Exploring the possibilities of Multi-Agent Reinforcement Learning to solve coordinated cooperative tasks in Flexible manufacturing systems

Manuel Ezequías Vázquez[12], Carolina Saavedra Sueldo[12], Luis O. Ávila[3], Gerardo G. Acosta[12], and Mariano De Paula[12]

[1] Centro de Investigaciones en Física e Ingeniería del Centro, UNCPBA-CICpBA-CONICET, B7400JWI, Olavarría, Buenos Aires, Argentina
{manuel.vazquez,ggacosta,mariano.depaula,carolina.saavedra}@fio.unicen.edu.ar
[2] Universidad Nacional del Centro de la Provincia de Buenos Aires (UNCPBA), Facultad de Ingeniería, INTELYMEC, B7400JWI, Olavarría, Buenos Aires, Argentina
[3] Laboratorio de Sistemas Inteligentes, CONICET-UNSL, D5730EKQ, San Luis, Argentina
loavila@unsl.edu.ar

Abstract. Advances in artificial intelligence and Multi-Agent Systems enable coordinated agents to achieve multiple, often conflicting, objectives—making them ideal for "flexible factories." These factories, driven by technologies merging physical, digital, and biological domains, are evolving into "smart factories." Modeling production processes as multi-agent systems allows simultaneous optimization of efficiency, waste reduction, sustainability (economic, social, and environmental), cost savings, and downtime reduction. However, the flexibility needed in reconfigurable environments increases the complexity of decentralized control. Small and medium-sized enterprises (SMEs) are a key example, as they often produce small batches or customized goods, requiring constant adaptation. Multi-agent reinforcement learning provides a viable solution, avoiding impractical centralized control in dynamic settings. This work explores multi-agent reinforcement learning for collaborative manufacturing tasks, such as material handling (a non-value-adding operation where efficiency is critical). A preliminary case study is presented, using virtual environments to train multiple agents in coordinated material manipulation across varying complexity scenarios.

Keywords: Process optimization, Intelligent control, Collaborative manufacturing, Smart factories

Explorando las posibilidades del Aprendizaje por Refuerzo Multiagente para resolver tareas cooperativas coordinadas en Sistemas de fabricación flexibles

Resumen Los avances en inteligencia artificial y Sistemas Multi-Agente permiten coordinar agentes para cumplir múltiples objetivos, incluso contrapuestos, aplicables en "fábricas flexibles". Estas, impulsadas por tecnologías que integran lo físico, digital y biológico, evolucionan hacia "fábricas inteligentes". Modelar un proceso productivo como un sistema multi-agente permite optimizar simultáneamente la eficiencia, reducción de desperdicios, sustentabilidad (económica, social y ambiental), ahorro de costos y reducción de tiempos de inactividad. Sin embargo, la flexibilidad requerida en entornos reconfigurables incrementa la complejidad del control descentralizado. Las pequeñas y medianas empresas (PyMEs) son un caso emblemático, ya que suelen producir lotes pequeños o bienes personalizados, lo que exige una adaptación constante. El aprendizaje por refuerzo multi-agente surge como una solución viable, evitando esquemas centralizados poco prácticos ante entornos cambiantes. Este trabajo analiza dicho enfoque para tareas colaborativas en manufactura, como la manipulación de materiales (una operación sin valor agregado donde la eficiencia es clave). Se presenta un caso de estudio preliminar que utiliza entornos virtuales para entrenar múltiples agentes en tareas de manipulación coordinada en escenarios de diversa complejidad.

Palabras clave: Optimización de procesos, Control inteligente, Manufactura colaborativa, Fábricas inteligentes

1 Introducción

La creciente demanda de productos personalizados y ciclos de vida más cortos impulsa la transición hacia sistemas de producción flexibles, un desafío crítico para las PyMEs, que deben adaptar rápidamente sus procesos manteniendo productividad y sustentabilidad (Chen. et al., 2018). Un reto clave es la gestión coordinada de recursos robóticos (manipuladores, móviles, drones) en tareas como el transporte de materiales, operaciones esenciales para la eficiencia global, aunque no agreguen valor directo (Saavedra Sueldo et al., 2024).

Los Sistemas Multi-Agente (SMA) y el Aprendizaje por Refuerzo Multiagente (MARL, por sus siglas en inglés) ofrecen una solución descentralizada, permitiendo que agentes autónomos colaboren dinámicamente en entornos cambiantes (Albrecht et al., 2024). Este trabajo propone una contribución mediante el diseño y evaluación de un escenario de manufactura colaborativa en Unity, donde múltiples agentes coordinan la manipulación conjunta de objetos en entornos con distintos niveles de complejidad. Nuestra propuesta se enfoca en analizar la emergencia de comportamientos colaborativos en tareas físicas compartidas, fundamentales para procesos adaptativos y flexibles en PyMEs.

2 Estado del arte

Los SMA y el Aprendizaje por Refuerzo (RL, por sus siglas en inglés) han ganado relevancia en entornos industriales, ofreciendo soluciones autónomas y adaptativas para mejorar la eficiencia productiva. En particular, el RL ha demostrado ser efectivo en la optimización de procesos en manufactura flexible. Por ejemplo, Li et al. (2022) emplea RL para planificación automática en líneas de ensamblaje mediante gemelos digitales, logrando mayor precisión en operaciones sin colisiones.

Otros enfoques combinan RL con sistemas de control tradicionales para adaptarse rápidamente a cambios en la producción (Schwung et al., 2018). Estudios recientes indican que el 60% de las aplicaciones de RL en la industria son experimentales, mientras que el resto se distribuye en celdas de producción y simulaciones (Velastegui et al., 2023), reflejando su potencial en manufactura avanzada.

Desde el control inteligente, los modelos holónicos y multiagente han mejorado la autonomía y cooperación en sistemas dinámicos (Boggino, 2005; Quintero Henao, 2009). Además, la integración de Inteligencia artificial (IA) en Manufacturing Execution Systems (MES) permite optimizar procesos y reducir la intervención humana (Durão et al., 2022; Mantravadi et al., 2019).

Para entrenamiento y evaluación, plataformas como Unity ML-Agents (Juliani et al., 2018) destacan por su flexibilidad, gráficos realistas y soporte para algoritmos como PPO (Proximal Policy Optimization) y SAC (Soft Actor-Critic) (Ilosvay & Iaccarino, 2024). Sin embargo, persisten desafíos en estrategias colaborativas para manufactura flexible.

Si bien los estudios mencionados han demostrado la aplicabilidad del RL en entornos industriales y simulaciones avanzadas, aún existen desafíos en la implementación de estrategias colaborativas en manufactura flexible. En este trabajo, proponemos un enfoque basado en múltiples agentes que trabajan de manera cooperativa para completar tareas en un entorno simulado utilizando Unity ML-Agents. A diferencia de estudios previos, nuestra investigación se centra en la interacción entre agentes y en la capacidad de adaptación a escenarios dinámicos, buscando evaluar la emergencia de comportamientos colaborativos en entornos de manufactura flexible.

3 Antecedentes metodológicos

Esta sección presenta los fundamentos teóricos del trabajo, organizados en tres ejes: 1) los SMA y su aplicación industrial mediante el paradigma holónico; 2) el marco del RL y su extensión MARL, incluyendo modelos como los MDPs (Markov Decision Processes,) y los Dec-POMDPs (Decentralized Partially Observable Markov Decision Processes,); y 3) el rol de la simulación en MARL, comparando plataformas como Unity y Gazebo para sistemas robóticos colaborativos.

3.1 Sistemas Multiagente

Los SMA son un paradigma de la IA distribuida donde múltiples agentes autónomos interactúan para resolver problemas complejos. En

manufactura flexible, los SMA permiten gestionar sistemas descentralizados con adaptabilidad, escalabilidad y tolerancia a fallos.

Un agente es una entidad computacional que percibe su entorno mediante sensores, procesa información y actúa mediante actuadores para cumplir tareas específicas. Según Russell & Norvig (2010), los agentes en SMA se caracterizan por cuatro propiedades clave: autonomía (operan sin intervención humana), reactividad (responden a cambios en tiempo real), proactividad (toman iniciativas para alcanzar objetivos) y habilidad social (comunicación y cooperación con otros agentes). En manufactura, los agentes pueden representar recursos físicos (robots, máquinas) o lógicos (sistemas de planificación), modelando fábricas como redes colaborativas.

El paradigma holónico (Van Brussel et al., 1998) integra los SMA con la estructura jerárquica de los sistemas de producción. Cada holón actúa como un agente que combina una entidad física (como un robot) con su correspondiente control lógico, organizándose en jerarquías recursivas que van desde el nivel de máquina hasta el de fábrica completa. La toma de decisiones se realiza de forma distribuida, donde cada holón negocia localmente con otros para optimizar objetivos globales como la minimización del tiempo de producción.

Matemáticamente, este sistema se representa mediante un grafo G=(V,E), donde V corresponde al conjunto de holones (agentes) y E define las relaciones de cooperación entre ellos, incluyendo flujos de materiales y canales de comunicación.

La coordinación se basa en negociación y teoría de juegos. Un protocolo común es el Contract Net Protocol (Smith, 1980), donde un agente manager anuncia una tarea, los contratistas envían ofertas (costo, tiempo) y el manager asigna la tarea al mejor candidato. En problemas colaborativos (ej: transporte coordinado), el equilibrio de Nash asegura que ningún agente se desvíe de la estrategia óptima colectiva (Shoham & Leyton-Brown, 2008). Por ejemplo, dos robots transportando una carga convergen a políticas de movimiento complementarias.

3.2 Aprendizaje por Refuerzo y MARL

El RL es un enfoque bioinspirado del aprendizaje automático donde un agente aprende a tomar decisiones óptimas mediante la interacción con un entorno, recibiendo recompensas o penalizaciones. En manufactura flexible, el RL resulta útil para optimizar procesos dinámicos en entornos cambiantes.

Un problema de RL se formaliza como un Proceso de Decisión de Markov (MDP), utilizado para modelar decisiones secuenciales en entornos estocásticos (Sutton & Barto, 2018). Un MDP se define como $\langle S, A, P(s'|s,a), R(s,a), \gamma \rangle$, donde S y A representan los espacios de estados y acciones; $P(s'|s,a)$ es la probabilidad de transición; $R(s,a)$ la recompensa inmediata; y $\gamma \in [0,1]$ el factor de descuento. El objetivo es hallar una política óptima π^* que maximice la recompensa acumulada descontada:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s\right] \tag{1}$$

La ecuación de Bellman permite calcular la función de valor de forma recursiva:

$$V^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s'} P(s'|s, \pi(s)) V^{\pi}(s') \tag{2}$$

Extensión a Entornos Multiagente. En contextos con múltiples agentes, un MDP se extiende a un Proceso de Decisión de Markov Parcialmente Observable Descentralizado (Dec-POMDP), definido como $\langle I, S, \{A_i\}, P(s'|s, \bar{a}), \{O_i\}, \{R_i\}, \gamma \rangle$. Aquí, I es el conjunto de agentes; S el espacio de estados global; $\{A_i\}$ las acciones individuales; $P(s'|s, \bar{a})$ la dinámica del sistema ante acciones conjuntas $\bar{a}$; $\{O_i\}$ las observaciones parciales; y $\{R_i\}$ las recompensas por agente. Cada agente i observa parcialmente el estado global y actúa según su política $\pi_i : O_i \to A_i$ (Oliehoek & Amato, 2016). En problemas cooperativos, donde $R_i = R$ para todo i, se busca maximizar la recompensa acumulada conjunta.

El desarrollo de algoritmos para RL Multiagente incluye enfoques como métodos basados en valores, actor-crítico y gradiente de política.

Los métodos basados en valores, como Q-Learning y sus variantes multiagente, estiman directamente la función de valor. QMIX (Rashid et al., 2018) introduce redes neuronales para combinar funciones de valor individuales de forma monótona. MADDPG (Multi-Agent Deep Deterministic Policy Gradients) representa el enfoque actor-crítico, empleando críticos centralizados durante el entrenamiento con acceso a las políticas de todos los agentes. PPO (Proximal Policy Optimization) (Schulman et al., 2017), basado en gradiente de política, destaca por su equilibrio entre estabilidad y eficiencia. Finalmente, COMA (Counterfactual Multi-Agent) (Foerster et al., 2018) aborda la asignación de crédito mediante un mecanismo contrafactual, facilitando la evaluación de contribuciones individuales en recompensas colectivas.

3.3 Entornos computacionales y MARL

Los entornos de simulación han sido clave en el desarrollo del MARL, al permitir la experimentación en escenarios complejos con múltiples agentes en entornos realistas, evitando los altos costos y tiempos que implicaría el uso de sistemas físicos. En contextos de manufactura flexible, estos entornos habilitan la evaluación de estrategias colaborativas para resolver tareas ad-hoc, aportando mayor adaptabilidad a los sistemas productivos.

Durante la última década surgieron diversas plataformas que se convirtieron en referencia para investigación en MARL. Unity ML-Agents destaca por su equilibrio entre realismo visual y simulación física, facilitando la creación de entornos 3D personalizados (Juliani et al., 2018). Gazebo, en combinación con ROS (ROS), resulta más afín a la robótica tradicional, con modelado preciso de sensores y actuadores (Koenig & Howard, 2004). Otras opciones como PyBullet y NVIDIA Isaac Sim ofrecen simulación acelerada por GPU (Graphics Processing Unit), lo que permite trabajar con grandes cantidades de agentes (Coumans & Bai, 2016).

Estas plataformas ya demostraron su utilidad en manufactura inteligente. En ensamblaje colaborativo, permiten probar estrategias de coordinación antes de

su implementación, reduciendo tiempos de configuración en hasta un 60% (Wang et al., 2021). En logística interna, la simulación de flotas de AGVs (Automated Guided Vehicles) con MARL mejoró la eficiencia entre un 30 y 40% frente a métodos tradicionales (Chen et al., 2022).

Estos entornos permiten modelar dinámicas físicas con alta precisión, incluyendo cuerpos rígidos, fricción y restricciones articulares. Esto es crucial para simular tareas como el transporte cooperativo de cargas, donde errores mínimos en la coordinación pueden causar inestabilidad (Zhang et al., 2021). Además, la técnica de aleatorización de dominios (domain randomization) mejora la transferencia simulación-realidad. Al variar parámetros como masas o fricciones durante el entrenamiento, se obtienen políticas más robustas que toleran diferencias de más del 20% entre entornos, sin pérdida de desempeño (Curşeu et al., 2020).

La simulación facilita la evaluación rápida de distintas configuraciones productivas, rutas de transporte o estrategias de colaboración, reduciendo tiempos y costos asociados al rediseño. Según estudios recientes, puede acortar en hasta un 45% la implementación de nuevos layouts (Zhang et al., 2021). Integraciones con gemelos digitales ya permiten transferir directamente las políticas aprendidas a entornos físicos, cerrando el ciclo entre simulación y operación. Este enfoque, basado en actualizaciones constantes a partir de datos reales, está habilitando el desarrollo de fábricas adaptativas y autoorganizadas (Saavedra Sueldo et al., 2023).

4 Metodología

En esta sección se presenta la metodología empleada para abordar el caso de estudio propuesto. En primer lugar, se define la problemática a resolver, vinculada a la manipulación colaborativa de materiales en espacios reducidos. Luego, se describe el desarrollo del entorno de simulación, la arquitectura del sistema, el diseño de los agentes, los mecanismos de interacción y el algoritmo utilizado. Finalmente, se detalla cómo los robots interactúan y aprenden a coordinarse para manipular y transportar una pieza dentro de un entorno físico restringido, representativo de condiciones reales en entornos de manufactura flexible.

4.1 Definición del problema

Tal como se menciona en la introducción, uno de los principales desafíos en entornos productivos modernos es la gestión coordinada de recursos físicos que deben colaborar para ejecutar tareas logísticas tales como la manipulación y el transporte de materiales (Saavedra Sueldo et al., 2024). Estas tareas, aunque no agregan valor directo al producto final, impactan de forma directa sobre el rendimiento global del sistema, afectando la eficiencia del flujo de producción, el uso del espacio físico y la seguridad operativa.

En este contexto, la irrupción de robots móviles con patas, tanto bípedos como cuadrúpedos, equipados con mecanismos de manipulación (como grippers) representa una oportunidad para transformar los entornos productivos actuales. Estas tecnologías permiten concebir layouts dinámicos, en los que las estaciones de trabajo puedan reconfigurarse de manera continua en función de

requerimientos variables, como cambios en la demanda o personalización de productos. Este nivel de flexibilidad exige el desarrollo de nuevas estrategias de colaboración y coordinación entre agentes autónomos capaces de levantar, mover, rotar y posicionar materiales en escenarios dinámicos y no estructurados.

4.2 Entorno simulado

Se diseñó un entorno tridimensional de $20 \times 20 \times 10$ unidades de longitud (ul) que representa una habitación industrial, donde dos agentes deben cooperar para transportar una pieza rígida a través de una abertura estrecha. Esta situación simula tareas reales en espacios reducidos, como pasillos o zonas de transferencia.

Los agentes, modelados como cápsulas móviles de $1 \times 1 \times 0.8$ ul, deben trasladar una tabla de 15×1 ul y 0.1 unidades de masa (um). La abertura tiene 10×10 ul, por lo que es necesario coordinar movimientos y orientación para evitar colisiones o caídas.

4.3 Arquitectura del sistema

El sistema se estructura de forma modular en dos componentes principales: `GameController`, que gestiona el estado general de la simulación, y la clase `Robot`, que implementa la lógica de cada agente. La Figura 1 muestra su diseño en UML (Unified Modeling Language).

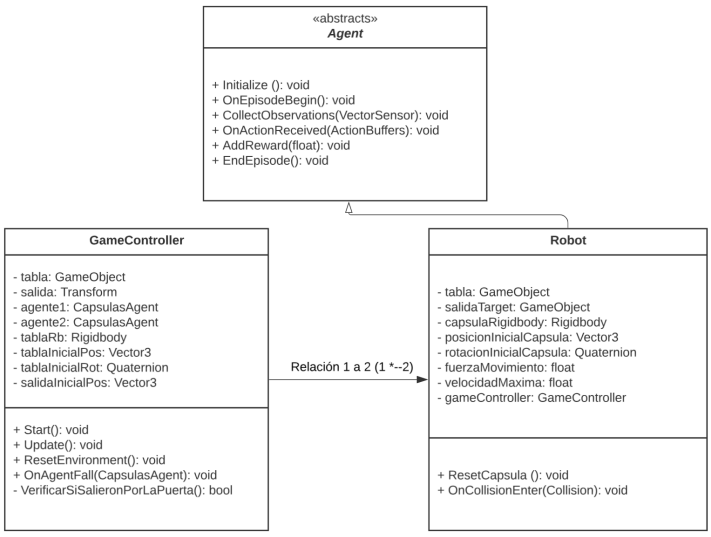


Fig. 1: Diagrama UML de la arquitectura del sistema.

`GameController` inicializa episodios, evalúa condiciones de éxito o fallo y administra las recompensas. Cada agente, mediante `CollectObservations`, obtiene información sobre su velocidad, la posición y orientación de la tabla, y actúa aplicando fuerzas en los ejes X y Z desde `OnActionReceived`.

El reinicio de episodio depende del sistema completo y no de cada agente por separado, ya que el éxito de la tarea requiere coordinación entre ambos.

4.4 Diseño de los agentes

Como se mencionó anteriormente, el comportamiento de cada agente se implementa en la clase `Robot`, la cual integra percepción, decisión y aprendizaje. Si bien cada agente opera de forma independiente en cuanto a su lógica de acción, el entorno y la tarea en sí los obligan a cooperar de forma implícita. Esta cooperación emerge del diseño conjunto de su espacio de observación, las acciones disponibles y el esquema de recompensas, el cual incentiva tanto el cumplimiento de metas individuales como de objetivos comunes.

La percepción del entorno se implementa a través del método `CollectObservations`, el cual recopila un vector de 12 dimensiones para cada agente. Este vector incluye información clave del estado del sistema, como la dirección relativa hacia la salida objetivo, la rotación actual de la tabla (considerando la componente Y del quaternion), la posición relativa de la tabla respecto al agente, la velocidad lineal del agente en los ejes X y Z, y la distancia euclidiana al objetivo. Estas observaciones permiten a los agentes construir una representación informativa del entorno, lo que les posibilita tomar decisiones razonadas en función de su situación actual. Para facilitar el proceso de entrenamiento y mejorar la estabilidad del modelo, todas las observaciones son normalizadas antes de ser procesadas por la red neuronal.

Espacio de acciones. El espacio de acción de cada agente consiste en valores continuos en el rango $[-1, 1]$, que representan las fuerzas aplicadas en los ejes X y Z del plano horizontal. Estas acciones se traducen en movimientos físicos mediante la aplicación de fuerzas al componente `Rigidbody` del agente, con una magnitud proporcional al atributo `fuerzaMovimiento` (15.0 N). Para garantizar estabilidad durante el aprendizaje, se impone un límite de velocidad máxima (5.0 unidades/segundo) que previene comportamientos erráticos.

Función de Recompensa. En este problema, los agentes deben transportar una pieza a través de un espacio reducido sin que esta se caiga ni se produzcan colisiones, manteniendo la estabilidad de la tabla que cargan en conjunto. Para fomentar tanto el progreso individual como la coordinación entre agentes, se ha diseñado una función de recompensa compuesta por múltiples términos que capturan distintos aspectos del desempeño.

La recompensa total que recibe cada agente en cada paso de simulación está dada por:

$$R_{\text{total}} = \frac{1}{1+d} - 0.01 \cdot \mathbb{I}_{\text{inactivo}} - \mathbb{I}_{\text{inestable}} - 0.5 \cdot \mathbb{I}_{\text{colisión}} + 10 \cdot \mathbb{I}\text{éxito} \qquad (3)$$

donde d es la distancia euclidiana entre el agente y la salida, $\mathbb{I}_{\text{inactivo}}$ vale 1 si la velocidad del agente es menor a 0.1 unidades/segundo, $\mathbb{I}_{\text{inestable}}$ vale 1 si la tabla está inclinada más de 10° en los ejes X o Z, $\mathbb{I}_{\text{colisión}}$ vale 1 si el agente colisiona con una pared, y $\mathbb{I}_{\text{éxito}}$ vale 1 cuando se completa correctamente la tarea de transportar la tabla a través del vano de luz. Esta función logra equilibrar penalizaciones por comportamientos no deseados (como inactividad, colisiones o inestabilidad) con incentivos por el progreso hacia el objetivo y la finalización exitosa de la tarea.

4.5 Implementación del aprendizaje

Se optó por utilizar el algoritmo Proximal Policy Optimization (PPO) para resolver la tarea, por su eficacia para resolver problemas con espacios de acción continua y su estabilidad en entornos multiagente (Schulman et al., 2017). La implementación se realizó mediante RLlib, que facilitó la gestión distribuida del entrenamiento.

Dado que el entorno de simulación se desarrolló en Unity y el entrenamiento se ejecutó en Python, se utilizó Ray como puente de comunicación entre ambas plataformas, permitiendo una correcta integración durante el proceso de aprendizaje.

Arquitectura e Hiperparámetros. Cada agente utiliza una red neuronal profunda con dos capas completamente conectadas de 256 neuronas cada una, empleando la función de activación ReLU para mejorar la representación de las características del entorno (Haykin, 2009). Las observaciones se normalizan automáticamente para estabilizar el aprendizaje y mejorar la convergencia. Durante la fase de entrenamiento se utilizan los siguientes hiperparámetros (Schulman et al., 2017): una tasa de aprendizaje $\alpha = 0.0003$, un factor de descuento $\gamma = 0.99$, una longitud de episodio de $T = 3000$ pasos, un tamaño de batch de 4000 muestras, un ratio de clipping de 0.2, y un gradiente máximo también limitado a 0.2 para evitar explosiones en la magnitud de los ajustes.

Configuración del entrenamiento. Para optimizar la eficiencia del aprendizaje, el entrenamiento se llevó a cabo con cuatro procesos paralelos, lo que permitió la recolección simultánea de experiencias y redujo el tiempo total de simulación. Se estableció un esquema de checkpoints cada 5 iteraciones, lo que permitió la reanudación del entrenamiento en caso de interrupciones. El entrenamiento se da por concluido al cumplirse alguno de los siguientes criterios: i) se alcanzaron 10 millones de pasos simulados; ii) la recompensa media obtenida por los agentes superó los 10.000 puntos; iii) se completaron 5.000 iteraciones de entrenamiento.

Estrategia Multi-agente. El comportamiento adoptado fue descentralizado, donde cada agente mantuvo su propia política de aprendizaje; no obstante, ambas políticas compartieron la misma arquitectura de red y el mismo conjunto de hiperparámetros. De esta manera, se logra que el aprendizaje de cada agente evolucione en función de su interacción con el entorno e independientemente del otro agente.

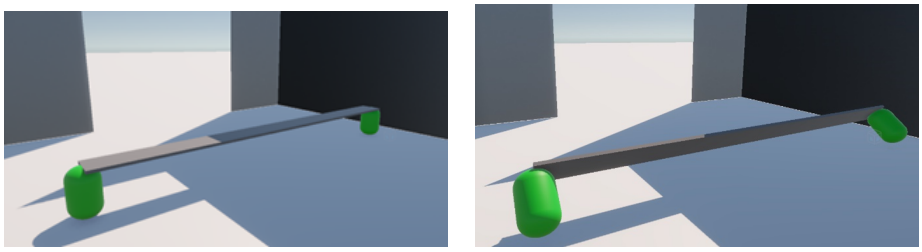
5 Experimentos

Esta sección describe la configuración y análisis de los experimentos realizados para evaluar el desempeño de agentes entrenados con MARL en tareas colaborativas de transporte. Primero se detallan el entorno de simulación, los recursos computacionales utilizados y las herramientas empleadas. Luego se analizan dos escenarios evaluados: uno base y otro con una carga adicional sobre la tabla, lo que incrementa la complejidad. Finalmente, se estudian métricas como recompensa media, tiempos de iteración, uso de recursos y trayectorias seguidas por los agentes.

Recursos computacionales. Los experimentos se realizaron en un equipo con procesador AMD Ryzen 7 3700X (8 núcleos, 16 hilos), 32 GB de RAM DDR4 a 3200 MHz, GPU AMD Radeon RX 6700 XT (12 GB), almacenamiento de 2 TB SSD + 3 TB HDD, y sistema dual Windows 10 Pro 22H2 y Linux con soporte ROCm para GPU AMD. Esta configuración permitió ejecutar múltiples simulaciones en paralelo, acelerando la recolección de experiencias y reduciendo significativamente los tiempos de entrenamiento.

Configuración de los experimentos. Se definieron dos escenarios con distinta dificultad. En ambos, los agentes deben cooperar para trasladar una tabla desde un punto inicial hasta cruzar una abertura. La tarea requiere aplicación de fuerza simultánea, ya que un solo agente no puede mover la carga. El entorno incluye física realista (peso, fricción y colisiones), por lo que la falta de coordinación puede provocar la caída de la tabla. En ese caso, el episodio finaliza y los agentes reciben una penalización máxima.

Escenario 1: Manipulación de una pieza simple. El primer escenario consiste en trasladar únicamente la pieza con forma de tabla. Este experimento implementa el entorno base descrito en la metodología, donde dos agentes deben trasladar una tabla rígida de 15 unidades de largo a través de una salida de 10 unidades de ancho (Figura 2a). La simulación evalúa en tiempo real la dinámica del sistema, considerando fuerzas y colisiones. Dado que la tabla solo puede moverse si ambos agentes aplican fuerzas coordinadas, los errores en la sincronización pueden generar oscilaciones o incluso la caída completa de la tabla (Figura 2b), lo que implica el reinicio del entorno.



(a) Transporte coordinado. (b) Caída de la tabla.

Fig. 2: Escenario 1: transporte de una pieza simple.

Escenario 2: Manipulación de una carga mixta. En este caso, de mayor complejidad que el anterior, se debe transportar la tabla con una caja encima (de 1 ul por lado y masa $m = 1$ kg), como muestra la Figura 3, lo que agrega un desafío extra en términos de coordinación de las acciones.

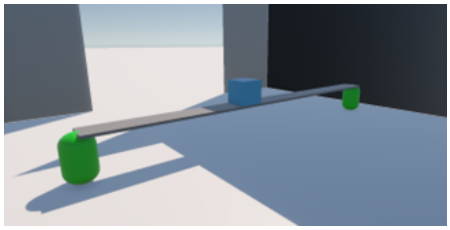


Fig. 3: Transporte coordinado con carga.

5.1 Análisis de resultados

El desempeño de los agentes en ambos escenarios se evaluó a partir de tres métricas principales: la recompensa media obtenida por episodio, el tiempo de iteración y el uso de CPU. Estas métricas permiten analizar tanto la convergencia del modelo como el impacto computacional de cada entorno.

En la Figura 4, se puede observar una clara tendencia a la convergencia de las políticas de los agentes. En el escenario sin carga, la estabilización se alcanza en unos 125 episodios, con un retorno promedio cercano a los 1000 puntos. En cambio, el escenario con carga requiere cerca de 210 episodios para estabilizarse.

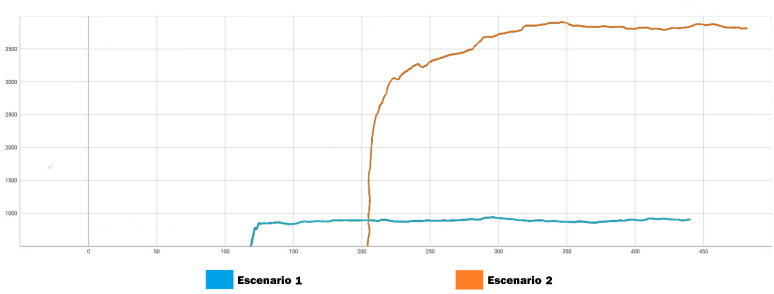


Fig. 4: Recompensa media por episodio.

Respecto al tiempo de iteración, se pudo observar un comportamiento relativamente estable, entre 60.000 y 61.000 ms por episodio, con ligeras variaciones atribuibles a la complejidad del entorno. Finalmente, el uso de CPU se mantuvo entre el 25% y 40%, con un incremento notorio durante la ejecución del segundo escenario, que exige mayor control y coordinación por parte de los agentes.

Comportamiento emergente En ambos escenarios, los agentes lograron desarrollar estrategias colaborativas sin intervención explícita . Esto evidencia la aparición de comportamiento emergente, entendido como la capacidad de los agentes para generar dinámicas colectivas complejas a partir de políticas individuales simples, entrenadas mediante refuerzo.

En el escenario sin carga, los agentes aprenden rápidamente a sincronizar sus movimientos para mantener la tabla equilibrada, evitando oscilaciones que llevarían al reinicio del episodio. Este comportamiento no fue predefinido, sino

que emergió a partir de la retroalimentación del entorno y la necesidad de maximizar la recompensa.

En el escenario con carga, el comportamiento emergente es más complejo: además de mantener la sincronización lateral, los agentes ajustan la intensidad y dirección de la fuerza aplicada para evitar que la caja caiga de la tabla. Este ajuste de fuerzas, surge como resultado de la experiencia acumulada en múltiples episodios, donde errores sutiles son penalizados al provocar la caída de la carga.

En ambos casos, se observa que los agentes desarrollan una forma de "comunicación implícita" basada en la respuesta dinámica del objeto compartido. Es decir, al reaccionar a los cambios en la posición de la tabla o de la caja, cada agente modula su comportamiento como respuesta al del otro, sin intercambiar mensajes explícitos. Esta coordinación emergente sugiere que es posible alcanzar formas de cooperación efectivas en tareas físicas compartidas mediante MARL, incluso en ausencia de canales de comunicación directa. La Figura 5 y la Figura 6. ilustran una secuencia típica del comportamiento aprendido en el escenario sin carga y con carga, respectivamente.

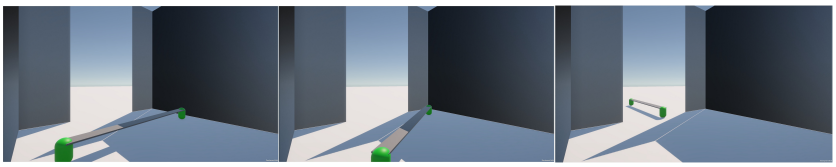


Fig. 5: Secuencia, transporte coordinado.

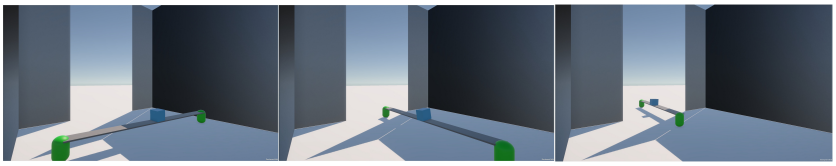


Fig. 6: Secuencia, transporte coordinado con carga.

6 Conclusiones

Este trabajo exploró el potencial del MARL para abordar tareas colaborativas típicas de entornos de manufactura flexible, en particular aquellas relacionadas con la manipulación y el transporte de materiales. A través del desarrollo de un entorno simulado y la implementación de dos escenarios de distinta complejidad, se demostró que es posible entrenar agentes capaces de coordinar sus acciones para alcanzar objetivos comunes, incluso bajo restricciones físicas como el transporte de una carga compartida.

Los resultados obtenidos, tanto en términos de métricas como de comportamiento emergente, evidencian que los agentes fueron capaces de aprender políticas efectivas sin necesidad de un controlador centralizado. Esto refuerza la idea de que los enfoques MARL constituyen una estrategia viable para enfrentar la creciente necesidad de flexibilidad, autonomía y adaptabilidad

en sistemas de producción modernos, particularmente en el caso de PyMEs que deben reconfigurar sus procesos de forma constante.

Si bien los experimentos se realizaron en un entorno virtual y bajo ciertas simplificaciones, los aprendizajes obtenidos sientan las bases para futuras líneas de investigación orientadas a trasladar estas estrategias a sistemas físicos reales. Entre las posibles extensiones del trabajo se destacan la incorporación de más agentes, el manejo de múltiples objetos con diferentes propiedades físicas, y la inclusión de mecanismos de comunicación explícita entre agentes, con el fin de escalar la complejidad de las tareas y aproximarse a escenarios más realistas propios de la industria 4.0.

References

Albrecht, S. V., Christianos, F., & Schafer, L. (2024). *Multi-agent reinforcement learning: Foundations and modern approaches*. MIT Press.

Boggino, A. S. G. (2005). *Anémona: Una metodología multi agente para sistemas holónicos de fabricación* [Doctoral dissertation].

Chen., Cheng, C., & Li, J. (2018). Resource-constrained assembly line balancing problems with multi-manned workstations. *Journal of Manufacturing Systems*, *48*, 107–119.

Chen, X., Chen, R., & Yang, C. (2022). Research to key success factors of intelligent logistics based on iot technology. *Journal of Supercomputing*, *78*, 3905–3939.

Coumans, E., & Bai, Y. (2016). Pybullet, a python module for physics simulation for games, robotics and machine learning.

Curşeu, P. L., Rusu, A., Maricuţoiu, L. P., Vîrgă, D., & Măgurean, S. (2020). Identified and engaged: A multi-level dynamic model of identification with the group and performance in collaborative learning. *Learning and Individual Differences*, *78*.

Durão, L. F. C. S., McMullin, H., Kelly, K., & Zancul, E. (2022). Manufacturing execution system as an integration backbone for industry 4.0. *IFIP Advances in Information and Communication Technology*, *639*, 461–473.

Foerster, J. N., Farquhar, G., Afouras, T., Nardelli, N., & Whiteson, S. (2018). Counterfactual multi-agent policy gradients. *32nd AAAI Conference on Artificial Intelligence*, 2974–2982.

Haykin, S. (2009). *Neural networks and learning machines* (Third). Pearson.

Ilosvay, V. B., & Iaccarino, E. (2024). Unity ml agents: Wall jump and soccertwos environment using reinforcement learning (rl) technique.

Juliani, A., Berges, V.-P., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., & Lange, D. (2018). Unity: A general platform for intelligent agents.

Koenig, N., & Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, *3*, 2149–2154.

Mantravadi, S., Li, C., & Møller, C. (2019). Multi-agent manufacturing execution system (mes): Concept, architecture & ml algorithm for a smart factory case. *ICEIS 2019 - Proceedings of the 21st International Conference on Enterprise Information Systems*, *1*, 465–470.

Oliehoek, F. A., & Amato, C. (2016). *A concise introduction to decentralized pomdps*. Springer.

Quintero Henao, L. F. (2009). *Un modelo de control inteligente para sistemas de manufactura basado en los paradigmas holónico y multi-agente* [Doctoral dissertation, Universidad Nacional de Colombia].

Rashid, A., Danezis, G., Chivers, H., Lupu, E., Martin, A., Lewis, M., & Peersman, C. (2018). Scoping the cyber security body of knowledge. *IEEE Security & Privacy*, *16*(4), 96–102.

Saavedra Sueldo, C., Perez Colo, I., De Paula, M., Villar, S. A., & Acosta, G. G. (2023). Ros-based architecture for fast digital twin development of smart manufacturing robotized systems. *Annals of Operations Research*, *322*(1), 75–99.

Saavedra Sueldo, C., Perez Colo, I., De Paula, M., Villar, S. A., & Acosta, G. G. (2024). Simulation-based metaheuristic optimization algorithm for material handling. *Journal of Intelligent Manufacturing*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms.

Schwung, D., Reimann, J. N., Schwung, A., & Ding, S. X. (2018). Self learning in flexible manufacturing units: A reinforcement learning approach. *9th International Conference on Intelligent Systems 2018: Theory, Research and Innovation in Applications*, 31–38.

Shoham, Y., & Leyton-Brown, K. (2008). *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*.

Smith, R. G. (1980). The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, *29*(12).

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd). MIT Press.

Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L., & Peeters, P. (1998). Reference architecture for holonic manufacturing systems: Prosa. *Computers in Industry*, *37*, 255–274.

Velastegui, R., Poler, R., & Díaz-Madroñero, M. (2023). Aplicación de algoritmos de aprendizaje automático a sistemas robóticos multiagente para la programación y control de operaciones productivas y logísticas: Una revisión de la literatura reciente. *Direccion y Organizacion*, *80*, 60–70.

Wang, C., Kim, Y. S., & Kim, C. Y. (2021). Causality between logistics infrastructure and economic development in china. *Transport Policy*, *100*, 49–58.

Zhang, M., Li, P., Xia, Y., Wang, K., & Jin, L. (2021). Labeling trick: A theory of using graph neural networks for multi-node representation learning. *Advances in Neural Information Processing Systems*, *11*, 9061–9073.