el comportamiento de conducción correcto para el vehículo a partir de las acciones atómicas. Observe que Auto-COP es agnóstico con respecto al origen de tales acciones en la traza de ejecución (p. ej., las acciones podrían generarse de las intervenciones humanas o automatizadas mediante RL). Al utilizar RL para generar acciones atómicas, de manera ocasional, el sistema ejecuta una acción errónea (cuando está explorando), lo que nos permite ilustrar cómo la generación de opciones corrige esas acciones, al producir adaptaciones a la ejecución del sistema en lugar de limitarse a repetir

2 Véase https://github.com/FLAGlab/DrivingAssistant

la traza de ejecución. En el proceso (atómico) de aprendizaje de acciones, el espacio de estados del vehículo es: la velocidad actual, que toma valores discretos múltiplos de 10 km/h en el intervalo [0, 70]; el carril actual, modelado como 0 para conducir por el carril derecho y 1 para el carril izquierdo; y la proximidad al vehículo de delante, dividida en el intervalo discreto [1, 4], que describe los pasos de tiempo para alcanzar al vehículo de delante. Una proximidad de 4 indica que no hay ningún vehículo delante, mientras que los valores 1, 2 y 3 denotan que hay un vehículo en proximidad inmediata y que se producirá un choque en 1, 2 o 3 pasos de tiempo en el futuro, si no se toma ninguna medida. El modelo de recompensa penaliza la colisión (−8), la conducción por el carril equivocado (−5), conducir por encima del límite de velocidad (−6), conducir demasiado despacio (−6) y proporciona una recompensa positiva cuando se conduce por el carril correcto sin un vehículo delante (8). La tasa de aprendizaje α se fija a 0,1; el factor de descuento γ a 0,6; y la selección de acciones es ε-*greedy*, con ε empezando en 0,2 en la etapa de exploración y reduciéndose a 0,001 en la etapa de exploración.

Dejamos que el sistema de entrenamiento se ejecute durante 8000 pasos, lo que genera una traza de ejecución de 8000 acciones atómicas. En la traza de ejecución, para cada acción registramos el estado actual del sistema (`speed, lane, vehicle_proximity`), la acción (atómica) ejecutada, el siguiente estado después de ejecutar la acción y la recompensa obtenida al ejecutar dicha acción.

Una vez registradas las trazas de ejecución, ejecutamos la etapa de extracción de opciones (líneas 4-15 del algoritmo 4), para construir las posibles opciones (secuencias de acciones). Con base en la traza de ejecución, en este paso generamos 26 opciones de ejecución en 13 estados diferentes. Solo se selecciona una opción en cada estado para generar la adaptación adecuada. Note que hay 72 estados en total en el entorno (de acuerdo con los valores posibles de las tres dimensiones de estado). Sin embargo, algunos de los estados no tienen ninguna opción asociada, dado que algunos estados no se experimentan durante la exploración (p. ej., no aparecen en la traza de ejecución) y solo nos interesa generar opciones para los estados en los que se requieren adaptaciones (únicamente cuando los objetivos del sistema no se cumplen). Si el vehículo circula al límite de velocidad correcto (60 km/h), en el carril correcto (0) y no hay ningún vehículo delante (4), no es necesaria ninguna adaptación.

Tabla 2.1. Espacio de estados para las opciones aprendidas y sus adaptaciones generadas

Estado	N.°	Secuencia de acciones	Frecuencia
50 0 1	**1**	**{steerLeft(), speedUp(), steerRight()}**	**29**
50 0 1	2	{steerLeft(), speedUp(), steerRight(), steerLeft(), steerRight()}	3
60 0 1	**1**	**{steerLeft(), steerRight()}**	**656**
60 0 1	2	{steerLeft(), steerLeft(), steerRight()}	1
60 0 1	3	{slowDown(), speedUp(), speedUp(), speedUp(), straight(), speedUp(), speedUp(), speedUp()}	1
60 0 1	4	{steerLeft(), straight(), steerRight(), steerLeft(), steerRight()}	1

Fuente: elaboración propia.

Para ilustrar este proceso, nos centramos en el ejemplo del comportamiento de adelantamiento (no predefinido como acción atómica), cuando un vehículo aparece delante del vehículo. La tabla 2.1 muestra todas las opciones extraídas para dos de esos estados. De forma intuitiva, adelantar a un vehículo que circula por delante se realiza mediante una secuencia de acciones atómicas; por lo tanto, la adaptación deseada en el estado [50,0,1] es la opción número 1 de la tabla, ya que adelanta al vehículo de delante a la vez que acelera hasta la velocidad objetivo de 60 km/h, para terminar en el estado objetivo [60,0,4]. Del mismo modo, en el estado [60,0,1] el comportamiento deseado es la opción 1, adelantar a un vehículo sin ninguna acción adicional. De hecho, la frecuencia de ejecución de estas acciones en la traza de ejecución muestra que estas son las opciones ejecutadas con más frecuencia para estos estados; sin embargo, la tabla también muestra opciones adicionales generadas para estos. Estas opciones se ejecutan con menor regularidad durante la exploración. Aunque no podemos asociar estas trazas a una situación exacta durante la ejecución, especulamos que la opción 2 en el estado [50,0,1] y la opción 4 en el estado [60,0,1] se deben a que el vehículo se encontró con un vehículo de tráfico en su carril inmediatamente después de adelantar a un primer vehículo, lo que resulta en tener que repetir la secuencia `{steerLeft(), steerRight()}` una vez más. Para la opción 3 en el estado [60,0,1], especulamos que la primera acción incorrecta en la secuencia `{slowDown()}` resultó en el choque del vehículo, lo que dio lugar

a una velocidad de 0, a partir de la cual el vehículo ejecutó `speedUp()` 6 veces para alcanzar el objetivo [60,0,4]. La generación de adaptaciones debería ser capaz de identificar estas secuencias como inadecuadas y aprender a extraer la opción 1 en una adaptación, para cada uno de los estados.

La lista de opciones extraídas, los estados y sus secuencias de acciones asociadas son la entrada para el proceso de aprendizaje (*adaptation generator*), en el que exploramos las opciones más a fondo a través de RL, para identificar qué opción es la más adecuada para el sistema (líneas 17-30 del algoritmo 4). Para generar adaptaciones, utilizamos el mismo espacio de estados, recompensas y parámetros de aprendizaje que para la generación de las acciones atómicas; no obstante, en este caso, permitimos el uso de todas las opciones disponibles para cada estado, junto con las opciones de aprendizaje y las cinco acciones atómicas. La opción con la mayor recompensa se selecciona para generar una adaptación para el estado dado mediante COP. Evaluamos y presentamos el rendimiento del sistema en la etapa de exploración, al comparar el rendimiento de solo el uso de las acciones atómicas y las combinaciones de acciones atómicas y adaptaciones generadas.

La figura 2.3 muestra la eficacia de las adaptaciones generadas, medida por el rendimiento respecto a los tres objetivos del sistema, al comparar la implementación de acciones atómicas y las adaptaciones generadas. Primero, se compara la cantidad de acciones ejecutadas entre los dos sistemas, durante los 8000 pasos de ejecución; cuantas más acciones atómicas ejecutadas, más adaptaciones utiliza el agente, lo que revela su idoneidad. La figura 2.3a muestra la cantidad de acciones ejecutadas por el vehículo al utilizar las adaptaciones generadas frente a las acciones atómicas. Se ejecutan un total de 1992 adaptaciones para los 8000 pasos de ejecución (p. ej., puntos de decisión), lo que resulta en un total de 15 516 acciones atómicas ejecutadas frente a las 8000 acciones atómicas en la implementación sin adaptaciones (ya que solo es posible ejecutar una acción atómica por paso de ejecución). Por lo tanto, confirmamos que el uso de opciones genera secuencias de acciones adecuadas y que el sistema aprende a utilizarlas como adaptaciones para mejorar su comportamiento, casi con el doble de eficacia de la ejecución del sistema o con la mitad de intervenciones necesarias.

En segundo lugar, comparamos la corrección de usar Auto-COP frente a acciones atómicas. La figura 2.3b muestra el comportamiento resultante de las adaptaciones generadas, en función del número de violaciones a los objetivos incurridas por el vehículo; cuantas menos violaciones, más correcto es el comportamiento. Así, observamos que hay muchas menos infracciones de carril (4) cuando se utilizan adaptaciones que cuando se emplean acciones atómicas (34).

Así mismo, el vehículo presenta un evento de colisión adicional (3 colisiones) en comparación con la ejecución de acciones atómicas (2 colisiones). Esto puede explicarse desde dos perspectivas. En primer lugar, al utilizar adaptaciones, ejecutamos efectivamente casi el doble de pasos que en el caso de las acciones atómicas, por lo que hay más del doble de vehículos en circulación (1497 en lugar de 692). Mientras que la cantidad total de choques aumenta, su porcentaje disminuye en 0,25 %. Esto constituye una mejora con respecto al comportamiento del sistema base. En segundo lugar, tras una inspección más detallada de la traza de ejecución, observamos que dos de los choques se producen durante la ejecución de acciones atómicas y no como consecuencia de la ejecución de las adaptaciones (como se señaló, el comportamiento final del sistema que implementa adaptaciones es una combinación de acciones atómicas y adaptaciones, ya que no todos los estados tienen adaptaciones asociadas). Por tanto, la cantidad de choques se reduce en un 50 % al utilizar las adaptaciones generadas. Por último, observamos que el número de infracciones de los límites de velocidad aumentó con respecto al caso en el que se utilizaron acciones atómicas, que superó el límite de velocidad 8 veces, frente a una infracción del límite de velocidad de las adaptaciones. Podemos concluir que el uso de adaptaciones generadas es beneficioso para el rendimiento del sistema, ya que las infracciones en las tres métricas disminuyen de forma significativa.

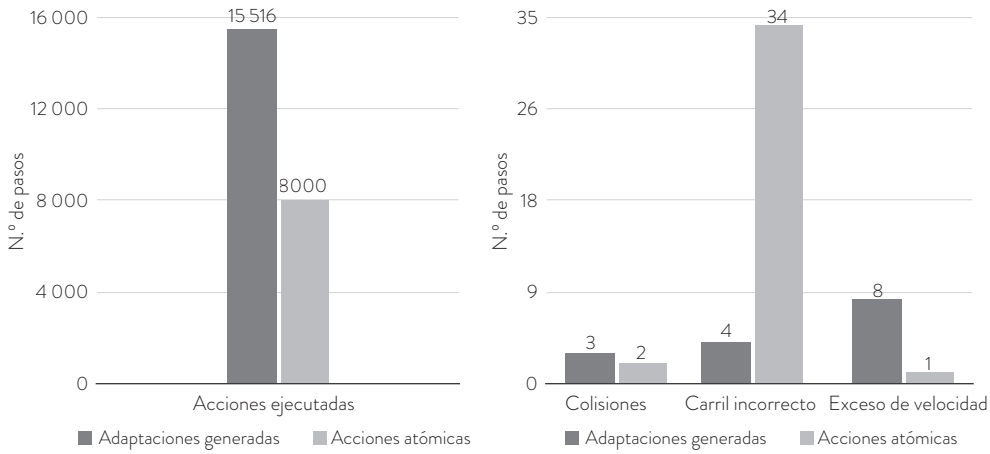


Figura 2.3. Correctitud y utilidad del comportamiento generado por las adaptaciones
Fuente: elaboración propia.

Como un ejemplo, las adaptaciones generadas para el estado [50,0,1] (contexto `BAContext5001`) se muestran en el algoritmo 10, las cuales coinciden correctamente con el comportamiento de adelantamiento deseado.

```
1  Context5001 = new cop.Contexto({ name:
   “Context5001”})
2  BAContext5001 = Trait ({
3     option: function (){
4        this.steerLeft();
5        this.speedUp();
6        this.steerRight();
7     }
8  })
9  Context5001.adapt (agent, BAContext5001)
```

Algoritmo 10. Adaptación generada para el estado `[50,0,1]`

Para integrar las adaptaciones generadas utilizamos COP, que adecua eficazmente el comportamiento de la aplicación: así, pasa de usar acciones atómicas a variaciones de comportamiento. El algoritmo 11 muestra la definición del vehículo base (líneas 1-7), junto con el comportamiento global del asistente de conducción. Para cada paso, de acuerdo con el estado actual, elegimos una acción (líneas 10-13). Si el estado actual no tiene una adaptación asociada, ejecutamos una acción primitiva. Si el estado tiene una adaptación asociada, entonces se ejecuta la variación de comportamiento correspondiente en la función `option()`. Para ejecutar la adaptación, seguimos el proceso de adaptaciones dinámicas utilizadas en COP. En primer lugar, activamos el contexto que representa el estado actual (línea 15). Esto compone la variación de comportamiento asociada al contexto con el sistema. En nuestro ejemplo, para el estado [50,0,1], la variación de comportamiento en el algoritmo 10 se compone con el sistema. El efecto de esta composición es que ahora se tiene una función `option()` definida. Luego, utilizamos la variación de comportamiento para llamar a la opción generada (línea 16). Por último, el sistema pasa a un nuevo estado, como consecuencia de la ejecución de la opción, y el contexto se desactiva (línea 17).

```
class Agent {
   speedUp() { ... }
   slowDown() { ... }
   steerRight() { ... }
   steerLeft() { ... }
   straight() { ... }
   }

while (true) {
   if(qtable[ this .currentState])
      action = qtable[ this .currentState]
   else
   action = this.randomAtomicAction()
   if (action >= Agent.actions.length) {
      eval('Context${state}'.activate ())
      agent.option()
      eval('Context${state}'.deactivate ())
   } else
   eval('agent.${actions[action]}'())
}
```

Algoritmo 11. Integración de las adaptaciones generadas mediante COP

Composición de adaptaciones en un sistema de transporte

La segunda aplicación de la validación TranCity es una aplicación para el control del servicio de transporte de una ciudad. El comportamiento base de TranCity permite supervisar los servicios de autobús de una ciudad, observando la ocupación del sistema (autobuses y estaciones), la frecuencia de autobuses, el número de autobuses que operan en una ruta y el estado de las carreteras (p. ej., si están bloqueadas). La figura 2.4 muestra una vista de TranCity y resalta la ruta utilizada en la evaluación (R4).

Como las condiciones de movilidad en la ciudad están en continuo cambio, el sistema necesita tomar medidas que modifiquen el comportamiento básico de TranCity en función del contexto de ejecución. Por ejemplo, si una estación supera su capacidad, se activa un contexto y se ejecutan las adaptaciones asociadas. La tabla 2.2 presenta los contextos y su variación de comportamiento asociada utilizados en la evaluación.

Durante la ejecución del sistema, varios contextos pueden activarse de forma simultánea. El sistema debe decidir qué adaptación o combinación de adaptaciones es la mejor para maximizar el rendimiento.

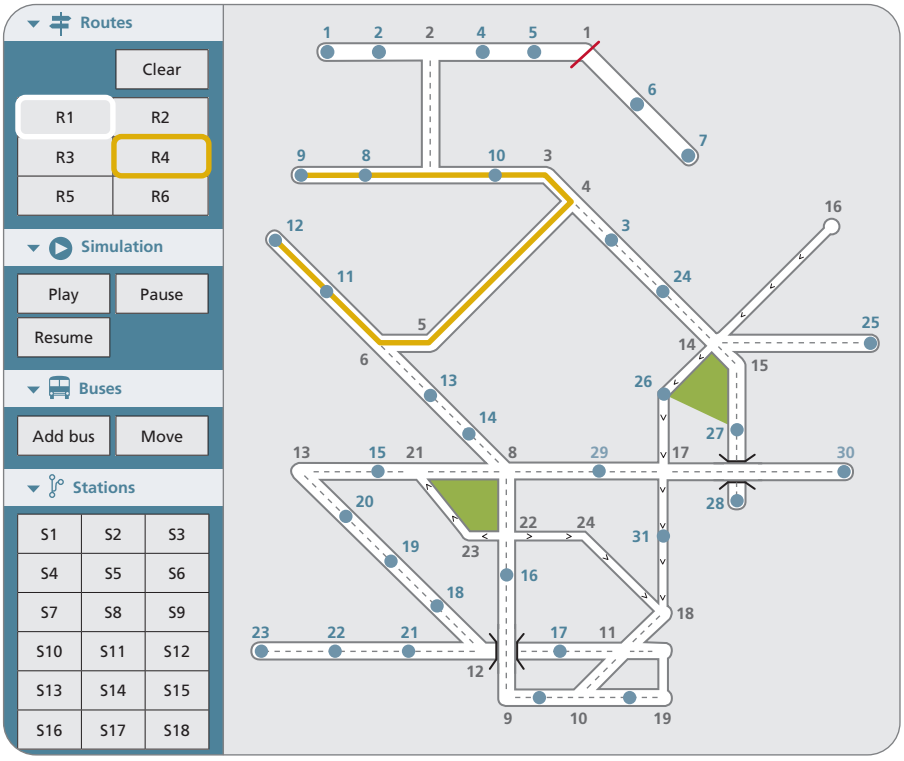


Figura 2.4. Rutas de bus en el sistema de TranCity
Fuente: elaboración propia.

Tabla 2.2. Contextos y sus variaciones de comportamiento asociadas

Contexto	Variación de comportamiento
`FullBus`	Saltar la próxima estación, enviar un bus a la estación llena.
`FullStation`	Cerrar la estación, enviar un bus, redirigir los pasajeros a otras estaciones.
`DepotEmpty`	Retornar un bus al parqueadero.

Fuente: elaboración propia.

Escenarios y parámetros

Evaluamos el comportamiento y el rendimiento de la combinación de adaptaciones en los siguientes escenarios específicos. Cada escenario se ejecutó durante 5000 episodios de aprendizaje, de los cuales dos tercios de los episodios se dedicaron a, por ejemplo, aprender los valores *Q* y *W*, y un tercio a explorar el comportamiento aprendido.

- Escenario 1: adaptaciones independientes/complementarias, en el que los contextos `FullBus` y `FullStation` pueden estar activos de forma simultánea.
- Escenario 2: adaptaciones conflictivas, en el que los contextos `Depot Empty` y `FullStation` pueden estar activos de forma simultánea.

En cada uno de los escenarios, comparamos el rendimiento de tres variaciones, dos alternativas de composición por medio de ComInA y la adaptación predefinida:

- ComInA individual. El sistema decide dinámicamente qué adaptación ejecutar para uno de los contextos activos.
- ComInA componible. El sistema puede ejecutar alguna de las adaptaciones individuales o una composición de las adaptaciones asociadas a ambos contextos.
- Adaptación predefinida. El comportamiento base proporcionado por enfoques existentes. La adaptación que se ejecuta en caso de múltiples contextos activos está predefinida. Nosotros tenemos dos líneas de base, una para cada uno de los dos contextos activos, siendo siempre un "ganador" predefinido.

El rendimiento del sistema se mide a través de dos métricas:

- El número de alertas de contexto, donde menos alertas significa un mejor rendimiento. Las alertas de contexto se producen como resultado de un funcionamiento anormal del sistema. Estas pueden provenir de un aumento/disminución de pasajeros en los autobuses/estaciones; sin embargo, las alertas no resueltas seguirán produciéndose en cada paso de tiempo hasta que se resuelvan. Las estrategias de adaptación efectivas tendrán un menor número de alertas.
- Retraso de los pasajeros, calculado en cada paso temporal como la diferencia entre el retraso acumulado total de todos los pasajeros en el paso temporal t y el retraso que experimentaron en el paso temporal anterior $t - 1$. Idealmente, si el sistema funciona sin problemas, el retraso adicional introducido en cada paso temporal es 0.

La combinación de las implementaciones de ComInA, comparado con las líneas base, permite evaluar si aprender la composición de adaptaciones efectivamente mejora el rendimiento del sistema, contra usar reglas de composición predefinidas. En particular, observamos si ComInA puede detectar relaciones entre adaptaciones activas de forma simultánea y aprender a componerlas o a ejecutar solo una de ellas.

Ejecución y resultados

Para el escenario 1, la figura 2.5 muestra el retraso de los pasajeros y la tabla 2.3 presenta el número de alertas lanzadas.

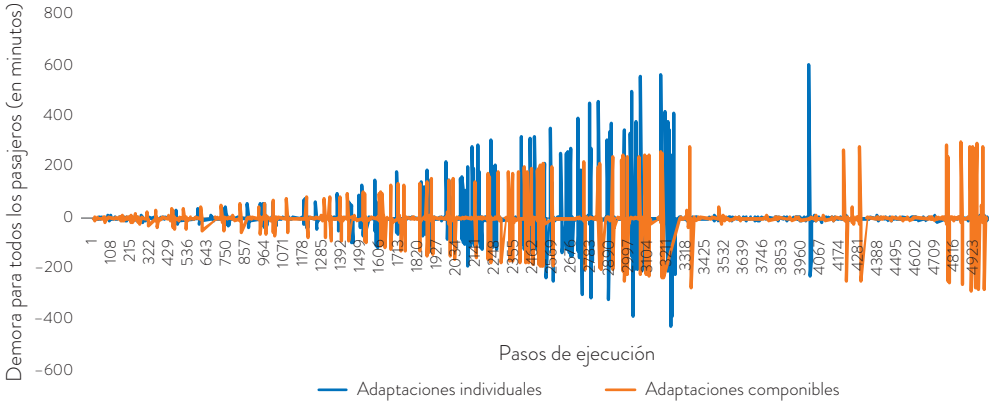


Figura 2.5. Demora por cada paso de tiempo para el escenario 1
Fuente: elaboración propia.

Tabla 2.3. Número de alertas de contexto en el escenario 1

	ComInA individual	ComInA componible	Predefinidas BusStation	
`FullBus`	2170	1766	1236	3315
`FullStation`	303	234	175	477
`Ambos`	173	109	48	320

Fuente: elaboración propia.

Intuitivamente, las adaptaciones para los dos contextos `FullStation` y `FullBus` son compatibles o complementarias, ya que ambas envían un bus adicional desde el parqueadero hasta una estación específica. Además, la adaptación `FullStation` cierra la estación para los nuevos pasajeros, hasta que se libere la capacidad. Observamos un mejor rendimiento general cuando se permite la composición de las dos adaptaciones (en términos de menor retraso máximo y número de alertas) al caso donde el sistema tiene que elegir una sola adaptación para ejecutar (el descenso del retraso alrededor del paso 3700 corresponde a que el sistema pasa a explorar las composiciones aprendidas). De igual manera, se observa que en 5000 pasos el número de veces que los contextos `FullStation` y `FullBus` se activan simultáneamente varía entre 48 y 320, lo que indica la frecuencia de estas situaciones y la necesidad de enfoques dinámicos para resolverlas. También observamos que en la implementación base, en la que solo se ejecuta la adaptación `FullBus`, se produce el menor retraso promedio de los pasajeros (que oscila en los 15 minutos, frente a 300 minutos en el caso compuesto) y el menor número de alertas. Si bien los resultados muestran un mejor rendimiento global, dar siempre prioridad a la adaptación `Bus` podría desequilibrar el sistema. Para detallar mejor el impacto a nivel de pasajero, bus o estación, se debe incluir una métrica de equidad, lo que evita que solo se cumpla una adaptación.

Para el escenario 2, la tabla 2.4 muestra el número de alertas emitidas. Intuitivamente, la relación entre las adaptaciones asociadas a los contextos `FullStation` y `DepotEmpty` son conflictivas, ya que una requiere del envío de un bus a una estación y la otra, retirar el bus del sistema para que sirva de reserva de emergencia en el parqueadero. Por lo tanto, para este escenario no proporcionamos una solución base al ejecutar una adaptación, dado que retirar buses continuamente del sistema eliminaría el servicio completo. En términos de retraso (durante la exploración), ambas implementaciones consiguen una métrica estable; no obstante, en términos de alertas lanzadas, permitir adaptaciones compuestas es más de un 20 % peor en todos los tipos de alertas. Esto indica que las métricas del sistema identifican correctamente que las adaptaciones son conflictivas y no deben componerse, a diferencia del escenario 1, en el que las adaptaciones compuestas mejoraron el rendimiento.

Tabla 2.4. Número de alertas de contexto para el escenario 2

	ComInA individual	ComInA componible
`FullStation`	2380	2761
`DepotEmpty`	713	1055
`Compuestas`	623	981

Fuente: elaboración propia.

Conclusión

Este estudio presenta el beneficio de usar sistemas de aprendizaje aplicados al caso de la ingeniería de *software*, en general, y a los SAS, en particular. Más aún, por medio de técnicas de RL, proponemos una solución a un problema hasta ahora abierto en los SAS: adaptarse a situaciones completamente desconocidas (*unknown-unknowns*).

Para lograr la adaptación del sistema a situaciones desconocidas formulamos una solución que genera adaptaciones y su mejor composición, a partir del conocimiento adquirido por medio de interacciones con el ambiente. El RL se utiliza para detectar los estados en los cuales se requiere una adaptación y las secuencias de acciones asociadas a dichos estados. Las secuencias de acciones se seleccionan utilizando el concepto de opciones o macroacciones, lo que promueve la mejor secuencia de acciones para un estado dado. Las secuencias de acciones y los estados en los que deben tener efecto se generan como adaptaciones de COP, definidas respectivamente como variaciones de comportamiento y contextos. De esta forma, se crean módulos que dan la flexibilidad para componer y descomponer dinámicamente, con el fin de adaptar el comportamiento del sistema sin que este sea prescrito. Una vez generadas las adaptaciones individuales, proponemos un proceso para aprender la mejor forma de componerlas. Este proceso utiliza una aproximación multiobjetivo basada en el aprendizaje W, donde, por medio de la interacción, el sistema aprende cuál es la mejor combinación de adaptaciones para un estado donde múltiples adaptaciones son aplicables.

Nuestros resultados demuestran, a través de dos aplicaciones diferentes, que la solución propuesta es efectiva en la generación y composición de adaptaciones que son útiles para alcanzar el objetivo del sistema. Por último, la validación demuestra que las técnicas de aprendizaje, como el RL, pueden ser aplicadas en el dominio de la ingeniería de *software* para facilitar y mejorar el desarrollo de los SAS.

Referencias

Cabrera, C., Paleyes, A. y Lawrence, N. D. (2024). Self-sustaining software systems (S4): Towards improved interpretability and adaptation. En *Proceedings of the 1st International Workshop on New Trends in Software Architecture* (pp. 5-9). Association for Computing Machinery.

Cardozo, N. (2016). Emergent software services. En *Proceedings of the 2016 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software* (pp. 15-28). Association for Computing Machinery.

Cardozo, N. y Dusparic, I. (2018). *Generating software adaptations using machine learning* [ponencia]. International Workshop on Machine Learning techniques for Programming Languages. Amsterdam, Países Bajos.

Cardozo, N. y Dusparic, I. (2020). Learning runtime compositions of interacting adaptations. En *Proceedings of the International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)* (pp. 108-114). Association for Computing Machinery.

Cardozo, N. y Dusparic, I. (2021). Adaptation to unknown situations as the holy grail of learning-based self-adaptive systems: Research directions. En *International Symposium on Software Engineering for Adaptive and Self-Managing Systems* (pp. 252-253). Institute of Electrical and Electronics Engineers (IEEE).

Cardozo, N. y Dusparic, I. (2022). Next generation context-oriented programming: Embracing Dynamic generation of adaptations. *Journal of Object Technology*, *21*(2), 1-6. http://dx.doi.org/10.5381/jot.2022.21.2.a5

Cardozo, N. y Dusparic, I. (2023). Auto-COP: Adaptation generation in context-oriented programming using reinforcement learning options. *Information and Software Technology*, *164*, 107308.

Cardozo, N., Dusparic, I. y Castro, J. H. (2017). Peace COrP: Learning to solve conflicts between contexts. En *International Workshop on Context-Oriented Programming*. https://doi.org/10.1145/3117802.3117803

Castagna, A. y Dusparic, I. (2022). Multi-agent transfer learning in reinforcement learning-based ride-sharing systems. En *Proceedings of the 14th International Conference on Agents and Artificial Intelligence (ICAART 2022)* (pp. 120-130). Science and Technology Publications.

D'Angelo, M., Gerasimou, S., Ghahremani, S., Grohmann, J., Nunes, I., Pournaras, E. y Tomforde, S. (2019). On learning in collective self-adaptive systems: State of practice and a 3D framework. En *International Symposium on Software Engineering for Adaptive and Self-Managing*

Systems (pp. 13-24). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/SEAMS.2019.00012

Elfwing, S., Uchibe, E., Doya, K. y Christensen, H. I. (2004). Multi-agent reinforcement learning: Using macro actions to learn a mating task. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, *4*, 3164-3169.

Girgin, S. y Polat, F. (2005). Option discovery in reinforcement learning using frequent common subsequences of actions. En *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)* (pp. 371-376). Institute of Electrical and Electronics Engineers (IEEE).

González, S., Mens, K., Colacioiu, M. y Cazzola, W. (2013). Context traits: Dynamic behaviour adaptation through run-time trait recomposition. En *Proceedings of International Conference on Aspect-Oriented Software Development* (pp. 209-220). Association for Computing Machinery. https://doi.org/10.1145/2451436.2451461

Hirschfeld, R., Costanza, P. y Nierstrasz, O. (2008). Context-oriented programming. *Journal of Object Technology*, *7*(3), 125-151. http://www.jot.fm/issues/issue_ 2008_03/article4/

Humphrys, M. (1995). *W-learning: Competition among selfish Q-learners* (inf. téc. UCAM- CL-TR-362). University of Cambridge, Computer Laboratory. https://doi.org/10.48456/tr-362

Humphrys, M. (1996). Action selection methods using reinforcement learning. En *Proceedings of the International Conference on Simulation of Adaptive Behavior* (pp. 135-144). Association for Computing Machinery.

Kephart, J. O. y Chess, D. M. (2003). The vision of autonomic computing. *Computer*, *36*(1), 41-50.

Khan, N. A., Brujic-Okretic, V. y Khaddaj, S. (2016). Self-adaptive service driven architecture for intelligent transport system. En *IEEE Intl Conference on Computational Science and Engineering and IEEE Intl Conference on Embedded and Ubiquitous Computing and Intl Symposium on Distributed Computing and Applications for Business Engineering* (pp. 669-672). https://doi.org/10.1109/CSE-EUC-DCABES.2016.258

McGovern, A. y Sutton, R. S. (1998). *Macro-actions in reinforcement learning: An empirical analysis*. University of Massachusetts.

Randløv, J. (1998). Learning macro-actions in reinforcement learning. En *Proceedings of the International Conference on Neural Information Processing Systems* (pp. 1045-1051). Association for Computing Machinery. http://dl.acm.org/citation.cfm?id=3009055.3009201

Sutton, R. y Barto, G. (2018). *Reinforcement learning, an introduction* (2.ª ed.). Massachusetts Institute of Technology Press.

Rutten, E., Marchand, N. y Simon, D. (2017). Feedback control as MAPE-K loop in autonomic computing. En R. de Lemos, D. Garlan, C. Ghezzi y H. Giese (Eds.), *Software Engineering for Self-Adaptive Systems III. Assurances: International Seminar, Dagstuhl Castle, Germany, December 15-19, 2013, Revised Selected and Invited Papers* (pp. 349-373). Springer International Publishing.

Salehie, M. y Tahvildari, L. (2009). Self-adaptive software: Landscape and research challenges. *ACM Transactions on Autonomous and Adaptive Systems*, *4*(2), 1-42. https://doi.org/10.1145/1516533.1516538

Salvaneschi, G., Ghezzi, C. y Pradella, M. (2012). Context-oriented programming: A software engineering perspective. *Journal of Systems and Software*, *85*(8), 1801-1817. https://doi.org/10.1016/j.jss.2012.03.024

Sanabria, M., Dusparic, I. y Cardozo, N. (2024). Learning recovery strategies for dynamic self-healing in reactive systems. En *SEAMS 24: Proceedings of the 19th International Symposium on Software Engineering for Adaptive and Self-Managing Systems* (pp. 133-142). Association for Computing Machinery. https://doi.org/10.1145/3643915.3644097

Schmerl, B., Andersson, J., Vogel, T., Cohen, M. B., Rubira, C. M. F., Brun, Y., Gorla, A., Zambonelli, F. y Baresi, L. (2017). Challenges in composing and decomposing assurances for self-adaptive systems. En R. de Lemos, D. Garlan, C. Ghezzi y H. Giese (Eds.), *Software engineering for self-adaptive systems III. Assurances* (pp. 64-89). Springer International Publishing.

Stolle, M. y Precup, D. (2002). Learning options in reinforcement learning. En *Proceedings of the International Symposium on Abstraction, Reformulation and Approximation* (pp. 212-223). Springer International Publishing.

Sutton, R. S., Precup, D. y Singh, S. P. (1998). Intra-option learning about temporally abstract actions. *International Conference on Machine Learning*, *98*, 556-564.

Watkins, C. J. C. H. y Dayan, P. (1992). Technical note: Q-learning. *Machine Learning*, *8*(3), 279-292.

Zhao, H., Liu, H., Leung, Y.-W. y Chu, X. (2018). Self-adaptive collective motion of swarm robots. *IEEE Transactions on Automation Science and Engineering*, *15*(4), 1533-1545.

Zhu, W., Oguz, S., Heinrich, M. K., Allwright, M., Wahby, M., Christensen, A. L., Garone, E. y Dorigo, M. (2024). Self-organizing nervous systems for robot swarms. *Science Robotics*, *9*(96).